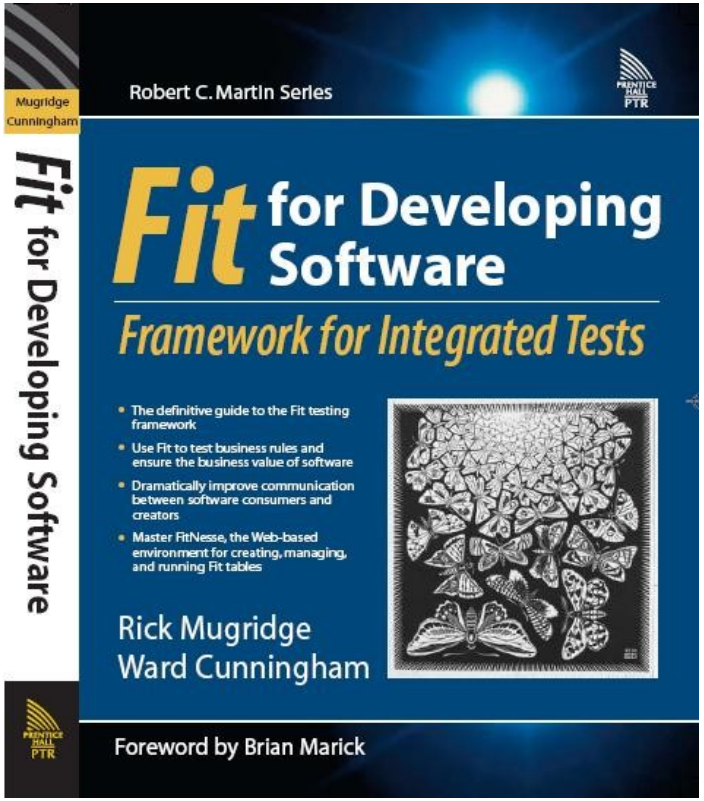
Doubling the Value of Automated Tests: FitLibrary Storytests



Rick Mugridge

www.RimuResearch.com

rick@rimuresearch.com

Automated Testing

- **Manual testing through UI**
- **Record and playback**
  - **Scripts**
- **Test Driven Development**
  - **Kent Beck**
  - **XP practice**
- **Storytest Driven Development (SDD)**
  - **XP practice ("acceptance testing")**
- **My goal is for storytests to support:**
  - **Automated testing**
  - **Detailed requirements**
  - **OOA and Domain Driven Design**

Grafiti

- Quality first
- Fast feedback
- DRY: Don't Repeat Yourself
- Emergence + Change
- Priority driven
- Clarity of domain
- Concrete examples: concise + precise

1. System Testing through UI

- Tools
  - Lots of commercial tools (record & playback)
    - Mercury, Rational
  - Open source:
    - Abbot, JfcUnit, HttpUnit, Selenium
- But such tests are:
  - Very slow
  - Verbose, sometimes in weird scripting languages
  - Repetitious as little scope for abstraction
  - Hard to alter as system changes
  - Focussed on traces:
    - Hard to see beyond the detail
    - Don't support clear thinking about the domain

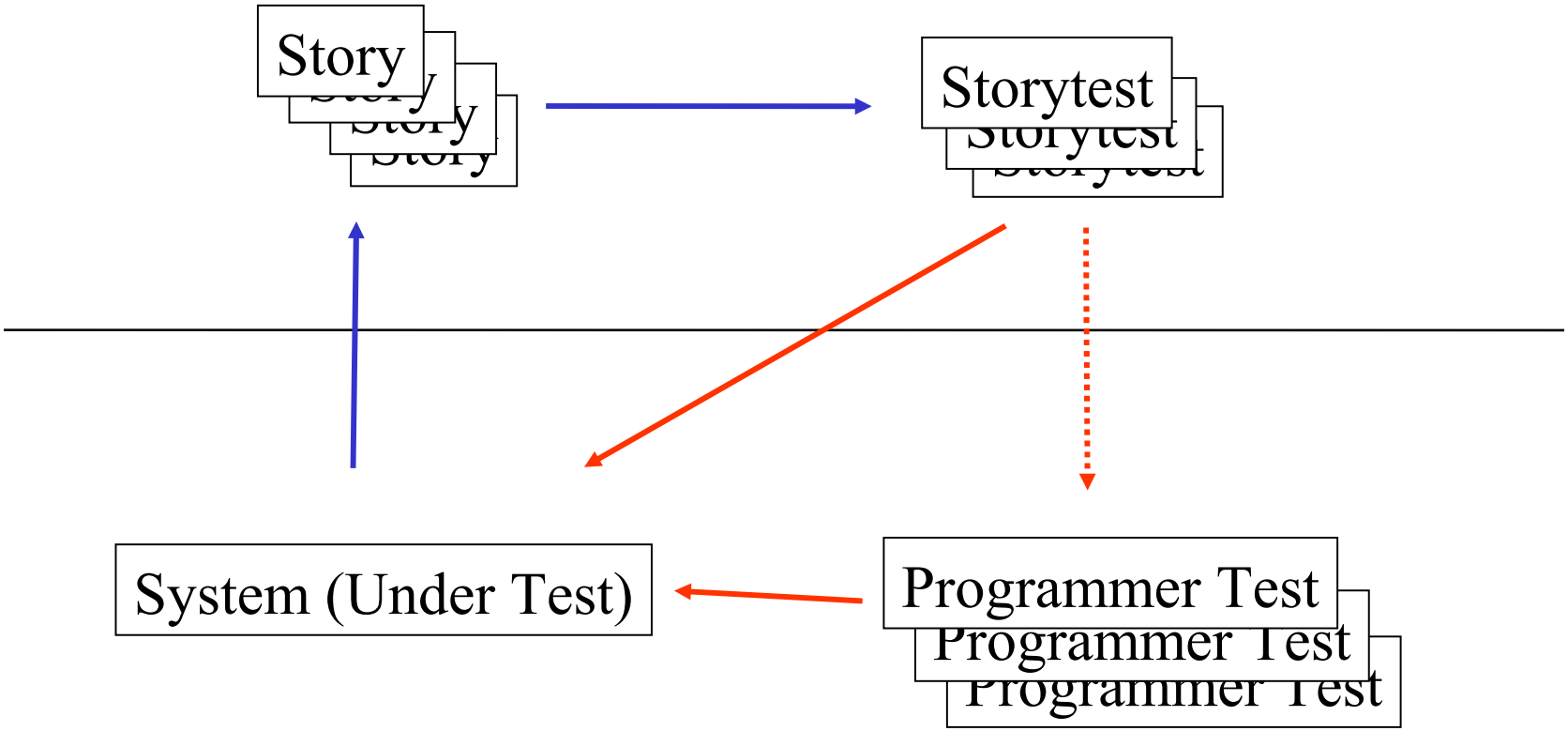
Quality Assurance

- 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
- Good evidence for:
  - Try to go fast → Poor quality → go slow
  - High quality → can go fast
- Switch emphasis to Quality Injection
  - Upfront tests
  - Avoid premature and unnecessary commitment

2. Test Driven Development

- Programmers drive development with unit tests
  - Specify, 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?
  - BAs can't or don't want to read code
  - UI-based testing is likely to be premature

3. Storytest Driven Development



Storytest Driven Development: Joshua Kerievsky, IndustrialXP

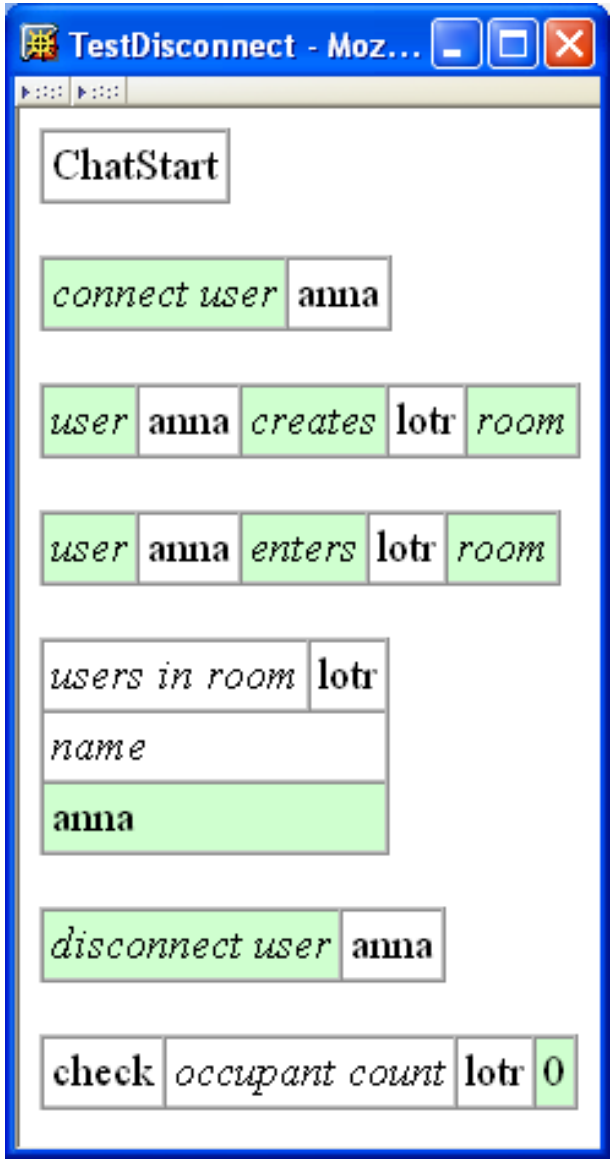
Fit

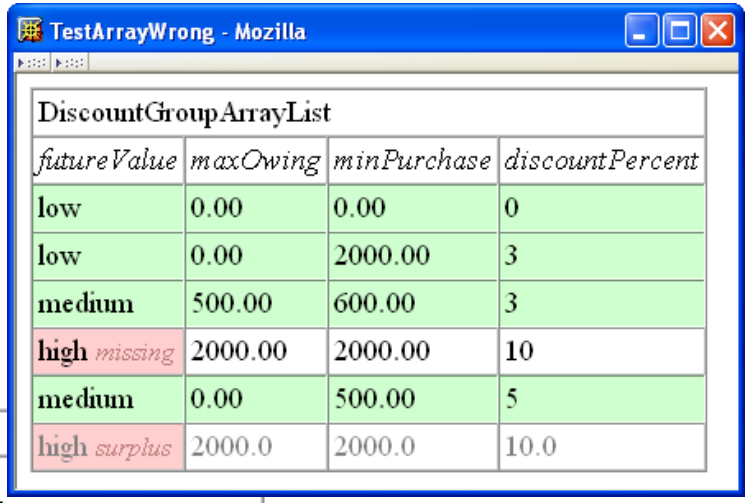
- Fit: Ward Cunningham in 2002, in Java
  - Versions for C#, C++, Python, Ruby, Smalltalk, ...
- Tables: storytest and report
- Fixtures mediate

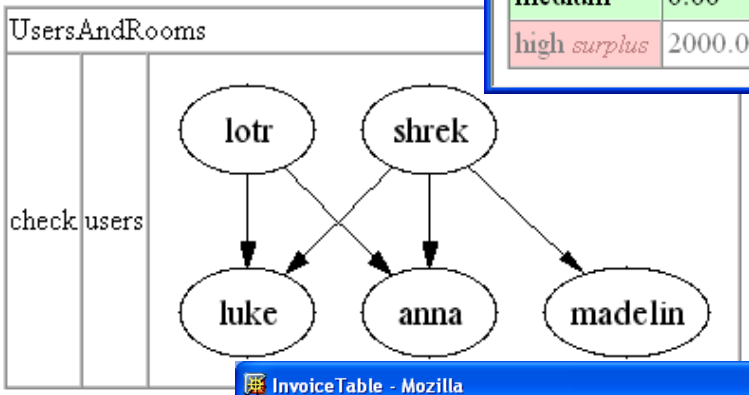
- Tables (and supporting text, etc) from:
  - HTML, FitNesse, Excel, MS Word, etc

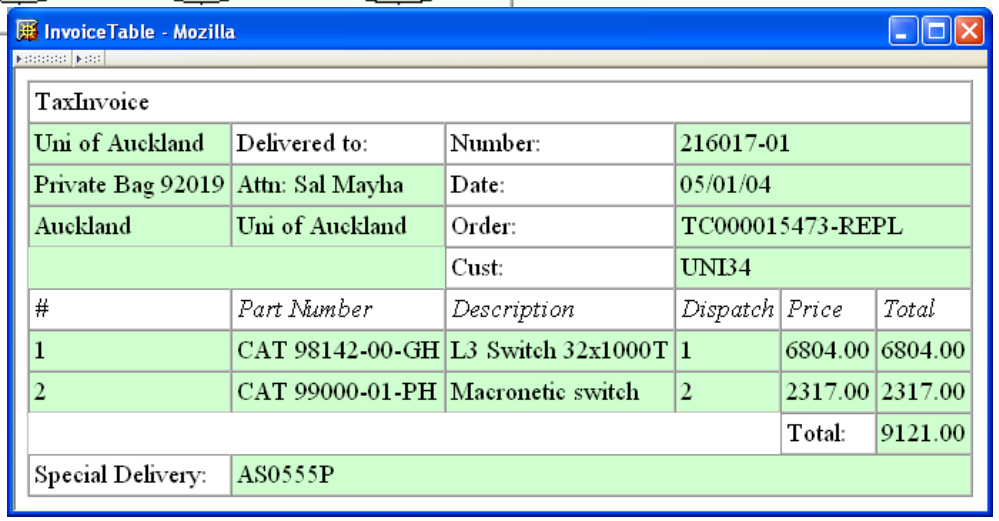
- FitNesse (Micah and Bob Martin)
- FitLibrary (Rick Mugridge)
  - Open source
  - Versions for Java, Python, Smalltalk, C#
  - Thousands of users

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
lotr
shrek
luke
anna
madelin


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

Storytests in Fit are a good start

- Think about testing much earlier
- Storytests
  - Help communicate specifics of story
  - Drive development for whole system
    - With TDD used deeper in system
  - Help tell when a story is done
  - Support evolution
- BUT, storytests often:
  - Written like traditional UI tests
  - Workflow-centric
  - Don't express the business domain
  - Require fixturing code to connect tables to SUT

- What can be done to better support a business perspective?

4. Business Perspective

- **What is the business context?**
- **What are the business problems/opportunities?**
- **What are the requirements?**
- **What is a suitable system design**

- **Spread across:**
  - **Requirements document**
  - **Domain Driven Design's *ubiquitous language***
  - **OOA with UML**
  - **Usage centred design**
  - **Architecture and code**
  - **Unit & System tests**
- **DRY**
  - **Unnecessary redundancy and mappings**

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**
  - **Avoid change if at all possible**
  - **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**

5. 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:**
    - **Business objects + associations ⟶ classes + associations**
    - **Business processes ⟶ services, class methods**
    - **Business rules ⟶ side-effect-free functions**
- **The language evolves as domain undergoes clarification**
  - **As business opportunities better understood**
  - **As business rules emerge and are reframed**

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

FitLibrary Examples

- **In FitNesse...**

Storytests & Domain Driven Design

- Storytests can play a central role in DDD
- Storytests owned by Product Managers
  - Written before the code is developed
  - Eg, Storytests are written the iteration before
- Express business rules as examples
    - Precise, concise, concrete and realistic
    - Develop a suitable language, in tables
- Encourage understanding, discussion and communication
- Evolve as understanding of business needs change
- Significant and valuable business resource

Storytests & Domain Driven Design

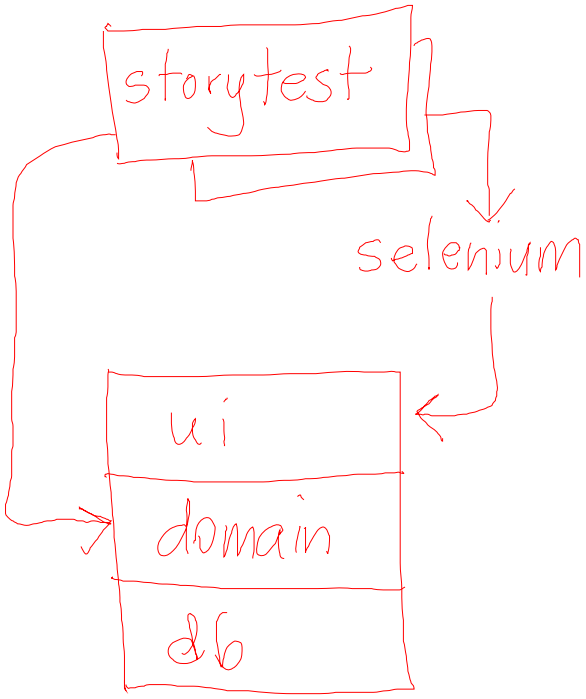
- **Between them, a team contributes:**
  - **Business knowledge**
  - **Testing perspective – eg, bad paths, boundary conditions**
  - **OO modelling perspective**
  - **Skills in developing a *ubiquitous language***
  - **Abstraction skills: refactoring, untangling**
  - **Ability to ask the right questions**

- **Storytests often evolve in response to developers' questions during implementation**

Storytests Encode Domain

- Combine specifications, domain driven design and automated testing
  - Executable Specifications
  - High coverage

- 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



Remaining Challenges

- Expressiveness and avoiding mappings
  - Fit, FitLibrary, FitLibrary2
- Design patterns and table smells
  - DDD patterns
- Organizing large set of storytests
  - Domain
- Suite fixture
- Refactoring, etc tool support
  - Refactor domain
  - Extract calculation and constraint rules
  - DSL
- Different skills

Conclusions

- **Double the value**
- **Avoid unnecessary redundancy**
  - **DRY**
- **Develop domain with storytests**
  - **Concrete examples**
  - **Executable specifications**
  - **Clarify thinking and abstractions**
  - **Communicating**
  - **Align with classes in domain layer**
  - **Testing completion & regression**
  - **Encourage usage centred design**