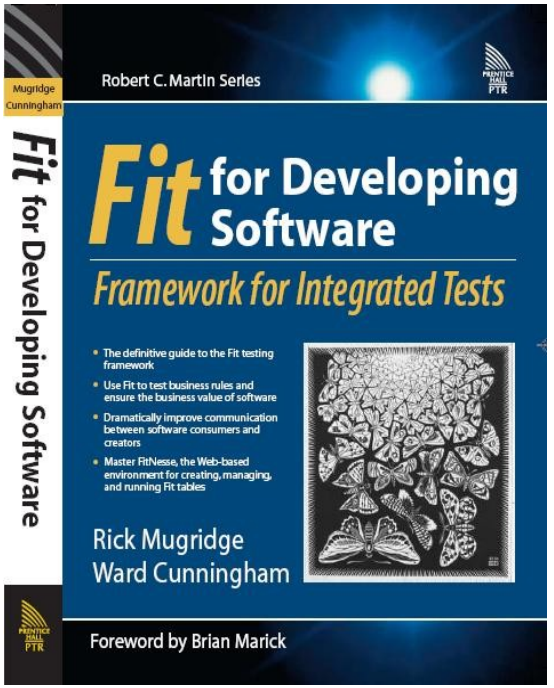
Using Fit:
An open-source testing framework for developers, testers, and product managers

Rick Mugridge

RimuResearch.com

rick@rimuresearch.com



Fit

- **Fit: Ward Cunningham in 2002, in Java**
  - **Versions for C#, C++, Python, Ruby, Smalltalk, ...**
- **Tables: storytest and report**
- **Fixtures mediate**

- **Tables (and supporting text, etc) from:**
  - **HTML, FitNesse, Excel, MS Word, etc**

- **FitNesse (Micah and Bob Martin)**
- **FitLibrary (Rick Mugridge)**

Fit

TestDisconnect - Moz...

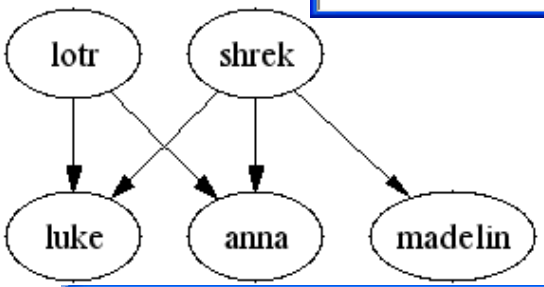
ChatStart				
connect user	anna			
user	anna	*creates*	lotr	*room*
user	anna	*enters*	lotr	*room*
users in room	lotr			
name				
anna				
disconnect user	anna			
check	*occupant count*	lotr	0	

TestArrayWrong - Mozilla

DiscountGroupArrayList			
futureValue	*maxOwing*	*minPurchase*	*discountPercent*
low	0.00	0.00	0
low	0.00	2000.00	3
medium	500.00	600.00	3
high *missing*	2000.00	2000.00	10
medium	0.00	500.00	5
high *surplus*	2000.0	2000.0	10.0

UsersAndRooms

check users



InvoiceTable - Mozilla

TaxInvoice					
Uni of Auckland	Delivered to:	Number:	216017-01		
Private Bag 92019	Attn: Sal Mayha	Date:	05/01/04		
Auckland	Uni of Auckland	Order:	TC000015473-REPL		
		Cust:	UNI34		
#	*Part Number*	*Description*	*Dispatch*	*Price*	*Total*
1	CAT 98142-00-GH	L3 Switch 32x1000T	1	6804.00	6804.00
2	CAT 99000-01-PH	Macronetic switch	2	2317.00	2317.00
				Total:	9121.00
Special Delivery:	AS0555P				

Four Development Issues

(1) How can we improve test coverage on a legacy system without discouraging its evolution?

(2) Now that we have a successful product with a great UI, how do we provide program access, such as through SOAP?

(3) How can QA be involved earlier and more effectively?

(4) How can product managers better utilize their expertise?

- **Fit storytests can help...**
  - **Concrete examples as tests**
  - **Storytest Driven Development (SDD)**
  - **Executable specifications**

1. Supporting Legacy System

- **Test coverage needs to be improved**
  - **But also allow for evolution**
- **Catch-22:**
  - **To test-infect and to evolve, need to reframe and refactor**
  - **To refactor well, need automated tests**
- **Michael Feathers, *Working Effectively with Legacy Code***
  - **Test-infecting with unit tests**
  - **On demand, bringing in TDD**
- **Support such change with Fit storytests**
  - **Test through UI initially**
  - **Support evolution**

Supporting Legacy

- **Standard approaches:**
  - **Manual test**
  - **Record and playback at UI level**
    - **"The UI is the system" and other *attractors***
  - **Automated test at UI level**
- **But:**
  - **Verbose tests, often in weird scripting languages**
  - **Repetitious as little scope for abstraction**
  - **Hard to alter as system changes**
  - **Hard to see beyond the detail**
  - **Don't support clear thinking about the domain**
  - **(Analogy: XIV, weak programming language)**

2. Providing Program Access

- **Usual problem:**
  - **Tangled UI, domain and database**
- **Reasons for separate layers:**
  - **Modularity**
  - **TDD**
  - **Replace or duplicate UI**
  - **Clarify domain**
  - **Program access**

Providing Program Access

- Ways to provide program access:

- Duplicate code that's in UI
- Pull out service layer, but very procedural
- Pull out domain layer
  - Design program access into that
  - OO
- Same approach as with legacy:
  - Use same Fit storytests to:
    - Clarify domain
    - Support change by testing through UI
    - Also test directly into domain layer as it emerges

3. Quality Assurance

- How to get QA involved earlier and more effectively?
- Usually QA test later, so that:
  - Feedback can be slow, especially: system-level, performance
  - Rework time increases with feedback delays
    - Compare 1 year, 1 week, 1 hour, 1 second
  - Testing perspective on requirements too late
  - Testing often squeezed

Quality Assurance

- **Good evidence for:**
  - **Try to go fast → Poor quality → go slow**
  - **High quality → can go fast**
- **QA:**
  - **Upfront with storytests**
  - **Continuous: exploratory testing**

- **Upfront tests:**
  - **Avoid premature and unnecessary commitment**

- **Switch emphasis to Quality Injection**

4. Product Managers

- **How can product managers better utilize their expertise?**

- **Requirements, ROI**
- **Hard to express requirements in a document**

- **Ongoing communication is key**
  - **Being precise**
- **Agile estimating and planning, iterations**

- **What if PMs help with upfront Fit storytests?**
  - **Product emphasis**
  - **Domain expertise: *ubiquitous language* of domain**
  - **Provide concrete examples**

Traditional Requirements

- **Gathered at start of project and commitments made**
- **But:**
  - **Last chance to include what *may be* needed**
  - **Analysis paralysis**
  - **Endless detail**
  - **Uncertainty about problem and value of solution**
    - **May require serious refinement of thinking**
  - **Changes**
  - **General statements, hard to write and to read**
  - **Hard to provide insight into the domain**
  - **Quickly out of date – irrelevant legacy**
  - **Rarely good enough to support validation**

Towards Solutions

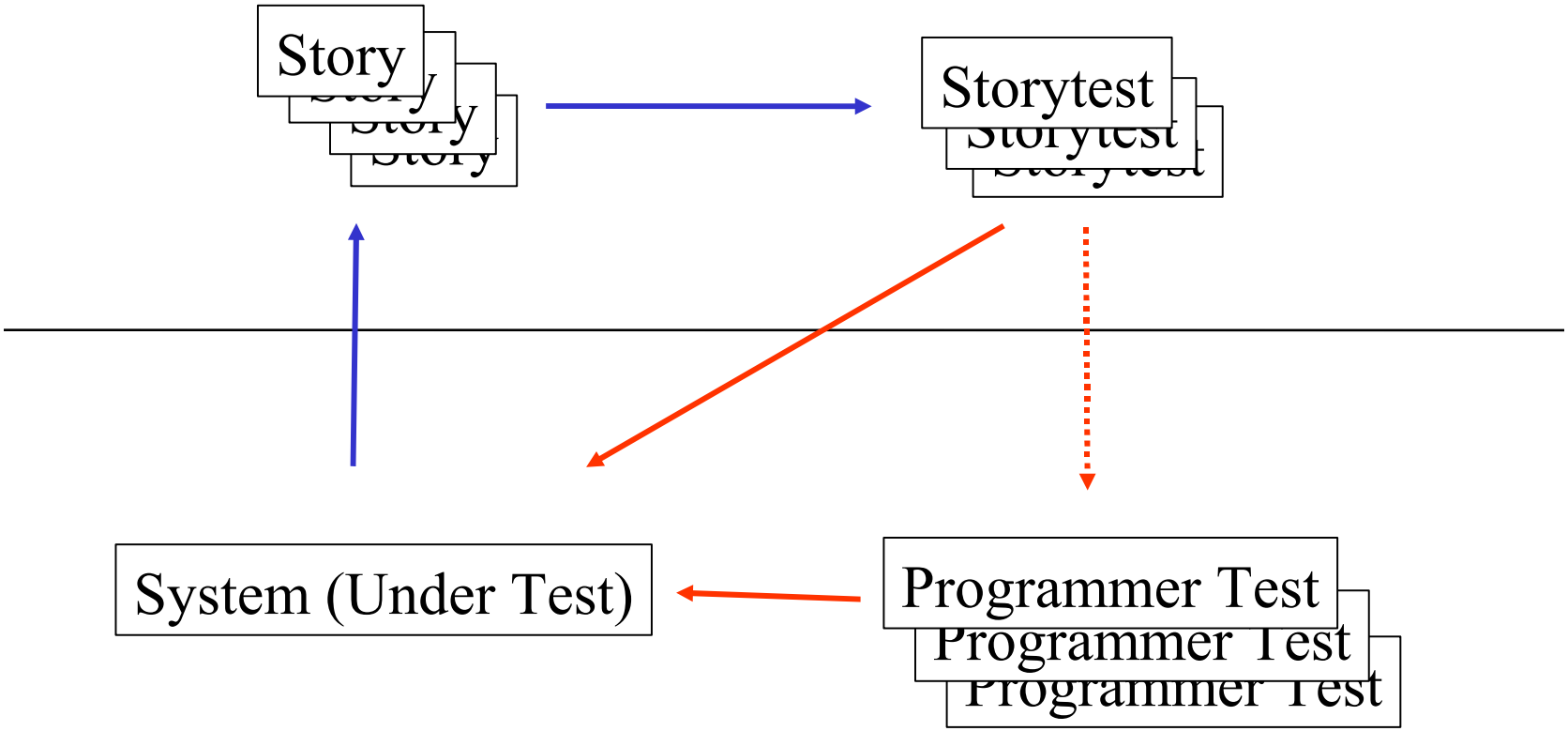
- **Test Driven Development: TDD**
- **Storytest Driven Development: SDD**
- **Fit storytests**
- **Concrete examples and Domain Driven Design**

Test Driven Development

- Programmers drive development with unit tests
  - Design, development, test all the time
  - Unit tests:
    - Used to make design decisions
    - Support refactoring to refine design
      - They support clear thinking!
    - Encourage strong modularity and agile code
    - Let you know when you're done
    - Regression testing

- Unit tests help to lower implementation risks
- But what about business risks?

Storytest Driven Development



Storytest Driven Development: Joshua Kerievsky, IndustrialXP

Storytests & Product Managers

- **Storytests owned by Product Managers**
- **Express business rules as examples**
    - **precise, concise, concrete and realistic**
- **Encourage understanding, discussion and communication**
- **Evolve as understanding of business needs change**

- **Significant and valuable business resource**

Storytests & Developers

- **Storytests:**
  - **Clearly communicating**
  - **Connected easily to the System Under Test for testing**
  - **Run independently of each other**
  - **Organised in test suites to run automatically in a build**
  - **Run quickly with good feedback**

- **Storytests can drive the development**
  - **Storytests are often added in response to developers' questions during implementation**

Business Rules & Storytests

- Eg, a conference management system:
  - Business objects
    - Eg, Attendee, session, track, proposal, reviewer
  - Business processes
    - Eg, Select a session to attend, submit a proposal, review a proposal
  - Business calculations
    - Eg, Calculate cost of attendance, combine reviewer ratings for proposal
  - Business constraints
    - Eg, Can't attend two sessions at same time, can't submit a change to a proposal after the deadline, can't review own proposal, can review a proposal only once, can't view reviews for own proposal

Domain Driven Design

- ***Domain Driven Design*, Eric Evans, Addison Wesley, 2004**
- **Develop a *ubiquitous language* for domain**
  - **Used in communication between product managers & developers**
  - **Used in the code**
- **The language evolves as domain undergoes clarification**
  - **As business opportunities better understood**
  - **As business rules emerge and are reframed**

Storytests Encode Domain

- **Combine specifications and automated testing**
  - **Executable Specification**

- **The same storytests can be used to:**
  - **Test into the domain layer**
  - **Test at protocol level, etc in subsystems**
  - **Test through the UI (Swing, Selenium)**
  - **Support performance and load testing**
  - **Auto-generate tailored user help**

Domain Driven Design

- **The whole team contributes:**
  - **Business knowledge**
  - **A testing perspective**
  - **An OO modelling perspective**
  - **Abstraction skills: refactoring, untangling**
  - **Asking the right questions**

Remaining Challenges

- Generations
  - Fit, FitLibrary, FitLibrary2
- Design patterns and table smells
  - DDD patterns
- Organizing large set of storytests
  - Domain
- Refactoring, etc tool support
  - Refactor domain
  - Extract calculation and constraint rules
  - Version control
  - Emergent: Use cases and UML class diagrams
- Different skills

- Domain Specific Language

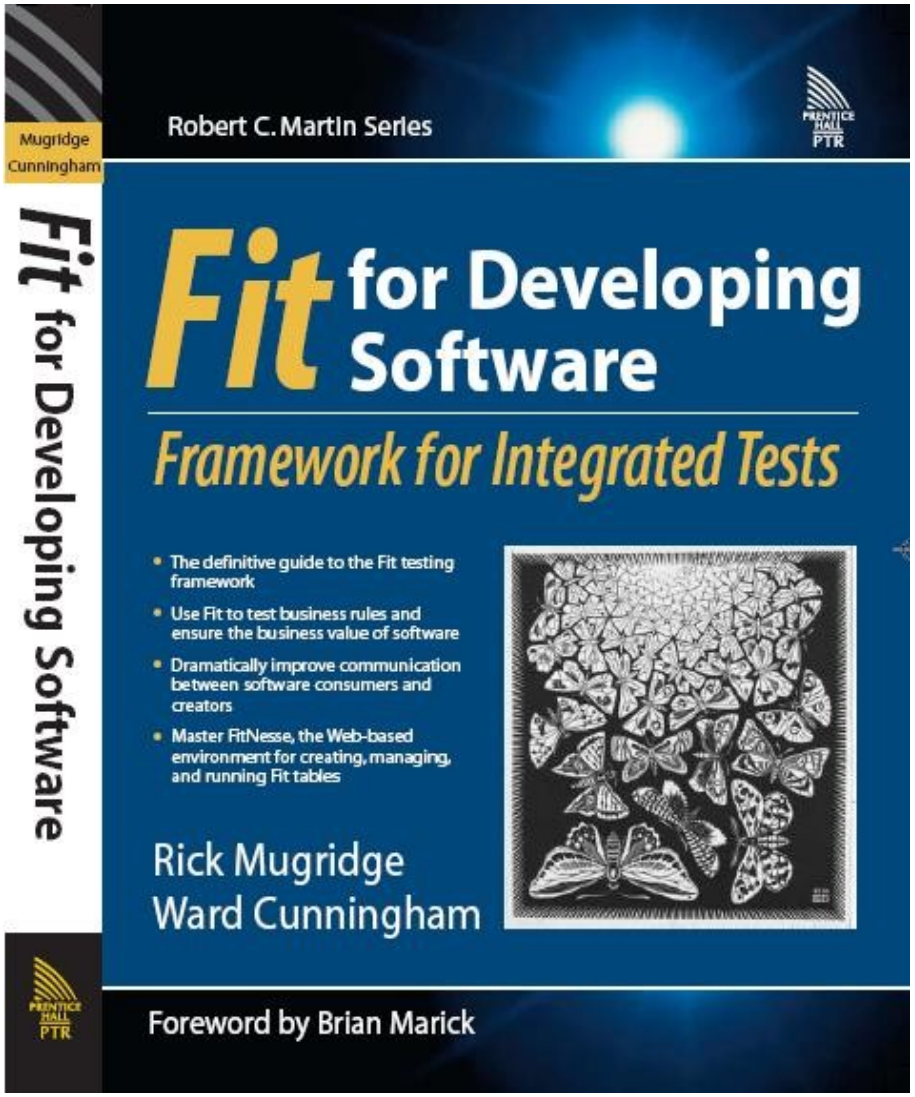
Conclusions

- **Four Issues: Legacy, Program Access, QA and PM**
  - **Testing → Requirements**

- **Develop domain with storytests**
  - **Concrete examples**
  - **Executable specifications**
  - **Clarify thinking and abstractions**
  - **Communicating**
  - **Align with classes in domain layer**
  - **Testing completion & regression**
    - **In different ways**

Fit

- "Rick and Ward continue to amaze me. Testing business rules is a fundamentally hard thing that has confounded many, and yet these two have devised a mechanism that cuts to the essence of the problem."
  – Grady Booch



Any Questions?

- www.rimuresearch.com