BEHAVIOR DRIVEN DESIGN CRASH COURSE

Leeland Artra
http://nodsw.com

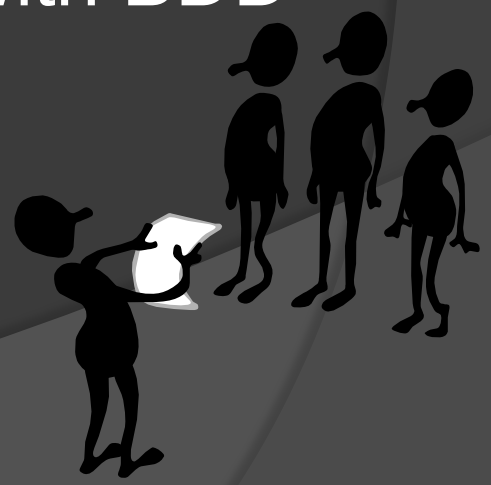
Present to Seattle Java Users Group November 16, 2010

Audience Litmus Test

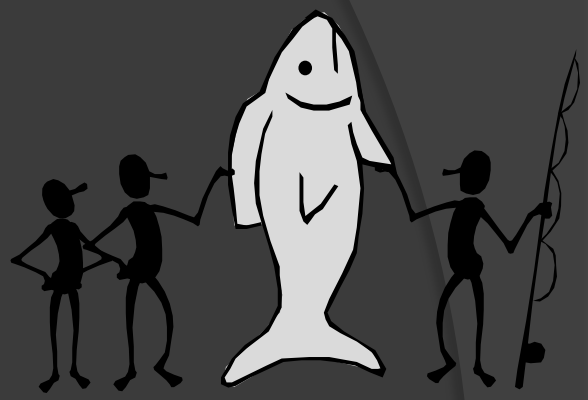
Raise Your Hands if you have:

- >1 year with Agile Development processes
- >1 year with actual TDD
- >6 months with actual BDD
- Zero or practically no experience with BDD

Any Managers in the audience?



Take Home For Managers



- Return on Training Investment
  - Requires +30% Investment in TIME

"Back" ≠ "Engaged"

- Why Spend $5000 with a $0 ROI?
  - Course $2000 (seminar fee)
  - Travel $1000 (hotel, travel, per diem)
  - 1 week $2000 (payroll)

- Give directly equivalent time on return
  - 1 week training = 1 week to put the pieces into play
  - 1.5 x T for larger conferences

Course Goals

- Understand affects of rigorous BDD on software development
- Understand core BDD concepts and terminology
- Provide guidance for doing BDD
  - Design patterns
  - Knowing when there is enough testing
  - References for further improvements

"Nothing is as simple as we hope it will be."
– Jim Horning

Where This Came From

- Books
  - Design Driven Testing
  - Pragmatic Unit Testing
- Internet Articles
  - All of C2 on BDD
  - Lots of blogs (Dan North, Scott Ambler, etc.)
- Technical Journals
  - Dr. Dobbs
- Experiences
  - Road of Hard Knocks
- With much thanks to 5-Hour Energy

(see references at the end)

"Good judgment comes from experience; experience comes from bad judgment."

- Jim Horning

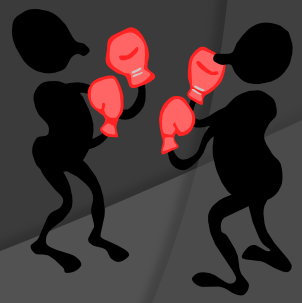
Agenda

- Development challenges
- Little Project Management Theory
- BDD basics
- Behavior Types
- BDD Examples
- Summary and Future Work

Ask questions at any time

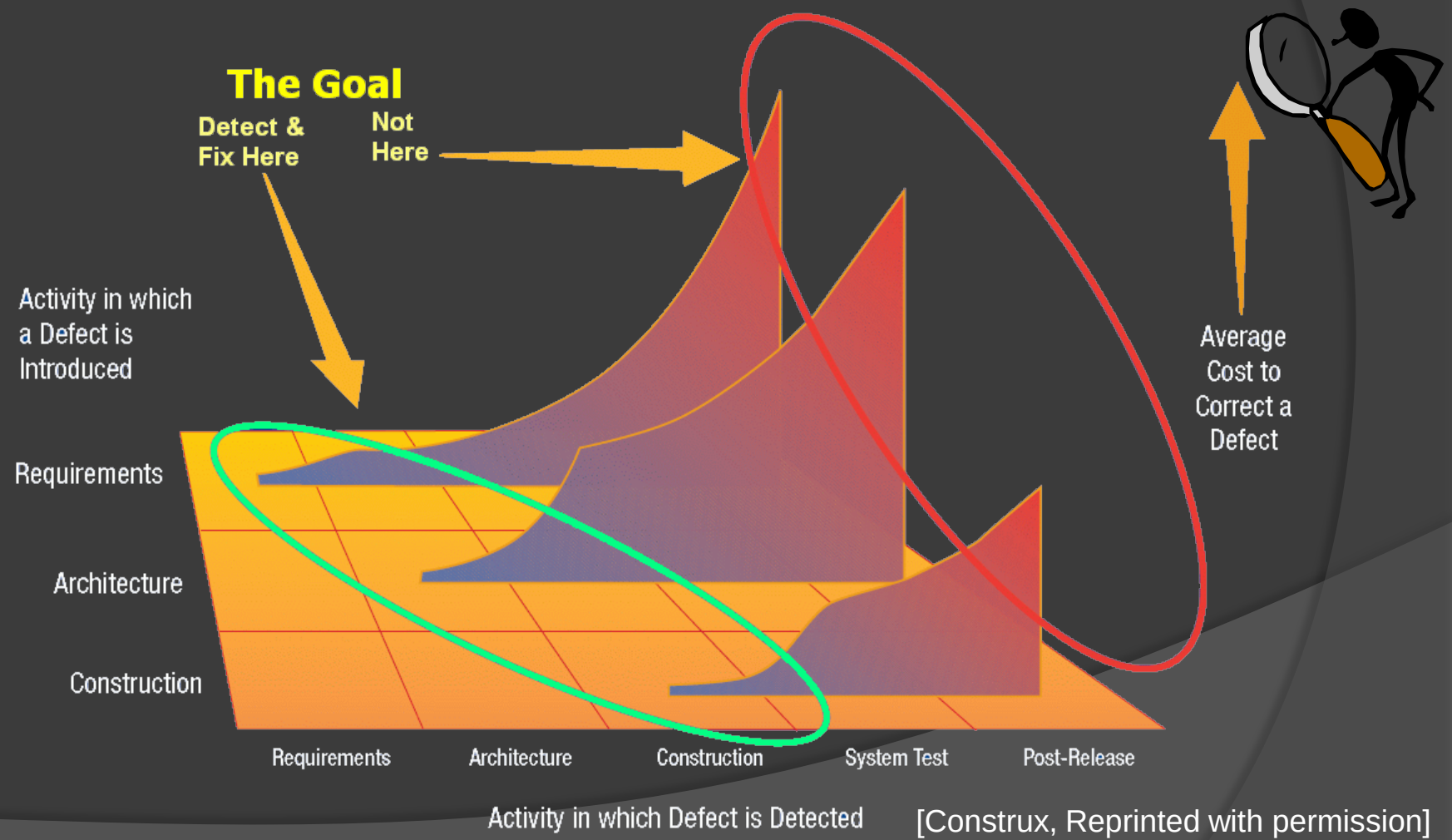
Development Challenges

- In 10 Minutes
- Hot button topics
  - Please debate out of band



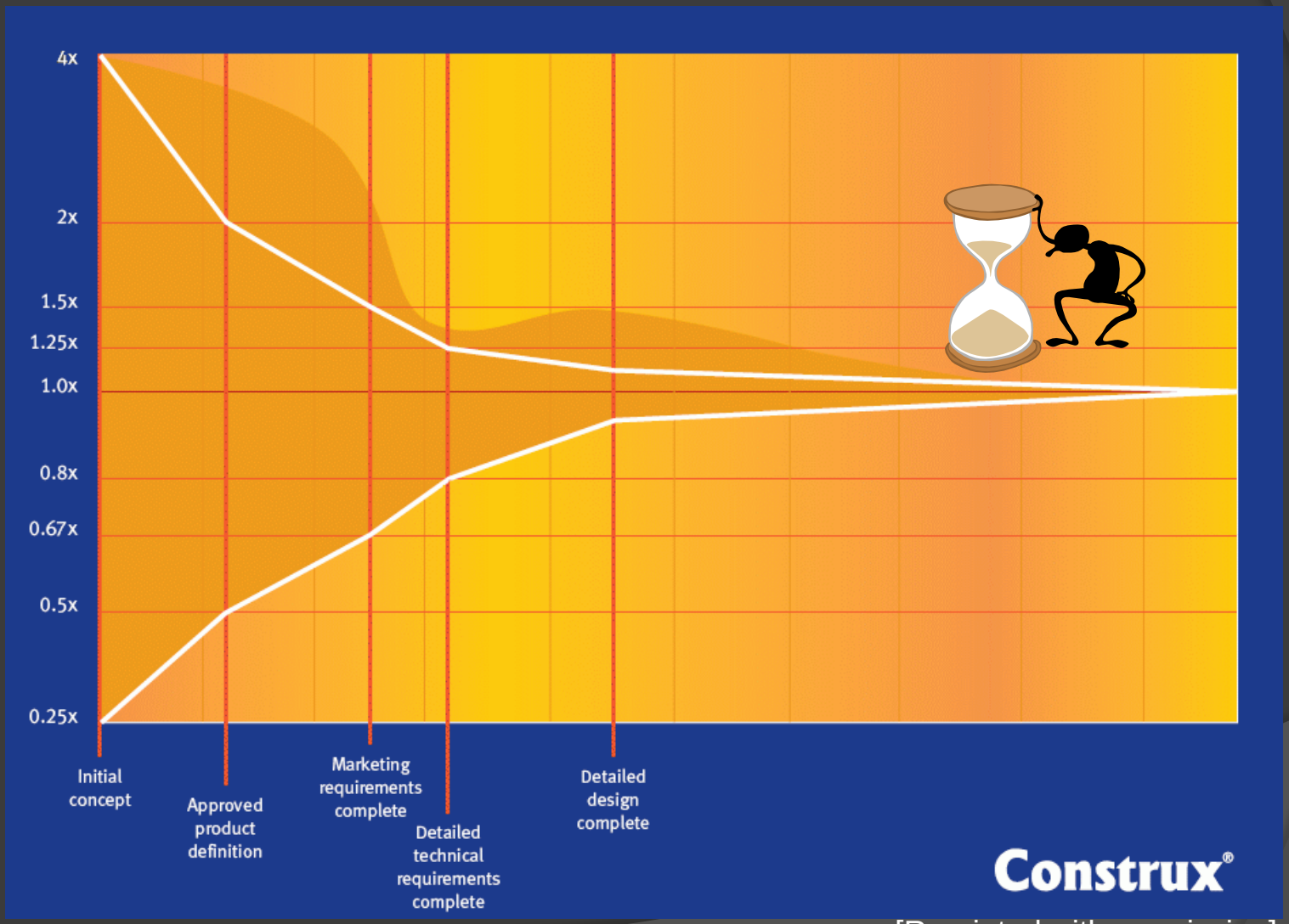
Challenge #1
Cost of Defect Correction

Upstream defects cost 50-200x as much to correct downstream



[Construx, Reprinted with permission]

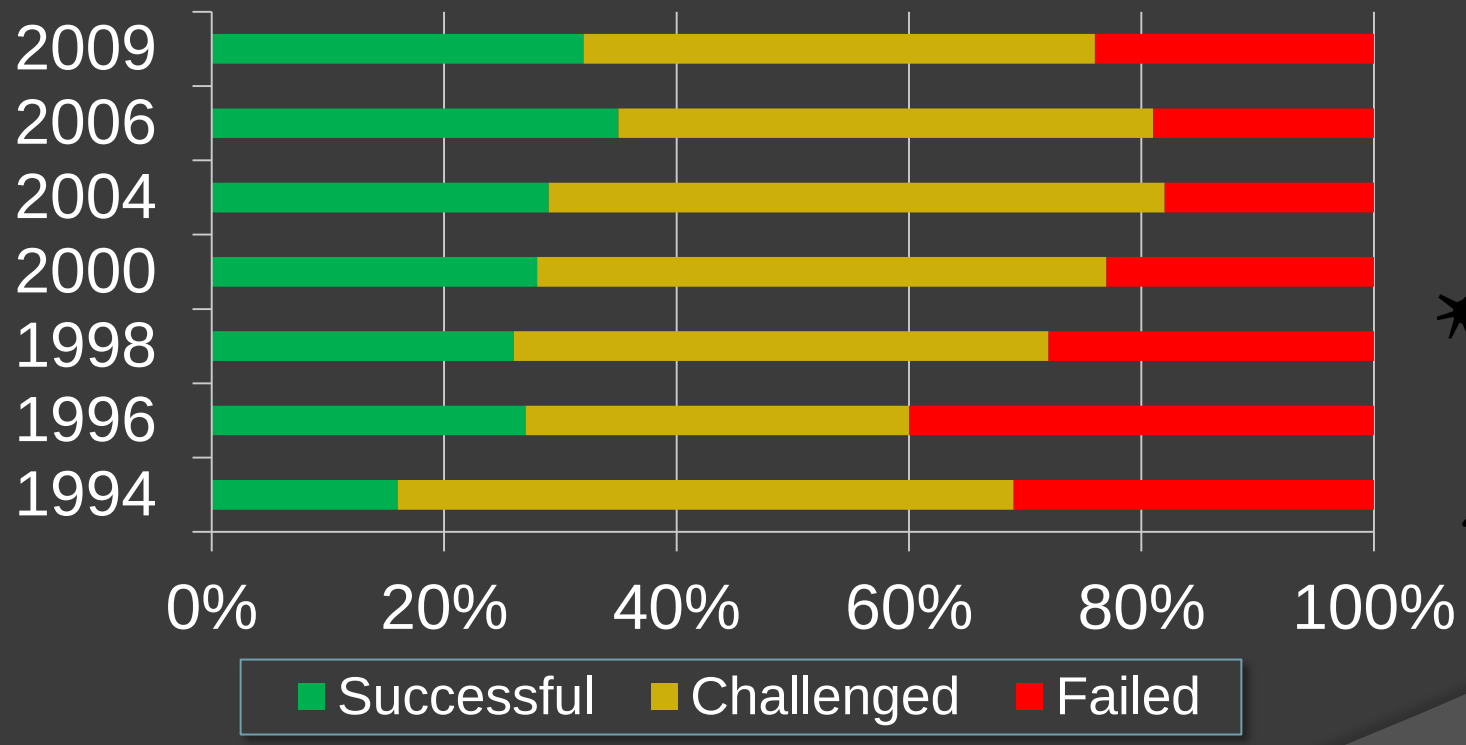
Challenge #2 Uncertainty



[Reprinted with permission]

Performance Results So Far
Project Failures (CHAOS Report)

WHAT ARE WE DOING WRONG!?

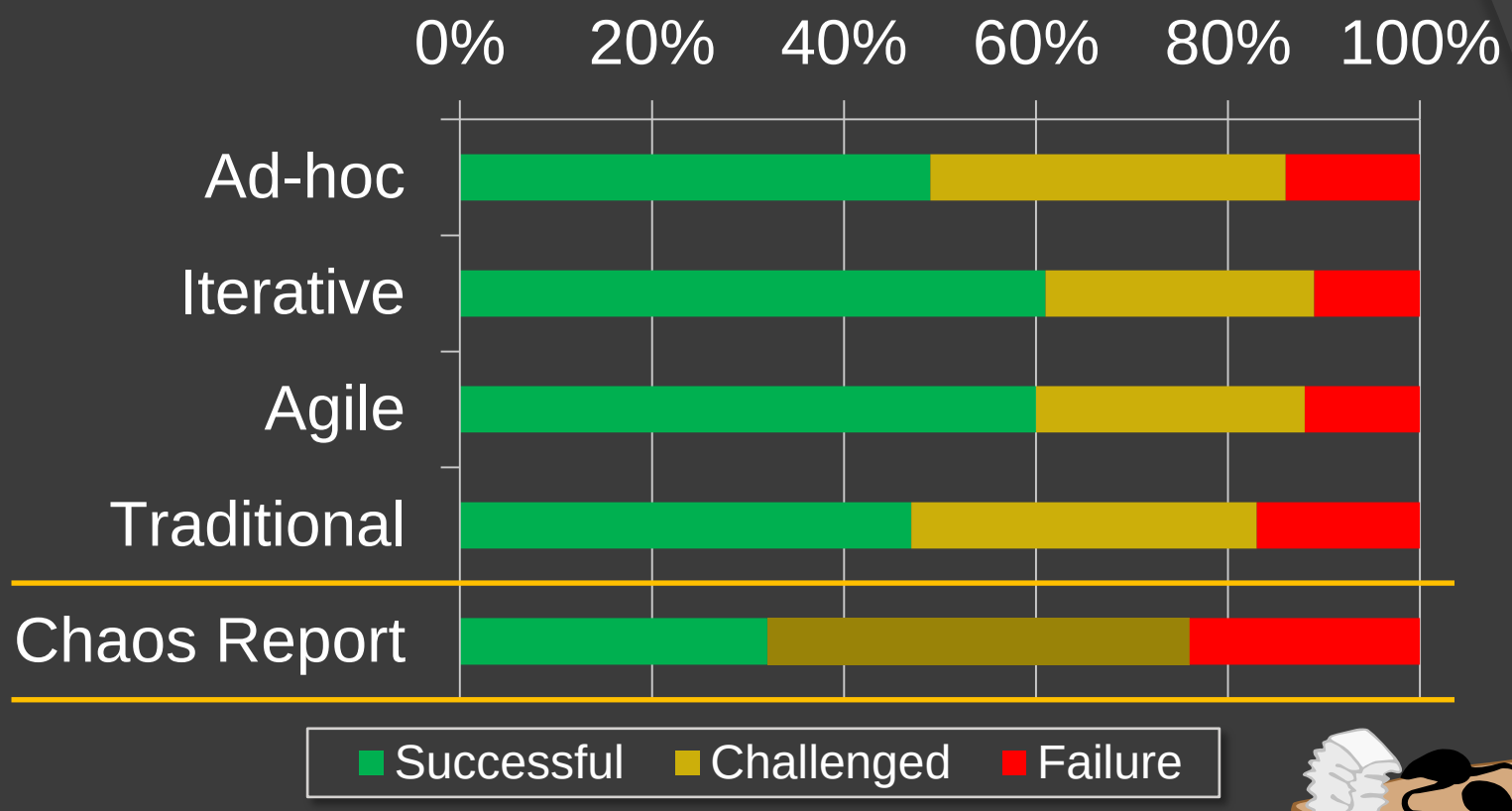


[Eveleens, J. and Verhoef, C. 2010]

Wait, if this is true why do we still have jobs?

Because, these are not saying what you think

Not So Bad - Still a Challenge

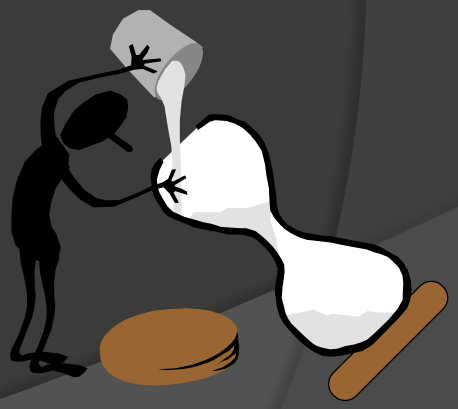


Note: Accurate to within +/-7%
Figures "normalized" to add to 100%

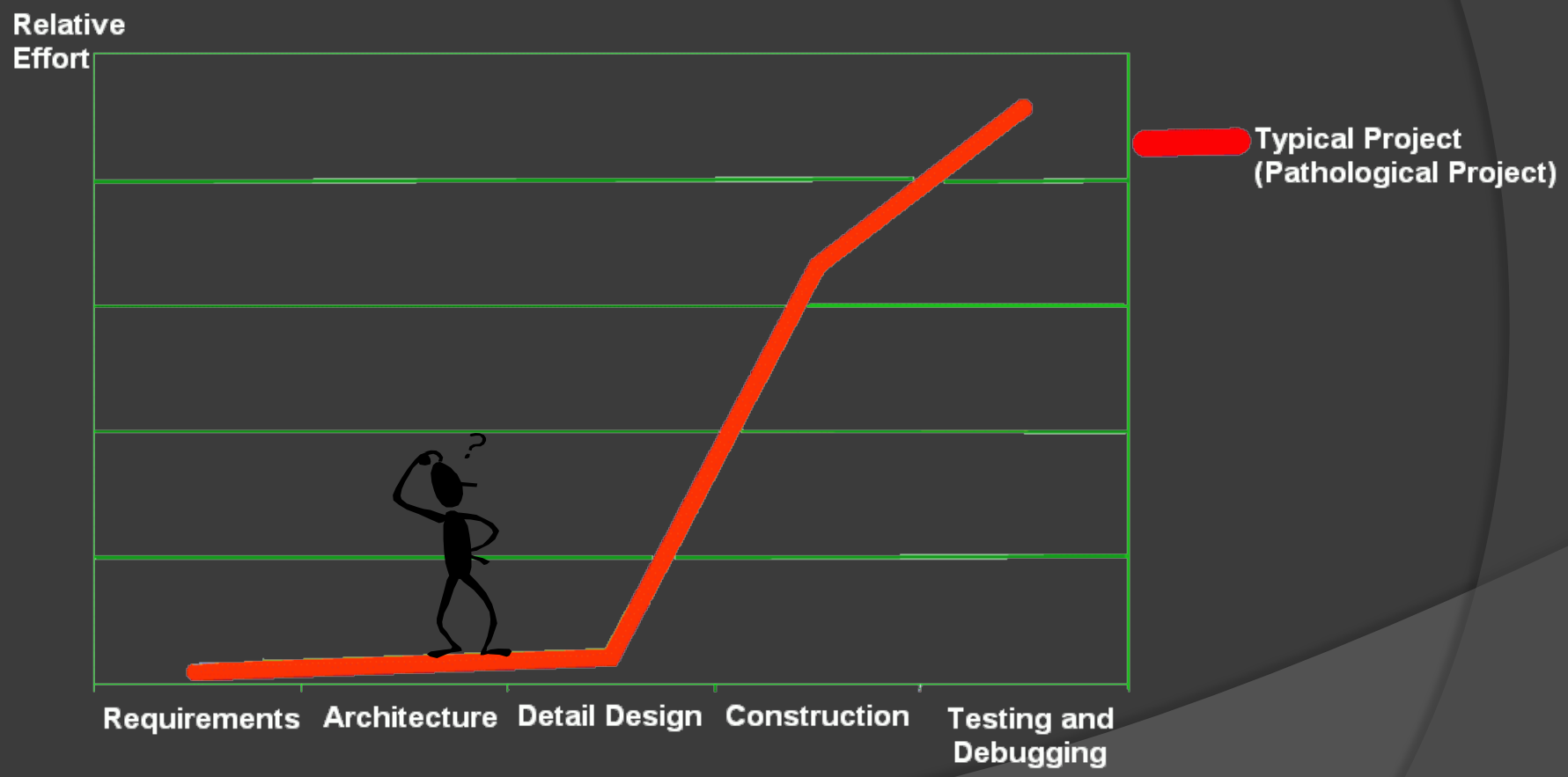
[Ambler, S. 2010]

Take Home #1

- **50-60% Success Rate**
  - Short Duration Projects
  - Iterative Delivery Projects
    - *A Chain of Short Duration Projects*
- **Regardless of Project Style**
  - **~32% Challenged Rate**
  - **~12% Failure Rate**
- **Why?**



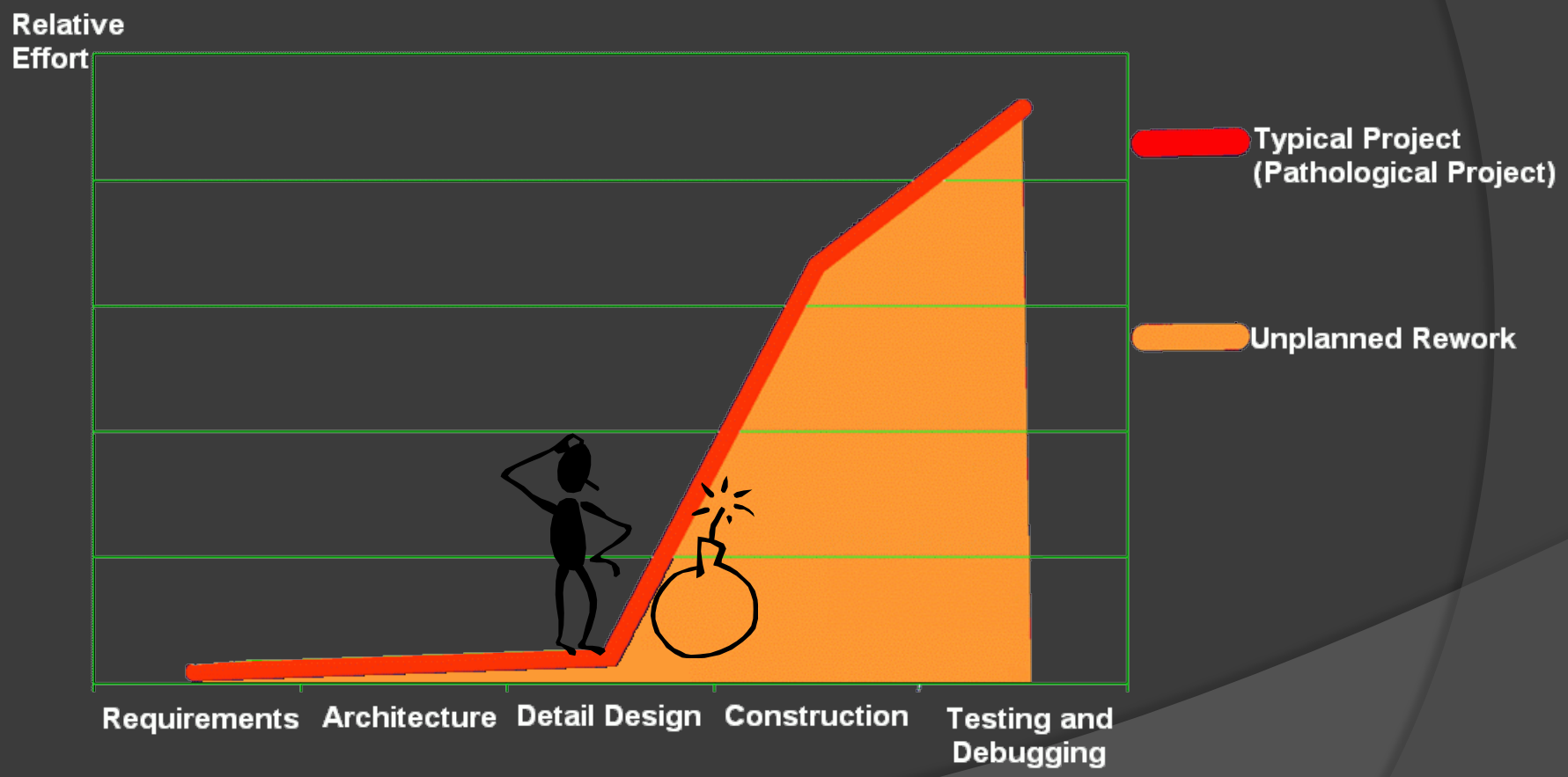
More Than 50% IT Projects Have Increasing Effort



[McConnell, S. 1998]

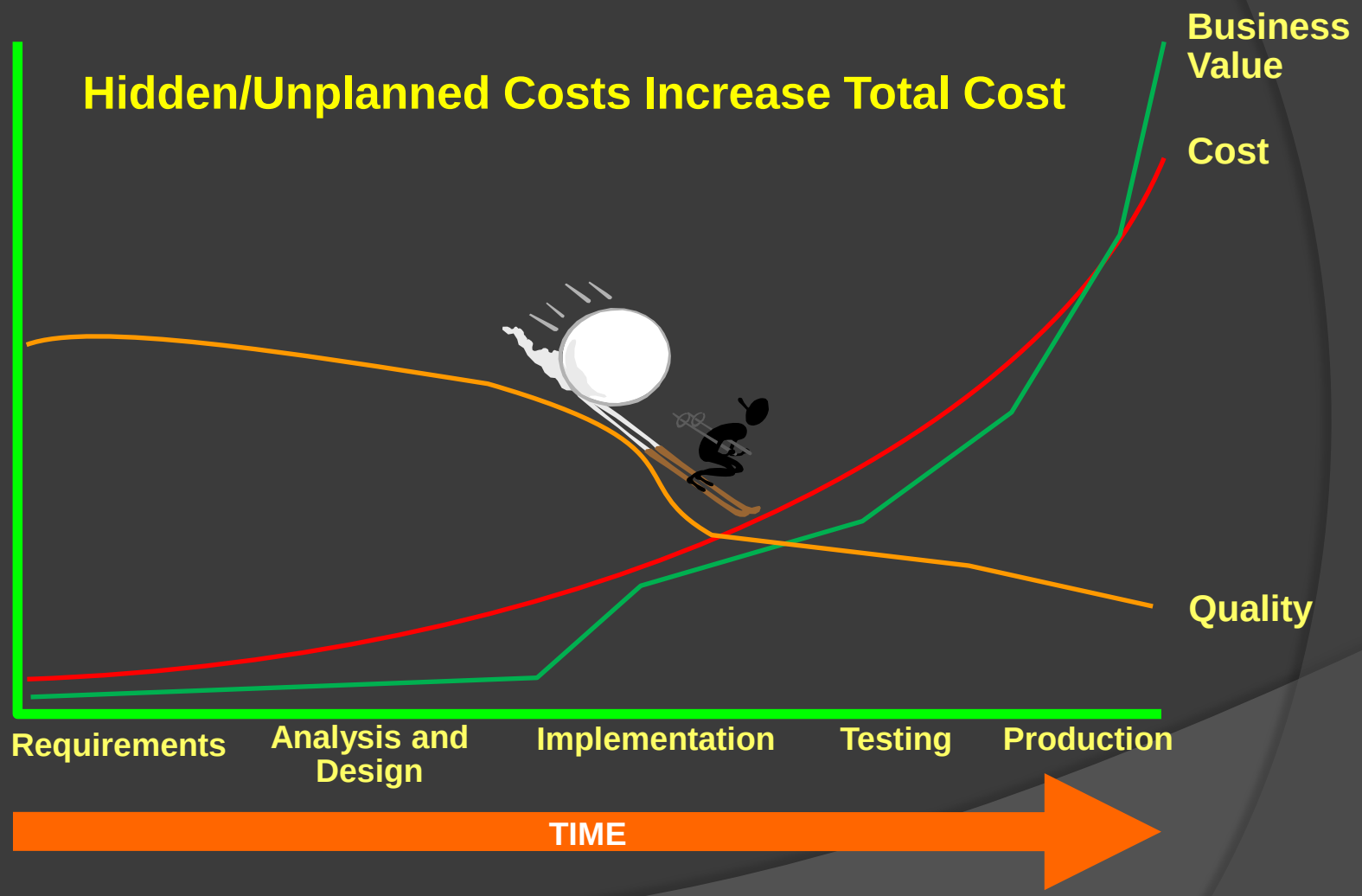
Unplanned Work Causes Waste

Late Defect Detection = High Effort Corrections



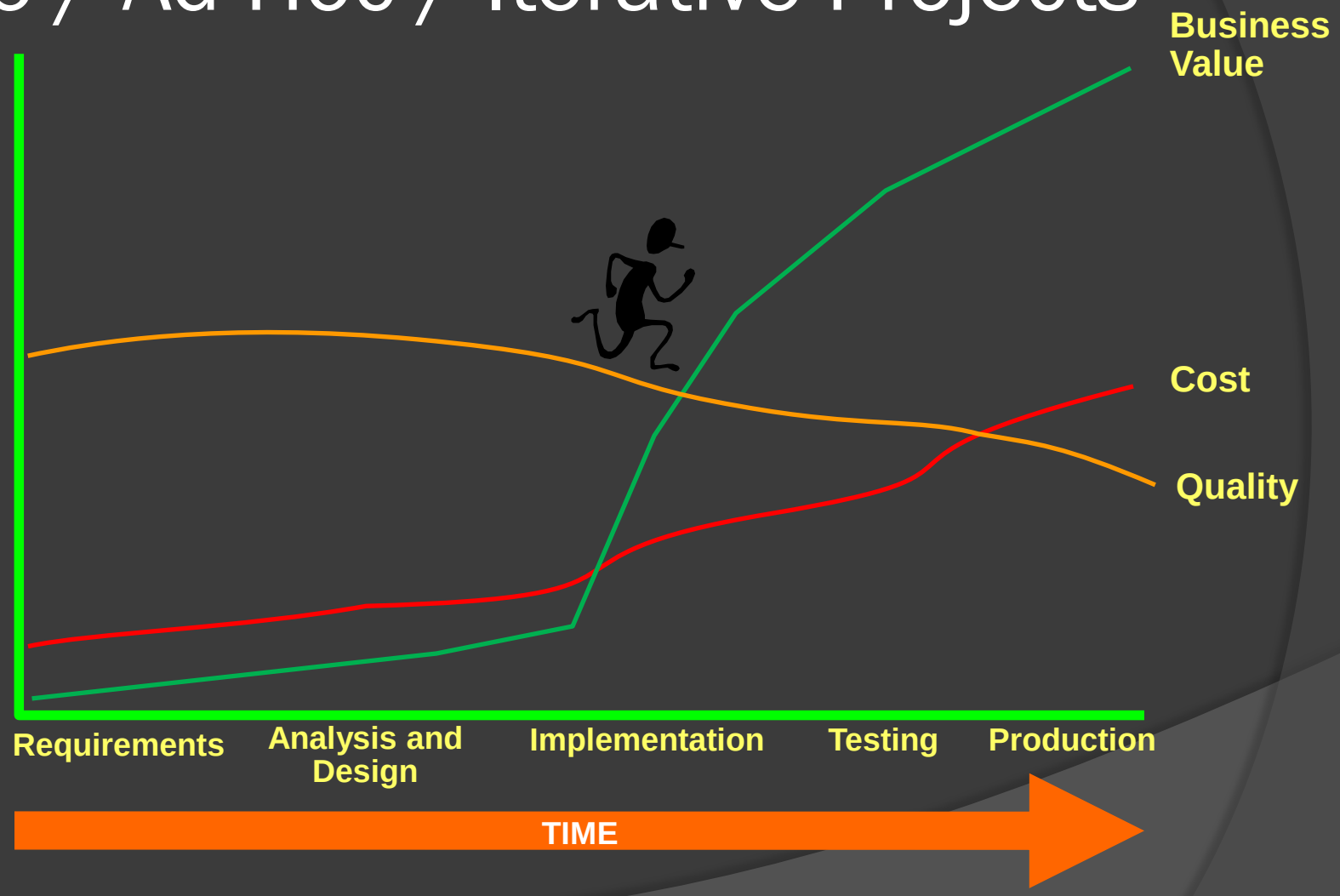
[McConnell, S. 1998]

Typical Development Project



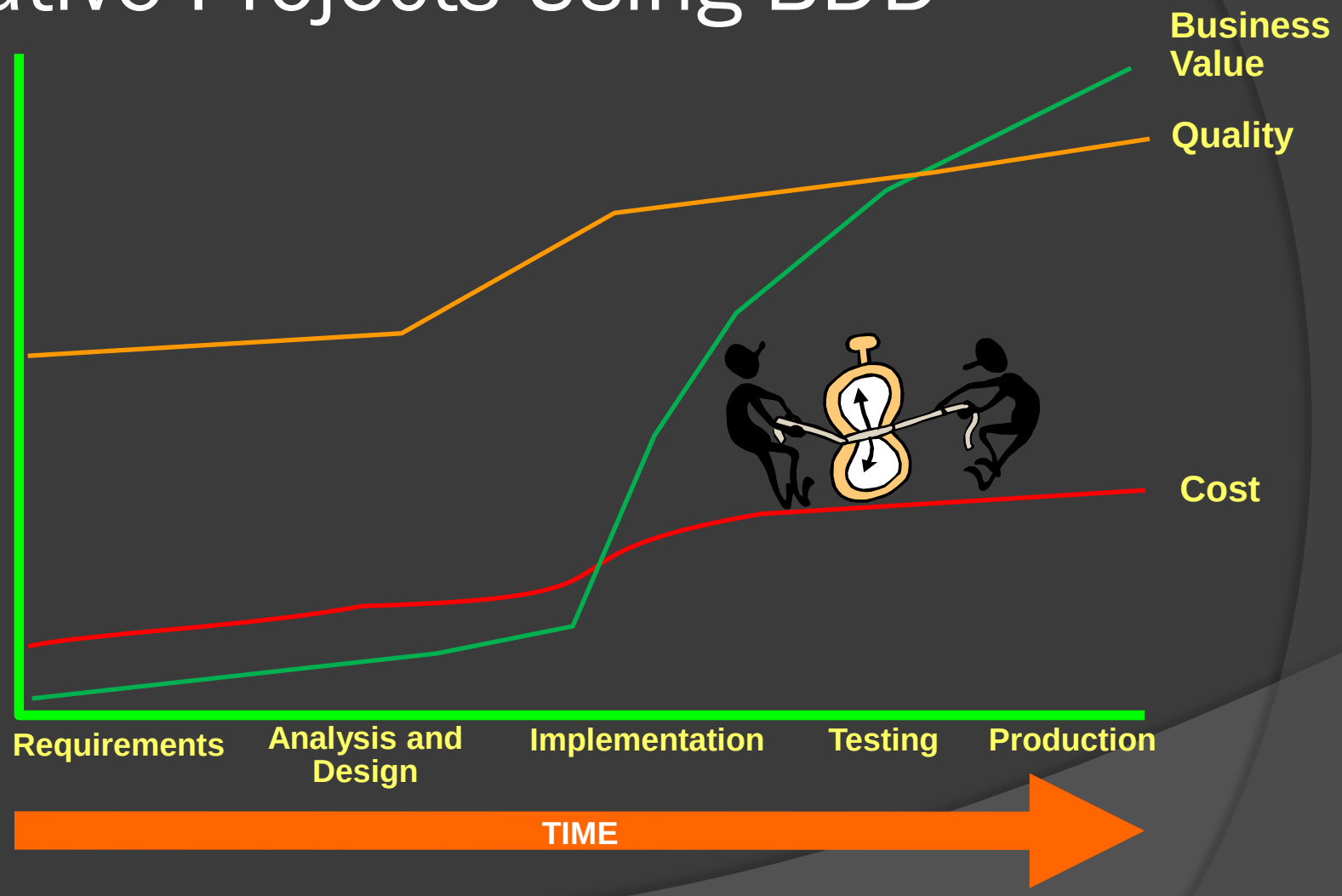
[McConnell, S. 1998] [O'Connell, F. 2001]

Goal : Higher Profit Projects
Agile / Ad-Hoc / Iterative Projects



[Ambler, S. 2004] [O'Connell, F. 2001]

Goal : Higher Profit Projects Iterative Projects Using BDD

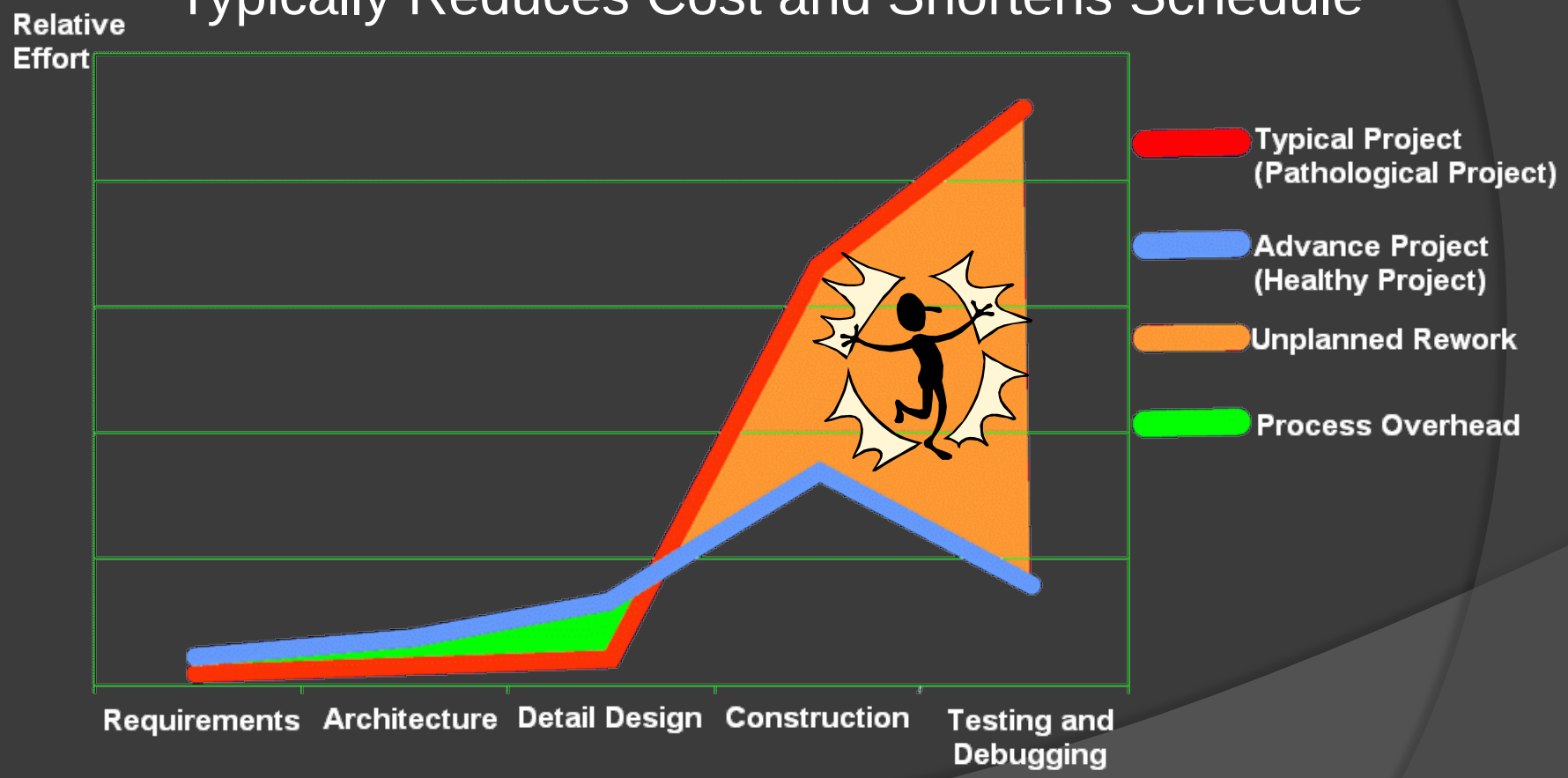


[North, D. 2005]

BDD ROI is Less Unplanned Work

Controlled Studies Demonstrate Focus On Quality

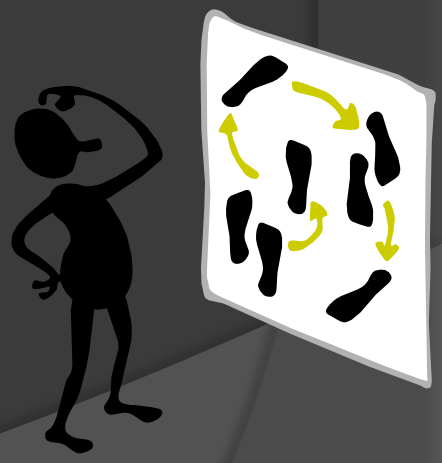
Typically Reduces Cost and Shortens Schedule



[McConnell, S. 1998] [Eveleens, J. and Verhoef, C. 2010] [Ambler, S. 2010]

Challenges To TDD?

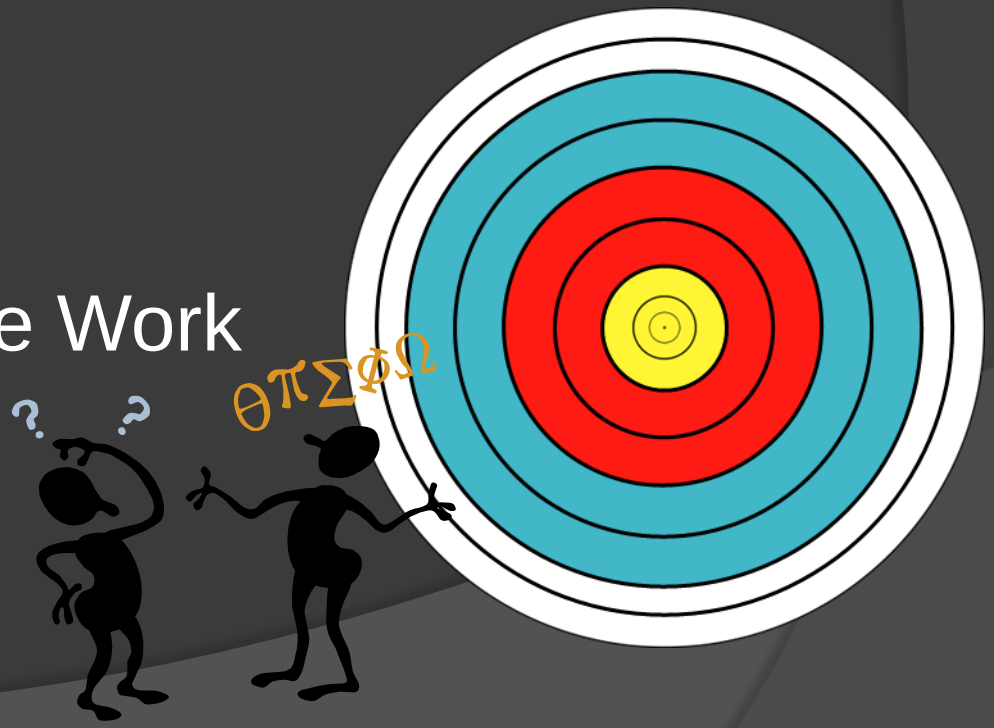
- Where to start?
- What to test?
- What not to test?
- How much to test?
- What to call the tests?
- Usually done too granularly and in a programmer's vernacular
- Really bad taste to the word "test"



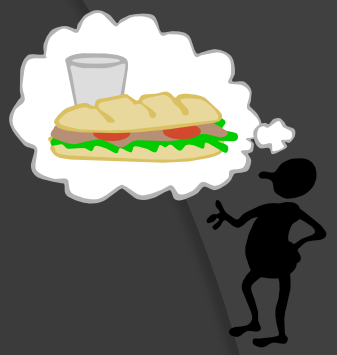
Development Challenges

Questions?

- Development Challenges
- Little Project Management Theory
- BDD basics
- Behavior Types
- BDD Examples
- Summary and Future Work



Natural Planning Model



Lets Eat Out

"Idea"

Why?

Define Purpose & Principles

satisfy hunger, socialize with friends, celebrate a special occasion, sign a business deal, or develop a romance

What?

Envision Outcome

"Italian food at Marcello's" or "Sitting at a sidewalk table at Pike Street Market."

How?

Brainstorming

"What time should we go?" "Is it open tonight?" "Will it be crowded?" "What's the weather like?" "Should we change clothes?" "Is there gas in the car?" "How hungry are we?"

When?

Organizing

"First we need to find out if the restaurant is open" and "Let's call the John and see if he'd like to go out with us." "We should send an email." (i.e. logistics, people, places and resources plus sequences 1st, 2nd, 3rd)

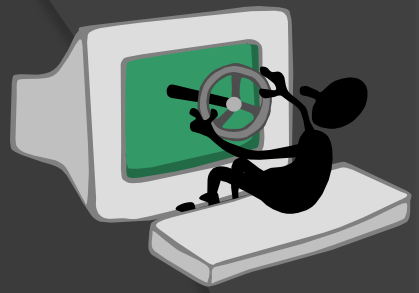
Action

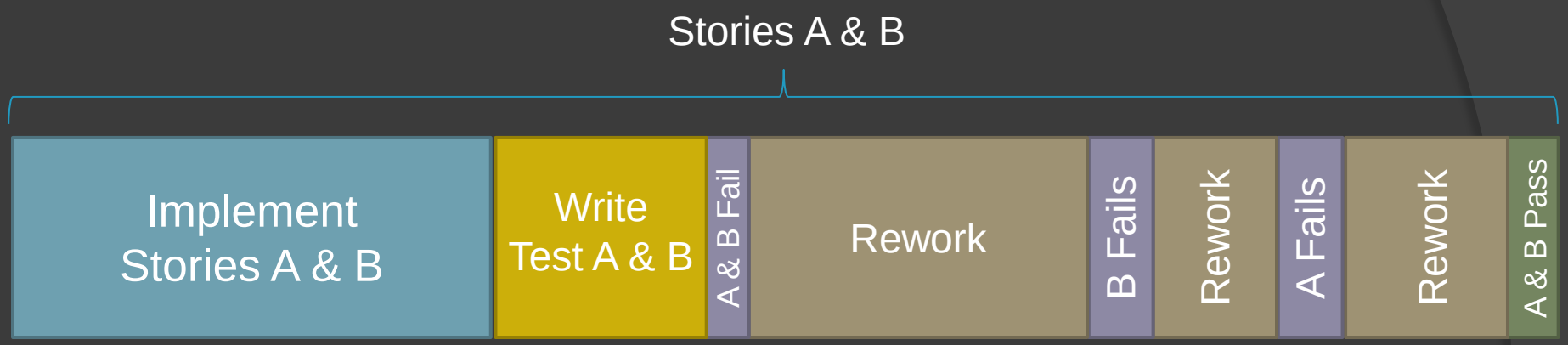
Identify Next Actions

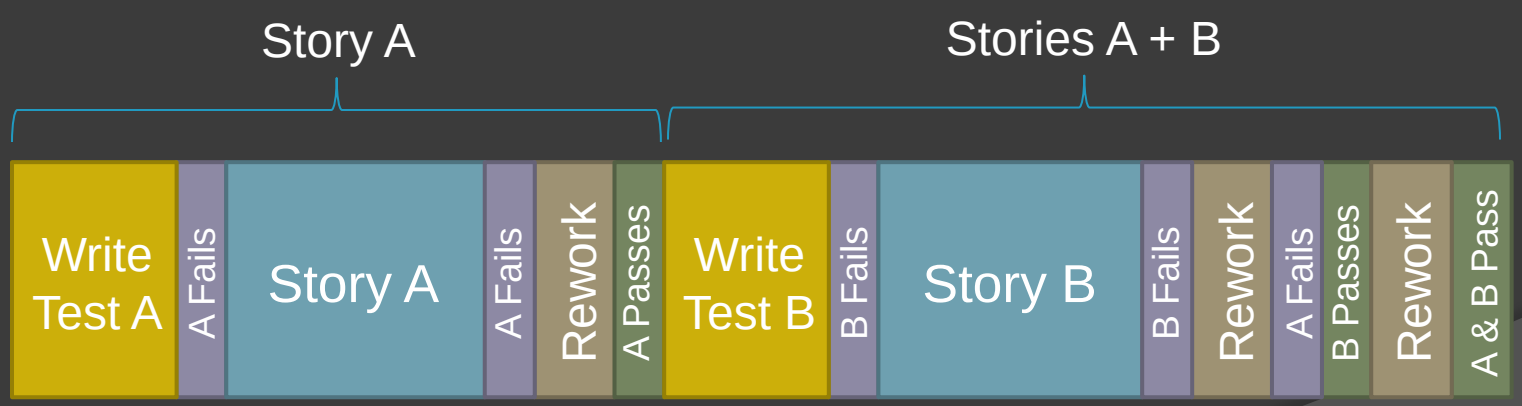
"Call Stella's to see if it's open, and make the reservation."

Development Styles

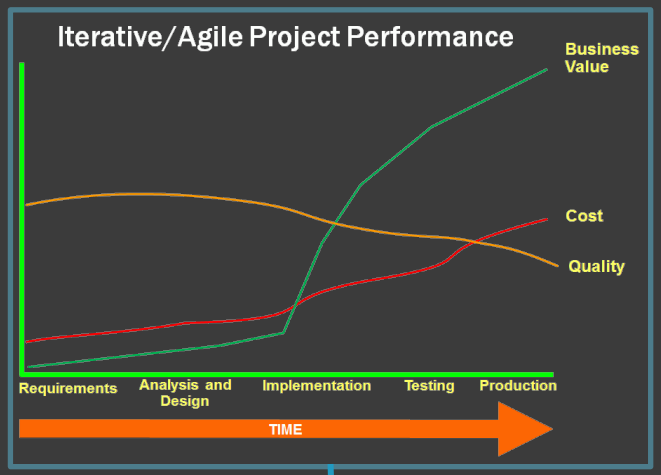
Test – Last vs. Test – First

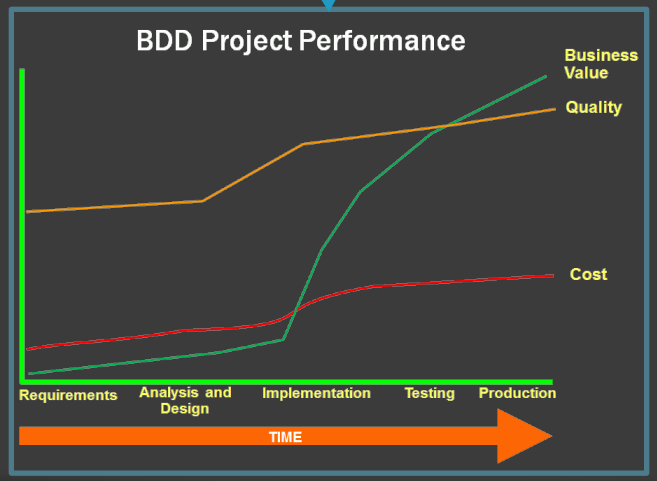




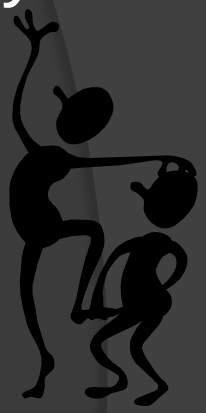


How Can BDD Reduce Cost AND Increase Quality?





- Increasing Quality Normally Means
  - More time on design
  - More time on testing
  - Rigorous process controls



- Reduced Cost Normally Means
  - Less resources
  - Shorter dead lines
  - Reduction of meetings, controls, and policing

Project Management Research Shows

- Project Pressure Points Are
  1. Scope
  2. Quality
  3. Resources (People, Cost, Tools / Techniques)
  4. Time Allowed

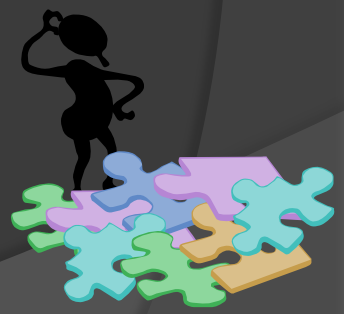
- **Successful Projects**
  - *Lead Controlled 2-3 primary pressure points*
- **Failed Projects**
  - *Lead Controlled < 2-3 primary pressure points*

[O'Connell, F. 2001]

Leeland's Law of Development

A project will only succeed if its fixed success constant (SC) is less then the function of the limited inputs of *Cost*, Energy and Genius reduced by the function of ***build Time*** (***Tb***) reduced by the max Time allowed (Ta).

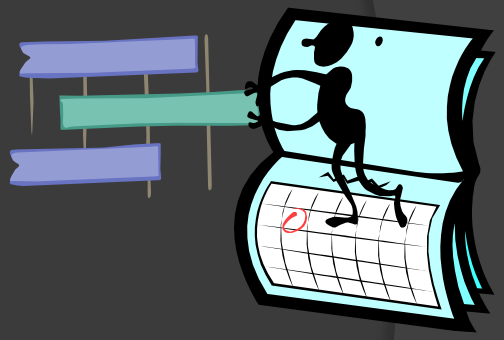
$$\mathbf{SC} < f(\text{Energy} + \text{Genius} + \textit{Cost}) - f(\textbf{\textit{Tb}} - \text{Ta})$$

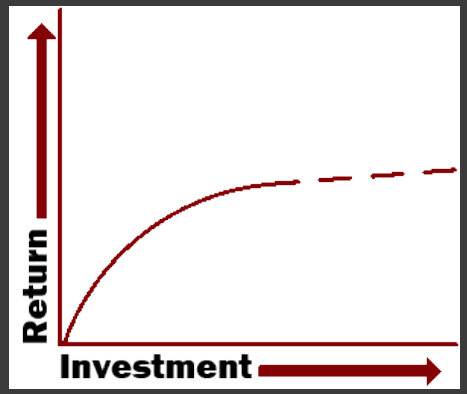


- ***Build Time*** and Cost are directly related to **SCOPE**
- ***Build Time*** is often **woefully** misjudged
- Energy, Genius and max Time allowed are (mostly) uncontrollable

Ways To Increase Project Success

- Add Energy
  - Jazz Everyone Up (free *Jolt Cola*  , 5-hour Energy )
    - Very limited effectiveness overall
  - BDD increases energy





- Add Resources 
  - Has been known to work
  - Increase resources (people) possibly reducing Time to Build
  - Diminishing point of return

- ***You Can Work Smarter*** (Add to Genius)
  - BDD improves Genius

Working Smarter

$$\textit{Quality} = \frac{\text{Processes(Time Allowed + Resources)}}{\textbf{Scope}}$$

BDD

$$\text{Build Time} = \frac{\textbf{Scope}}{\textbf{Resources}}$$

Scope is a major control variable!

Improving Quality

- Controlled research studies [Erdogmus, H. 2005]
  - Code Quality is directly related to number of tests
  - Test-First methodologies produce more tests per unit of work than any other methods
- Many consider tests as overhead [Ambler, S. 2010]
  - Skip it in favor of visual inspection
  - Delegate to Quality Assurance Team
- BDD is a rigorous Test-First methodology

BDD's Favorable Influence on Project Pressure Points

SC < *f*(Energy + Genius + *Cost*) – *f*(***Tb*** – Ta)

1. Scope
2. Quality
3. Resources
4. Time

- Scope
  - Requirements Specification
- Build Time
  - Enhancing Genius
    - Technology
    - Processes
    - Training/Skills
  - Improved Productivity
    - Increased Energy
- Quality
  - Statistically Better

Total Cost of Ownership

"Delivery" ≠ "End of Cost"

- Software has a 10-20 Year Life
- 60% > of TCO is in Maintenance
- Writer Rarely = Maintainer
  - Developers tend to roll on to new projects once initial acceptance is done

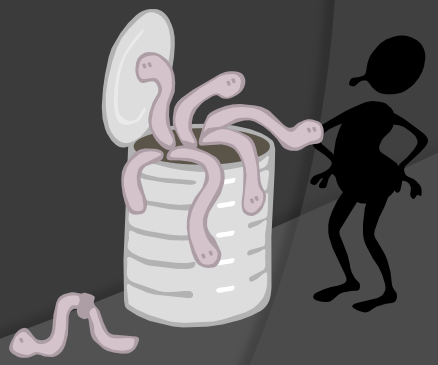
- BDD Reduces TCO
  - More Tests = Less Defects
  - Higher Quality Tests = Easier Maintenance
  - Behavioral Specifications = Better Understanding

[O'Connell, F. 2001] [Kane 2010]

#1 Way To Increase TCO
Claim Code Is Documentation

Larger Code Base = Greater Risk of Emergent Properties

- Code Will Never Tell You:
  - "*It **Has To Be** This Way*"; or
  - "*It **Happens To Look** This Way*"
- BDD's Process Provides Answers

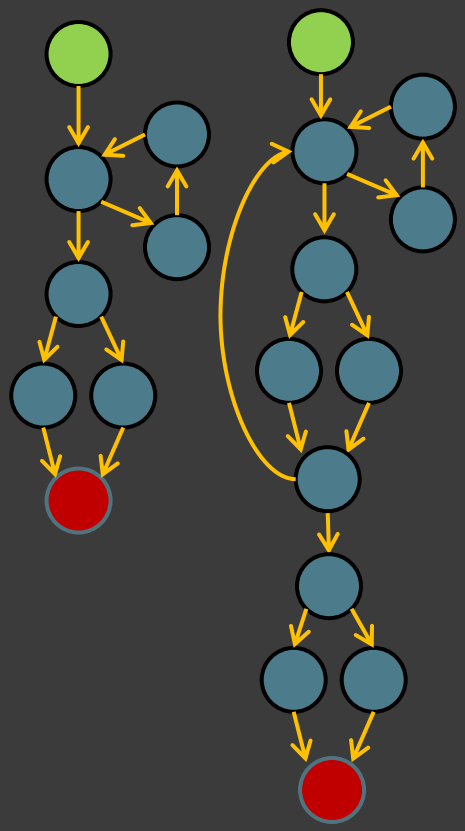


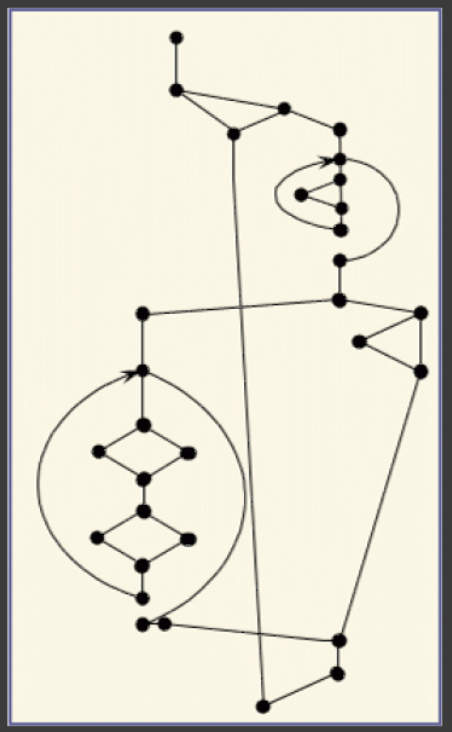
Achieving High Quality Code

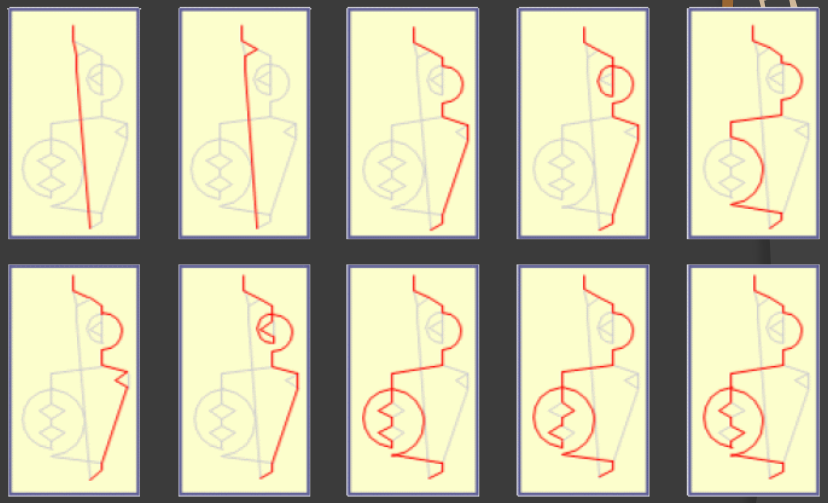
- Coding by contract
  - Require contract for every method
  - Explicit definition of secrets
- Code style requirements with teeth
  - Upper bounds on cyclomatic complexity
  - Upper bounds on fan-in / fan-out
  - Code coverage / code analysis bounds
- BDD
  - Provides contracts
  - Keeps complexity down

Cyclomatic Complexity

- Measure of the number of logical paths

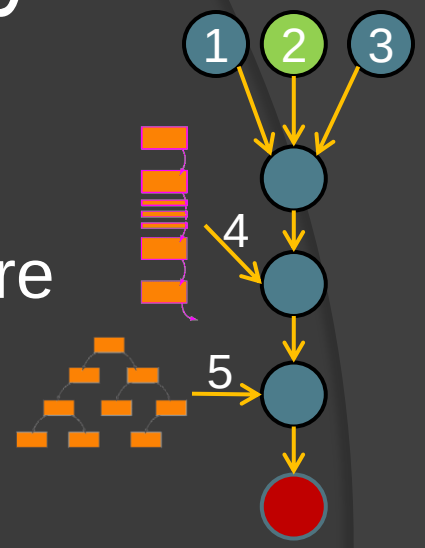




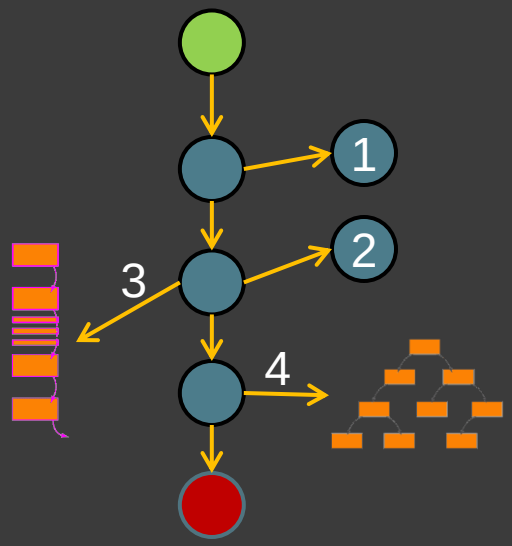


Complexity of 10 means
Need >= 10 Tests to Cover
- Code
- Logic

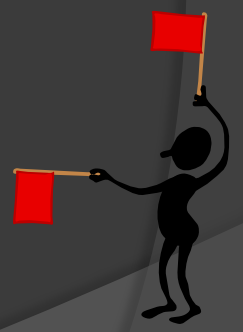
Fan-in / Fan-out Complexity



- Fan-In
  - Number of local flows into that procedure
  - number of data structures accessed

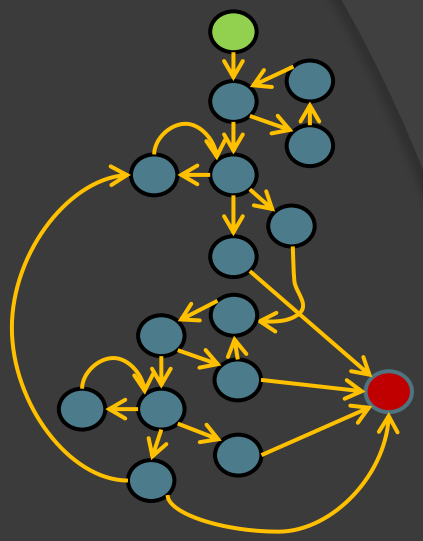


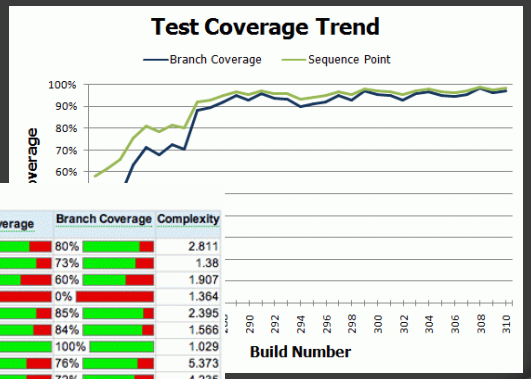
- Fan-out
  - Number of local flows out of that procedure
  - number of data structures updated



Code Analytics

- Computing and tracking
  - Cyclomatic Complexity
  - Fan-In
  - Fan-Out
  - Lines of Code (LOC)
  - Lines of Test (LOT)
  - Code Coverage



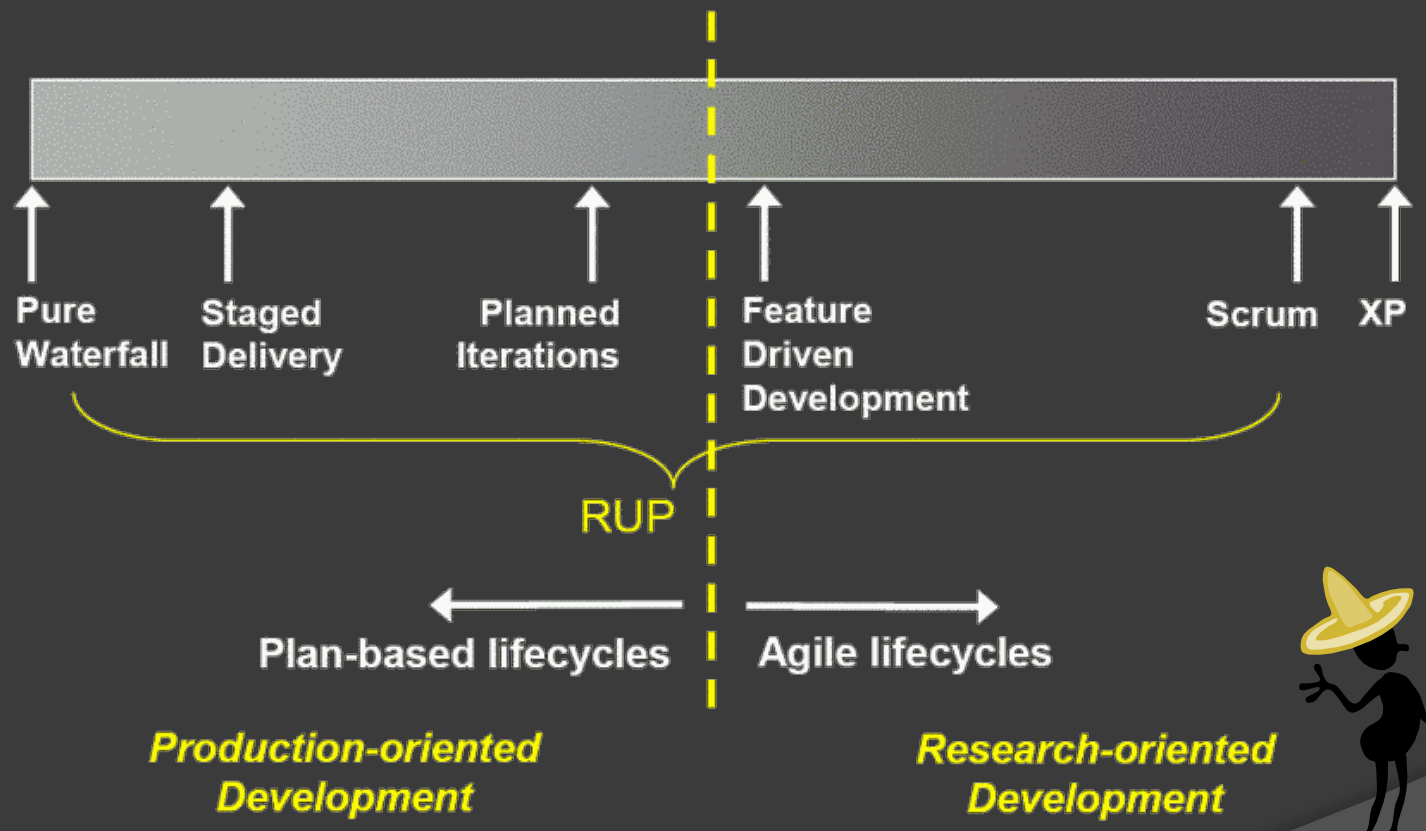


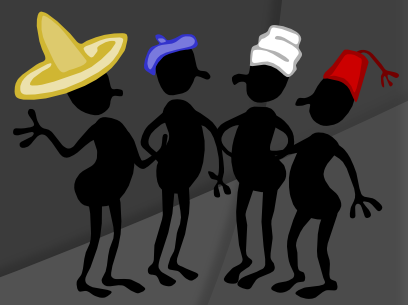
Coverage Report - All Packages

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	205	69%	80%	2.811
org.jaxen	24	77%	73%	1.38
org.jaxen.dom	3	55%	60%	1.907
org.jaxen.dom.html	2	0%	0%	1.364
org.jaxen.dom4j	2	78%	85%	2.395
org.jaxen.expr	73	73%	84%	1.566
org.jaxen.expr.iter	14	98%	100%	1.029
org.jaxen.function	27	64%	76%	5.373
org.jaxen.function.ext	6	63%	72%	4.235
org.jaxen.function.xslt	1	86%	100%	2.5
org.jaxen.javabean	4	44%	72%	1.87
org.jaxen.jdom	3	62%	63%	2.897
org.jaxen.pattern	13	49%	52%	2.135
org.jaxen.saxpath	8	51%	81%	1.887
org.jaxen.saxpath.base	6	95%	100%	10.723
org.jaxen.saxpath.helpers	2	28%	83%	1.34
org.jaxen.util	15	41%	50%	2.432
org.jaxen.xpm	2	71%	66%	1.783

Reports generated by Cobertura.

Applies to Entire Process Spectrum





"Software Development Best Practices"

[Tockey, S. 2005]

Test First Benefits for Developers

- Improved design
  - Writing test focuses mind on what needs to be done
- Improved productivity
  - Know when it is "done" and move on
- Improved quality
  - More Tests Per Unit of Work
  - Code is constantly cross checked
- Reduced TCO
  - Future changes can be done without fear of what might break

Project Management Theory

Questions?

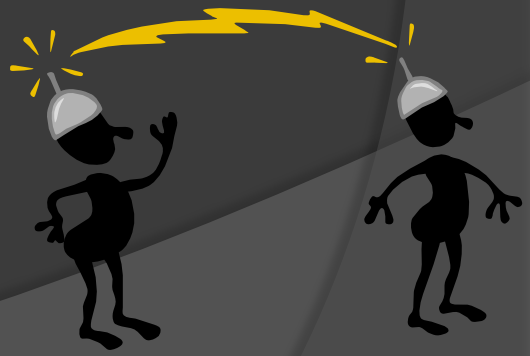
- Development challenges
- Little Project Management Theory
  - BDD basics
  - Behavior Types
  - BDD Examples
  - Summary and Future Work

Behavior Driven Design

Came from Agile & Extreme Programming movements

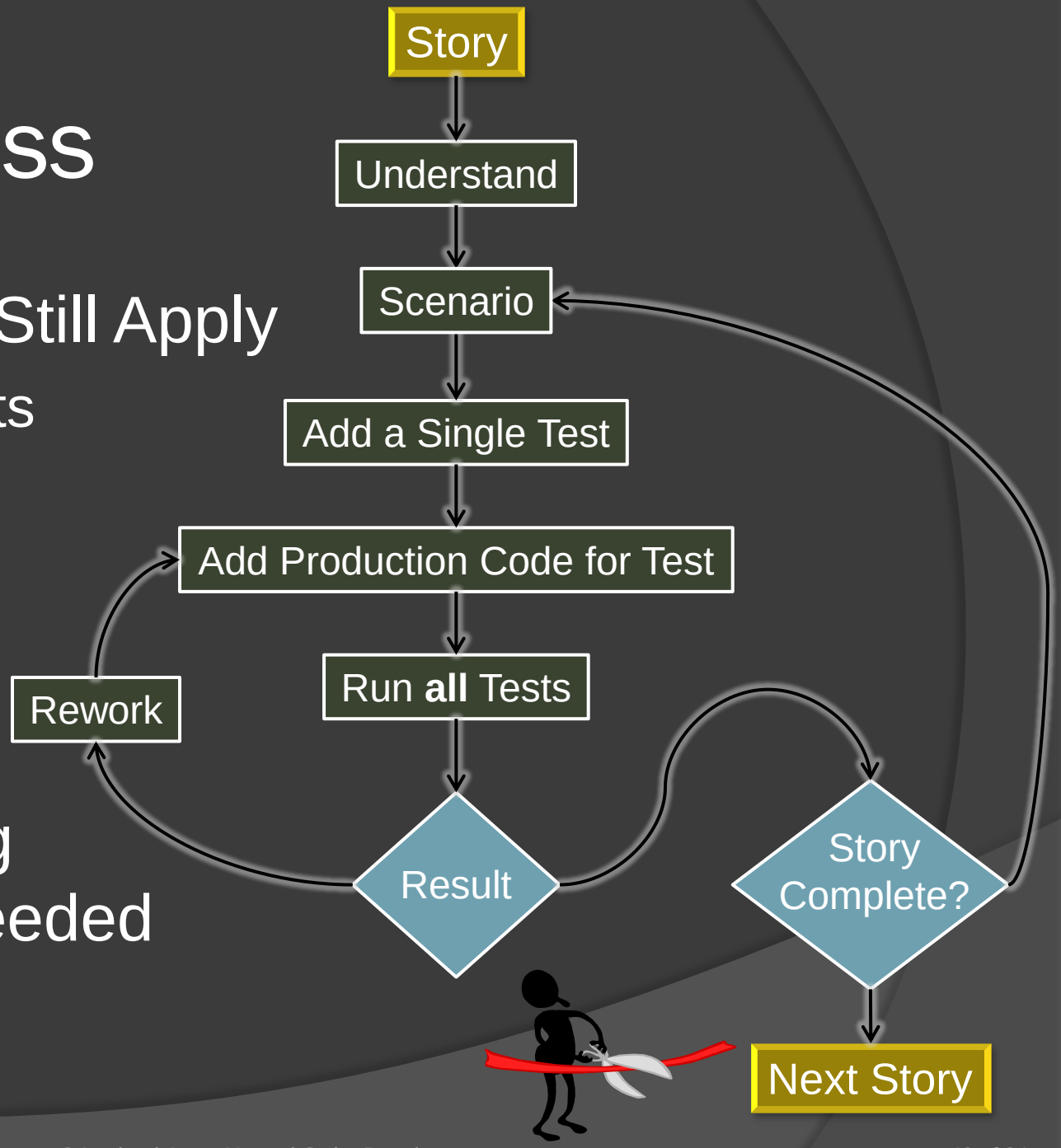
- Feedback becomes test "stories"
- Feedback occurs in short iterations
- Ruthless refactoring

- The Point is to drive the design and build using functionality and feedback
- As project progresses you end up with more and more functional tests



BDD Process

- Basic Rules Still Apply
  - Requirements
  - Design
  - Construction
  - Test
- Good Testing Skills Still Needed

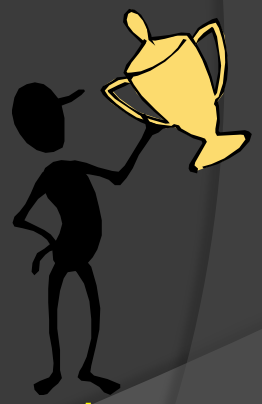


“Motion” ≠ “Progress”

- Writing Code Does Not Guarantee:
  - Required Business Behaviors Are Addressed
  - Project Goals Are Being Met
  - Code Will Be Understandable
- Testing Does Not Guarantee:
  - All Requirements Are Fulfilled
  - The Code Is Understandable
  - The Tests Fully Explain What is Being Modeled
  - The Results Are Well Designed

Success = Consistently Good

- Consistent work enhances skills
- Easy to be great
  - Perfect moments
  - Statistical accidents (rolling doubles just at the right time)
  - Cannot count on them occurring when needed
- Harder to be consistently good
  - day in and day out
  - regardless of circumstances
- Every Situation is unique

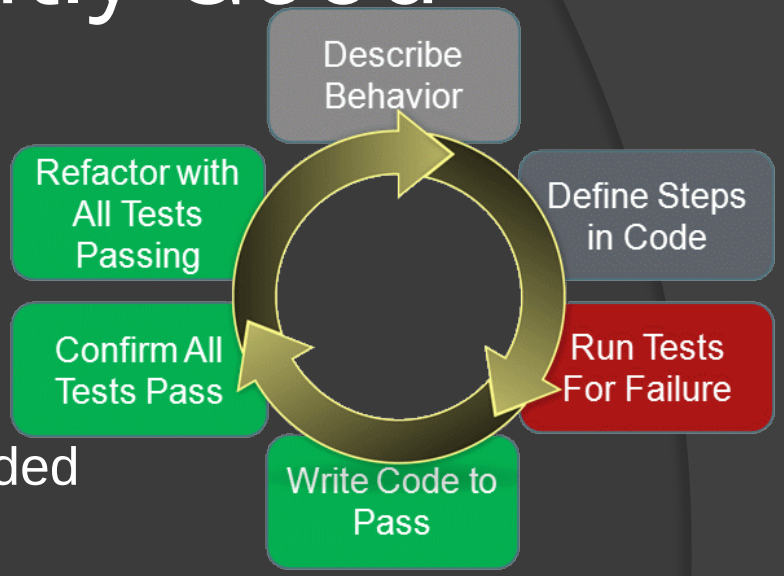


Never let on your bombing: "*this is funny, you just haven't gotten it yet*" - *Steve Martin*

[Martin, S. 2007]

How to be Consistently Good

- Always work from a plan
- Prioritize work
- Do incremental development
- Be present
- Never assume, understand what is needed
- Rely on reliable things
- Document all assumptions
- Test assumptions then test behavior
- Past performance is no guarantee
- Tangible daily results



"If I had eight hours to chop down a tree,
I'd spend six hours sharpening my ax"
- Abraham Lincoln

[Hunt, A. & Thomas, D. 2003]

Shhh It's a Secret

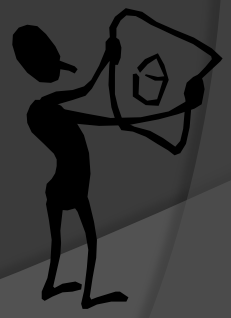
BDD = Vocab.Translate(TDD) + Min(Process);

- BDD uses Business Domain Terminology
- Translation
  - "Behavior" is more meaningful than "test"
  - "Should" is focusing and expressive
  - Use Template of "The class *should* do something"
  - Requirements = Behaviors
  - Test Case = Story
  - Test Methods = Scenarios
  - Unit Tests = Steps

BDD is

- Small set of project management techniques
- TDD wrapped in Requirements Analysis
- Ubiquitous business domain language

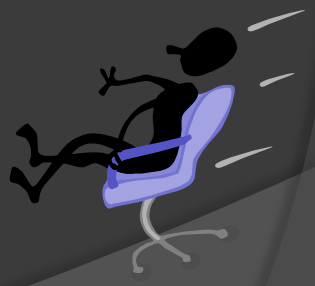
```
Story: [Title]
 As a [role]
 I would like [feature]
 So that [benefit]

Scenario: [Title]
Given [context]
When [event]
Then ensure [outcome]
```



Stories

```
Story: [Title]
 As a [role]
 I would like [feature]
 So that [benefit]
```

- Specific Narrative
- Clearly Identifies
  - Actor
  - Feature / Behavior
  - Business Value



Scenarios

```
Scenario: [Title]
 Given [context]
 AND [more context]
 ...
 When [event]
 AND [another event]
 ...
 Then ensure [outcome]
 AND ensure [another outcome]
 ...
```

- "`ensure`" identifies responsibilities of the scenario
- Titles are active
- Results use "`should`" or "`should not`"

Running BDD

- Configuration-by-convention approach

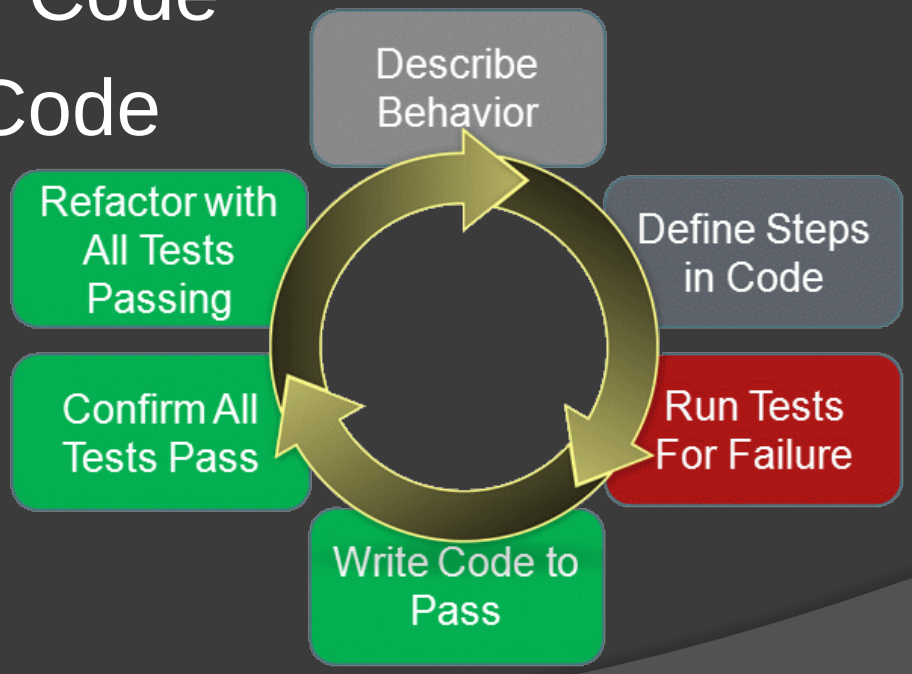
- ConfigurableEmbedder
  - Story or Stories with module specified settings
  - Direct TDD style
- AnnotatedEmbedder
  - Configuration and controls set via annotations
  - Direct TDD annotation style
- AnnotationBuilder
  - Direct manipulation of the annotation sets
  - Dependency injection

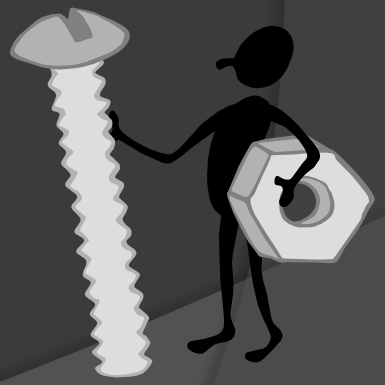
BDD Promotes Programing With Purpose

- Use defined business domain vocabulary
- Choose good names
  - Nouns for objects/modules
  - Adjectives or generic nouns for interfaces
  - Verbs or better verb phrases for methods
- Test First / Only Tests determine what to write
- Write Tests and User Docs with Assumptions
- Refactor to eliminate all code smells
- Let the compiler be your to do list
- “No Comment” (mostly)
- Do the Simplest Thing

What Is The Simplest Thing?

- All Of The Tests Run
- No Duplicate Code
- Clarity Of Code
- Minimal Code





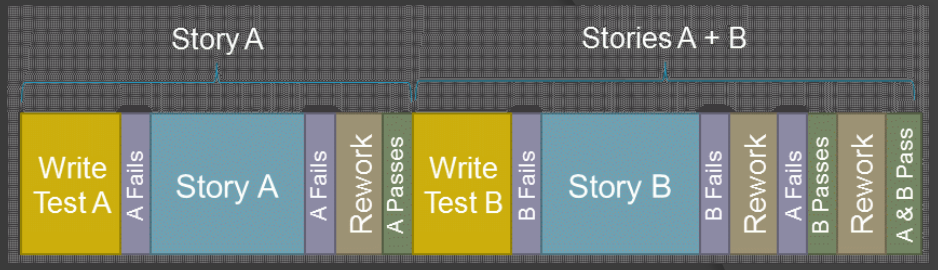
Experience = Better Results

- Controlled studies showed Test-First improvements were greater for higher skilled developers
- The larger the group the greater the leveling of results
- Consistently
  - More productive
  - More reliable quality levels

BDD Benefits for Developers

- Improved design
  - Writing test focuses mind on what needs to be done
- Improved productivity
  - Know when it is "done" and move on
  - Know what to do next
- Improved quality
  - More Tests Per Unit of Work
  - Code is constantly cross checked
- Reduced TCO
  - Future changes can be done without fear of what might break
  - Requirements, Tests, and Code provide context for updates

BDD Results

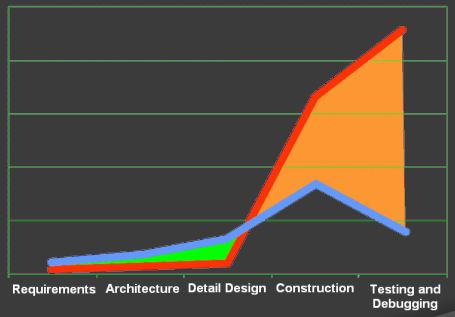


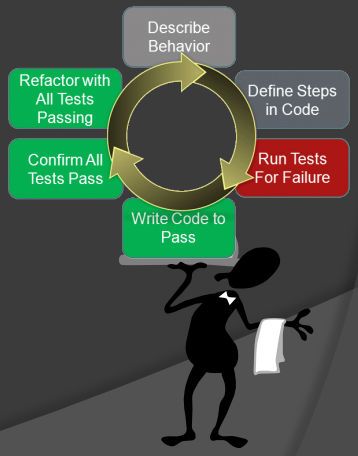
- Higher individual productivity
- Consistently good quality results
- Rapid feedback learning of business needs
- Reduced development effort
- Daily positive results

$$\text{Quality} = \frac{\text{Processes(Time Allowed + Resources)}}{\text{Scope}}$$

BDD

$$\text{Build Time} = \frac{\text{Scope}}{\text{Resources}}$$



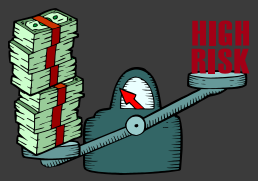


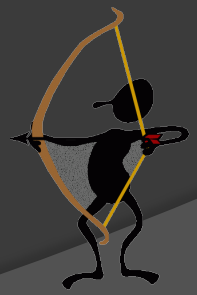
BDD Corrects TDD Issues

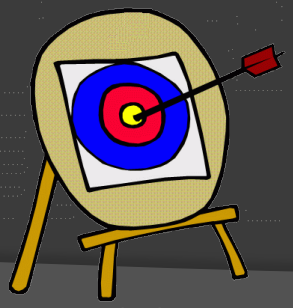
- Where to start?
  - Business Requirement Stories
  - Highest Priority Behavior
- What to test?
  - Everything that can be expressed as Should or Should Not
- What not to test? (Should it?)
  - Anything that doesn't fit into the context
- How much to test?
  - All the behaviors needed to meet the Story Requirement
- What to call the tests?
  - ShouldDoX
  - ShouldFailIfY

Secrets of Success

- Preparedness
- Anticipating the risks
- Knowing how to react
- Knowing where and when to act
- Doing what is needed with surgical precision

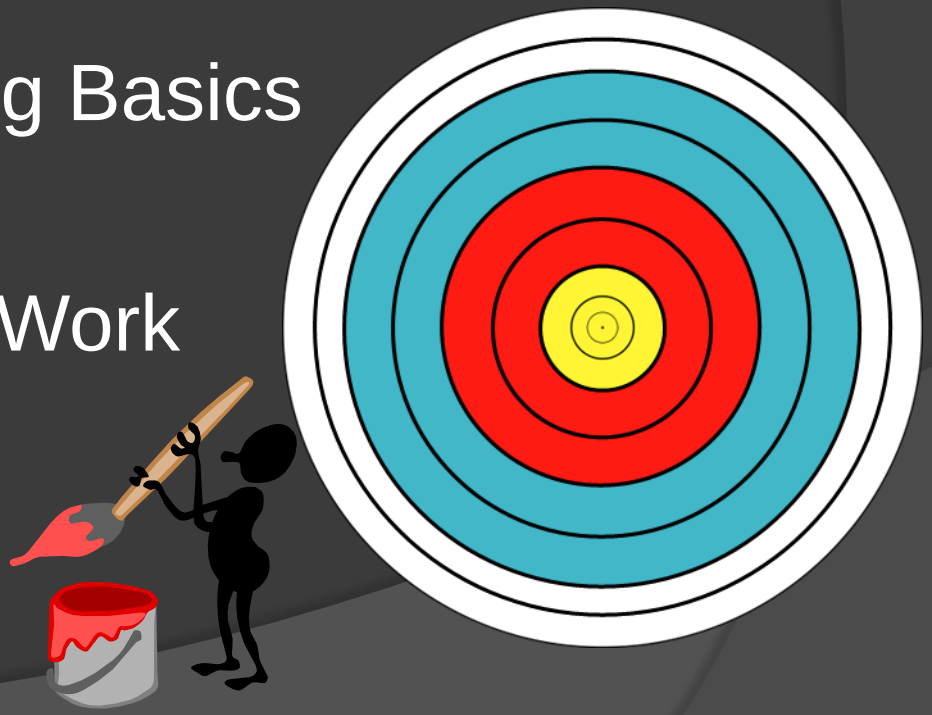






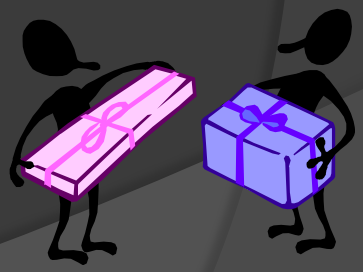
BDD Basics Questions?

- Development challenges
- Little Project Management Theory
- BDD basics
- ~~Behavior Types~~ Testing Basics
- BDD Examples
- Summary and Future Work



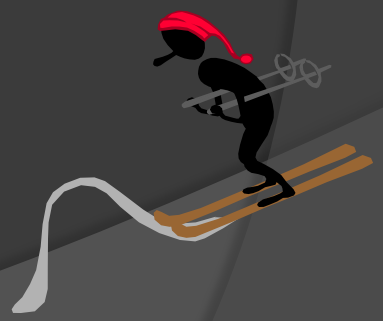
Back to Basics – Test Case

- Describes inputs / trigger events
- Describes expected results
- Describes prerequisites and environment
- Authoritative source for defect detection



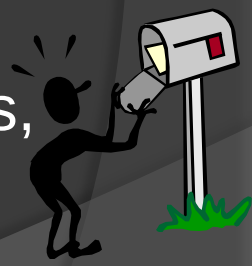
Back to Basics – Defect

- Any behavior that reduces value
- Behavioral Defect:
  - Application does not do what is *reasonably* expected by an end user
- Specification Defect:
  - An inconsistency between application *behavior and specification*'s description of expected behavior



Back to Basics – Classifications

- Expectation Based
  - Positive Case : Verify expected behavior
    - Provided by the requirements
  - Negative Case : Challenge assumptions (inputs, unanticipated states, etc.)
    - Usually not provided by requirements
- Perspective Based
  - Functional - (black box) analysis of input/output
  - Structural – (Glass box) analysis of flow, algorithms, design, etc.
- Phase
  - Acceptance, Beta, System, Regression, Integration, Unit

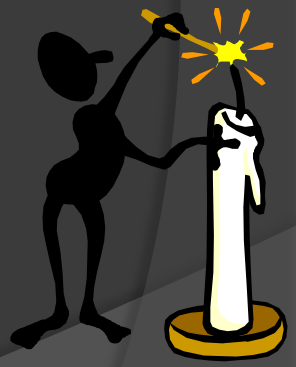


Back to Basics – Types of Testing

- Acceptance: minimum behaviors provided/tested by client
- System: [automated] process to diff vs. original requirements
- Integration: inter-component interaction & communication
- Unit/Programmer: small method testing
- Regression: System test suites confirm legacy behavior and new features
- Beta: Exposing end users to identify defects

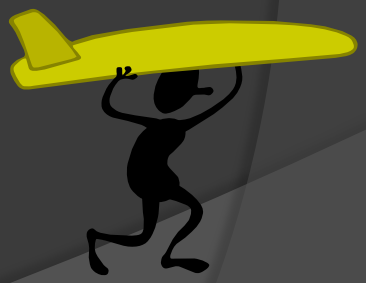
Back to Basics – A Good Test

- Repeatable / Idempotent
- Atomic
- Simple to Perform (fast is good too)
- Fails for one reason only
- Is Unique for a specific behavior
- Improves test coverage (not redundant)



Back to Basics – Fixture

- Test Case including:
  - All preconditions
  - All assumptions
  - The Runtime context including setup
- Fixture is an instantiated running Test
- In BDD a Fixture is a Scenario



BDD Basics Questions?

- Development challenges
- Little Project Management Theory
- BDD basics
- ~~Behavior types~~ Testing Basics
- BDD Examples
- Summary and Future Work

Where Stuff Goes

Using Maven For Example (not required)

```
/pom.xml (the maven project descriptor)
 /src
  /main
   /java (app classes)
   /resources (properties-files)
  /test
   /java (test code)
   /resources (testing properties-files)
```

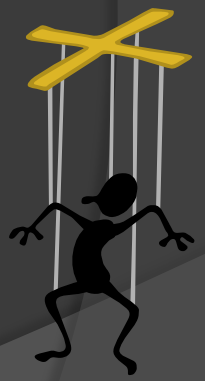
Minimum Files Setup

1. pom.xml
2. src/test/java/classcost/PresenterIsGivenClassCostStory.java
3. src/test/java/classcost/ClassCostSteps.java
4. src/test/resources/classcost/presenter_is_given_class_cost_story.story

Example Story

presenter_is_given_class_cost_story.story

```
Story: calculate the effective business cost of a
meeting
As a Presenter
I can compute the meeting business cost
So that the ROI for a meeting may be known

Scenario: presenter enters class count of 10
Given an average hourly rate of 50.00
When attendance count is set to 10
Then ensure class cost should be 500.00
```



The Class 1 of 2

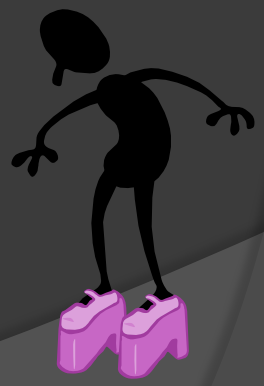
PresenterIsGivenClassCostStory.java

```
public class PresenterIsGivenClassCostStory extends JUnitStory {

    // the configuration, starting from default MostUsefulConfiguration, and changing only what is needed
    @Override
    public Configuration configuration() {

        return new MostUsefulConfiguration()
            // where to find the stories
            .useStoryLoader(new LoadFromClasspath(this.getClass().getClassLoader()))
            // CONSOLE and TXT reporting
            .useStoryReporterBuilder(new
StoryReporterBuilder().withDefaultFormats().withFormats(Format.CONSOLE, Format.TXT));
    }

    // Here we specify the steps classes
    @Override
    public List<CandidateSteps> candidateSteps() {
        return new InstanceStepsFactory(configuration(), new ClassCostSteps()).createCandidateSteps();
    }
}
```

Class 2 of 2

`ClassCostSteps.java`

```
public class ClassCostSteps {

}
```

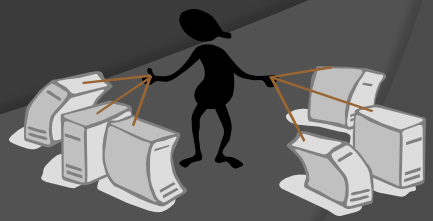


The POM part 1 of 2 <dependencies/>

```
<properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 </properties>

 <dependencies>
   <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>3.8.1</version>
     <scope>test</scope>
   </dependency>
   <dependency>
       <groupId>org.jbehave</groupId>
       <artifactId>jbehave-maven-plugin</artifactId>
       <version>3.1.1</version>
   </dependency>
 </dependencies>
```

The POM part 2 of 2 <build/>

```
<plugin>
        <groupId>org.jbehave</groupId>
        <artifactId>jbehave-maven-plugin</artifactId>
        <version>3.1.1</version>
<executions> <execution>
            <id>run-stories-as-embeddables</id>
            <phase>test</phase>
            <configuration>
              <scope>test</scope>
              <includes> <include>**/*Story.java</include> </includes>
              <systemProperties>
                <property> <name>java.awt.headless</name> <value>true</value> </property>
              </systemProperties>
              <ignoreFailureInStories>true</ignoreFailureInStories>
              <ignoreFailureInView>false</ignoreFailureInView>
            </configuration>
            <goals> <goal>run-stories-as-embeddables</goal> </goals>
</execution> </executions> </plugin>
```



The Results

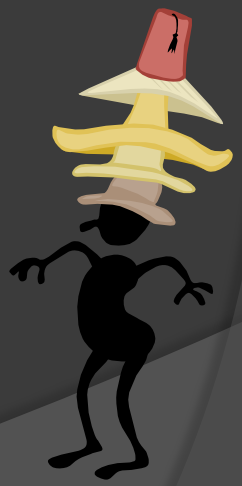
[INFO] Running story com/nodsw/bddcourse/classcost/presenter_is_given_class_cost_story.story
Story: calculate the effective business cost of a meeting
As a Presenter
I can compute the meeting business cost
So that the ROI for a meeting may be known
(com/nodsw/bddcourse/classcost/presenter_is_given_class_cost_story.story)
Scenario: presenter enters class count of 10
Given an average hourly rate of 50.00 (PENDING)
When class attendance count is set to 10 (PENDING)
Then ensure the class cost should be 500.00 (PENDING)

[INFO] Reports view generated with 1 stories containing 1 scenarios (of which 0 failed)

BDD frameworks built on xUnit

- Jbehave – Java
- Cucumber – Ruby
- Lettuce – Python

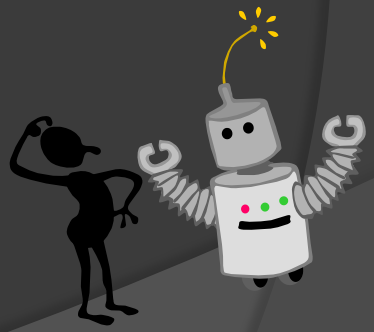
- Lots more to play with



Class 2 of 2 Ready

ClassCostSteps.java

```
import org.jbehave.core.annotations.Given;
import org.jbehave.core.annotations.Then;
import org.jbehave.core.annotations.When;

public class ClassCostSteps {

    @Given("an average hourly rate of $hourlyRate")
    public void setHourlyRate ( double hourlyRate ) {
    }


    @When("class attendance count is set to $classSize")
    public void theClassSizeIs ( int classSize ) {
    }

    @Then("ensure the class cost should be $classCost")
    public void theClassCostShouldBe ( double classCost ) {
    }
```



Now It Looks Like Working

[INFO] Running story com/nodsw/bddcourse/classcost/presenter_is_given_class_cost_story.story

Story: calculate the effective business cost of a meeting

As a Presenter

I can compute the meeting business cost

So that the ROI for a meeting may be known

(com/nodsw/bddcourse/classcost/presenter_is_given_class_cost_story.story)

Scenario: presenter enters class count of 10

Given an average hourly rate of 50.00

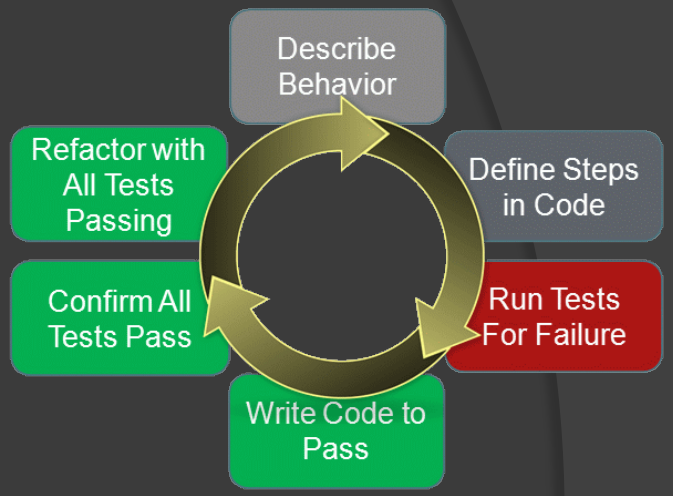
When class attendance count is set to 10

Then ensure the class cost should be 500.00

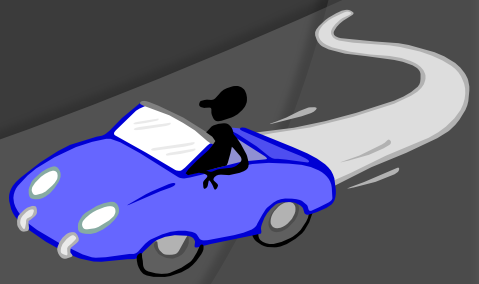
[INFO] Reports view generated with 1 stories containing 1 scenarios (of which 0 failed)

Framework Done, Now Roll On

```
Story: calculate the effective business cost of a meeting
As a Presenter
I can compute the meeting business cost
So that the ROI for a meeting may be known

Scenario: presenter enters meeting details directly
Given an empty meeting
When attendance count is set to 10
And average hourly rate is set to 50.00
And meeting length is set to 1
Then ensure meeting cost should be 500.00

Scenario: presenter does not provide hourly rate
Given an empty meeting
When attendance count is set to 10
And meeting length is set to 1
Then ensure meeting cost should not be available

Scenario: presenter enters series of hourly rates for average
Given an empty meeting
And no meeting count
When presenter adds attendee with hourly rate of:
|hourlyRate|
| 45.0 |
| 30.0 |
| 10.0 |
And meeting length is set to 1
Then ensure meeting cost should be 85.00
And meeting count should be 3
```



- Enter Scenarios as you think of them
- The Compiler will tell what else is needed

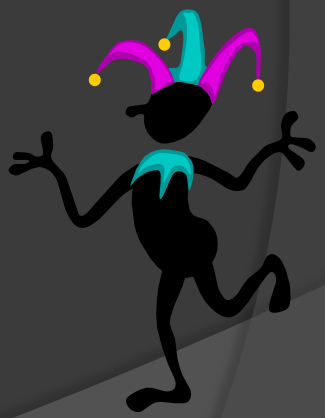


BDD Questions?

- Development challenges
- Little Project Management Theory
- BDD basics
- ~~Behavior Types~~ Testing Basics
- BDD Examples
- Summary and Future Work

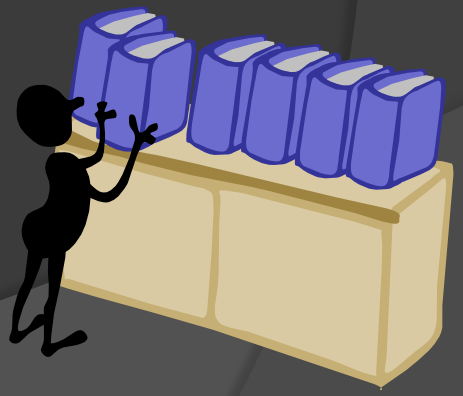
Summary

- Project Success Depends on
  - Understanding the requirements
  - Properly scoping work (on going)
  - Robust Code Stewardship
- BDD (Test First Programmers)
  - Write More Tests per Unit of Work
  - Improved productivity
  - Consistent Good Quality of code



Future Work

- Mocks
- Design Patterns
  - Inversion of Control (IOC) / Dependency Injection
- Requirements Analysis
- Agile Modeling
- Diagramming
- Agiledocs and Jbehave



Take Homes

- Iterative projects have better success chance
- Write user guides and behaviors with assumptions
- Strive for consistently good
- Study patterns
  - when & how to use
  - when not to use
- Use BDD style regardless of pressures
  - it makes you iterative

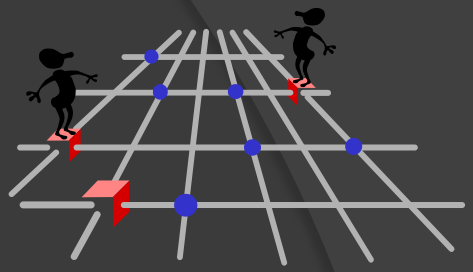
Where To Start (Roughly in Order)

- Dan North's "***Introducing BDD***" online article (the actual beginning of BDD) http://blog.dannorth.net/introducing-bdd/
- ***A Beginners Guide to Dependency Injection*** by Dhananjay Nene (online article) http://www.theserverside.com/news/1321158/A-beginners-guide-to-Dependency-Injection
- ***Spring 3 Tutorial: Setting Up & Configuring The Environment*** by Jason Tee (online article) http://www.theserverside.com/tutorial/Spring-30-Tutorial-Setting-Up-Configuring-The-Environment
- ***Inversion of Control Containers and the Dependency Injection pattern*** by Martin Fowler (online article) http://martinfowler.com/articles/injection.html
- C2 "Extreme Programming Roadmap" http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap
- ***Test-Driven Development: A Practical Guide*** (book) by David Astels http://www.amazon.com/Test-Driven-Development-Practical-David-Astels/dp/0131016490
- ***Test Driven Development By Example*** by Kent Beck (book) http://www.amazon.com/Test-Driven-Development-Kent-Beck/dp/0321146530
- ***Pragmatic Unit Testing in Java with Junit*** (book) http://oreilly.com/catalog/9780974514017/
- ***The Pragmatic Programmer*** by Andy Hunt and Dave Thomas (book) http://www.amazon.com/Pragmatic-Programmer-Journeyman-Master/dp/020161622X
- ***Data Structures and Algorithm Analysis in Java (2nd Edition)*** by Mark A. Weiss (book) http://www.amazon.com/Data-Structures-Algorithm-Analysis-Java/dp/0321370139
- ***Refactoring: Improving the Design of Existing Code*** by Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts (book) http://www.amazon.com/Refactoring-Improving-Design-Existing-Code/dp/0201485672
- ***Implementation Patterns*** by Kent Beck (book) http://www.amazon.com/Implementation-Patterns-Kent-Beck/dp/0321413091
- ***Software Project Survival Guide*** by Steve McConnell (book) http://www.amazon.com/Software-Project-Survival-Guide-Practices/dp/1572316217
- ***Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*** by Craig Larman (book) http://www.amazon.com/Applying-UML-Patterns-Introduction-Object-Oriented/dp/0131489062

Highly Recommend

- Construx Software Seminars
  - Developer Testing Boot Camp
  - Requirements Boot Camp
  - Software Estimation in Depth
  - Object-Oriented Requirements Analysis and Design Using the UML
- http://construx.com/

Contributors



- Jon Wilmoth II, Development Senior Expert, QP PGM Program Management, Amdocs Inc. Seattle, Washington (jon.wilmoth@amdocs.com)
- Jason Pringle, Senior Software Architect, Product Engineering, Amdocs Inc., Seattle, Washington (jpringle@amdocs.com)
- Joseph Wright, Infra Senior Subject Matter Expert, Application Systems, Amdocs Inc., Seattle, Washington (robert.wright@amdocs.com)
- Maurizio Calabrese, Infra Group Leader, Qpass Infra Management, Amdocs Inc., Seattle, Washington (maurizio.calabrese@amdocs.com)
- Construx Software, 10900 NE 8th Street, Suite 1350, Bellevue, WA 98004. http://construx.com/

References

1. Ambler, S. (Aug 2, 2010). IT Project Success Rates. Dr. Jobbs Journel.
2. Ambler, S. (2004). The Object Primer 3rd Edition, Agile Model Driven Development with UML 2. Cambridge University Press.
3. Erdogmus, H. (Jan 2005). On the Effectiveness of Test-First Approach to Programming. IEEE Transactions on Software Engineering, 31(1) January 2005.
4. Eveleens, J. and Verhoef, C. (2010). The Rise and Fall of the Chaos Report Figures. IEEE Software, January/February 2010. Retrieved from ProQuest: ProQuest Computing database.
5. Hunt, A. and Thomas, D. (Apr 18, 2003). How to Be a Better Coder. Addison Wesley.
6. Kane (February 25, 2010). The benefits of TDD are neither clear nor are they immediately apparent. Retrieved October 28, 2010 from http://www.scrumology.com/2010/agility/the-benefits-of-tdd-are-neither-clear-nor-are-they-immediately-apparent/
7. Martin, S. (Feb 2008). Being Funny, How the pathbreaking comedian got his act together. Smithsonian Magazine, Retrieved Oct 2009 from http://www.smithsonianmag.com/arts-culture/funny-martin-200802.html .
8. McConnell, S. (1998). Software Project Survival Guide. Microsoft Press.
9. O'Connell, F. (May 2001). How to Run Successful Projects III, The Silver Bullet. Addison-Wesley.
10. Rebecca J, W. (Feb 1, 2007). Driven to ... Discovering Your Design Values. IEEE Software, 24(1), 9. Retrieved from ProQuest: ProQuest Computing database.
11. Tockey, S. (2006). Object Oriented Requirements Analysis and Design Using UML, week training course. Construx http://construx.com/

Thank You!

Always Available for a cup of coffee

For more information, contact:

Leeland Artra
Leeland . Artra at Amdocs . Com
Leeland at NODSW . Com

Amdocs
2211 Elliott Avenue, Suite 400
Seattle, Washington 98121
United States
+1.206.447.6000 Phone
+1.206.447.0669 Fax

Technical Site: http://nodsw.com

Figure Art Licensed from Screen Beans http://screenbeans.com