RE06, Minn
September 14, 2006

Testing to improve requirements – is it mission impossible?

Prepared and presented by

Dorothy Graham

Grove Consultants

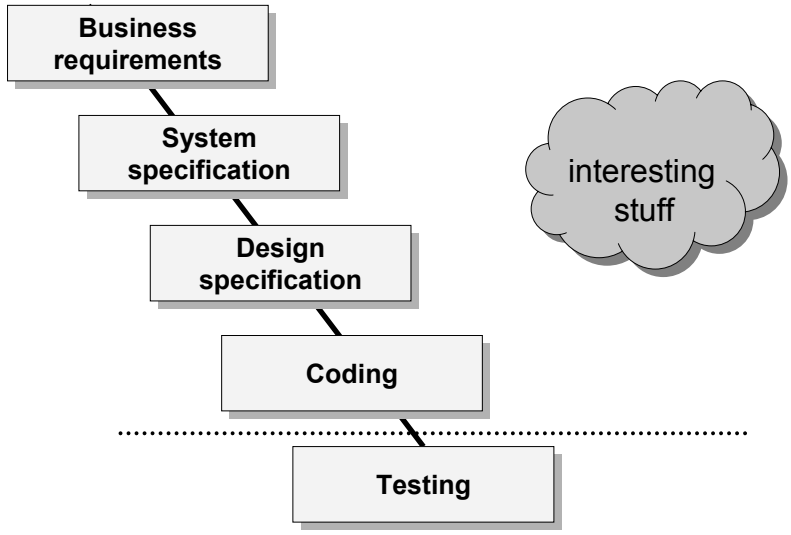
www.grove.co.uk

Testing and Requirements

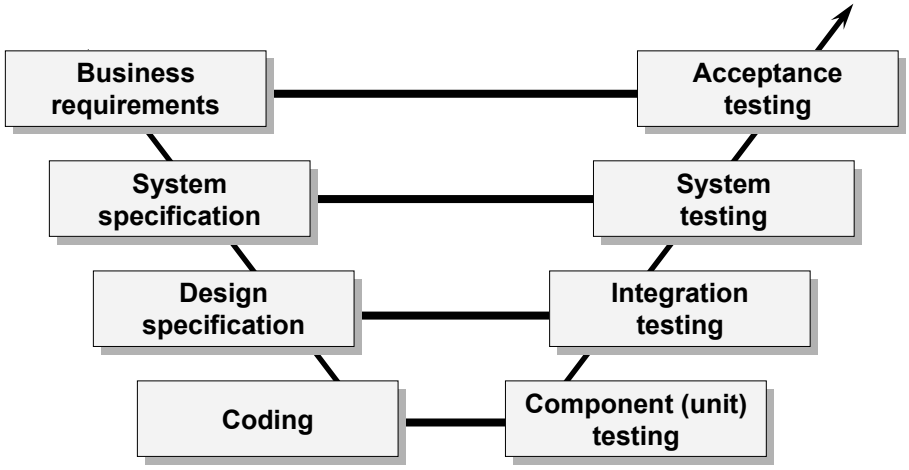
Contents

conference theme:
understanding the stakeholders' desires and needs

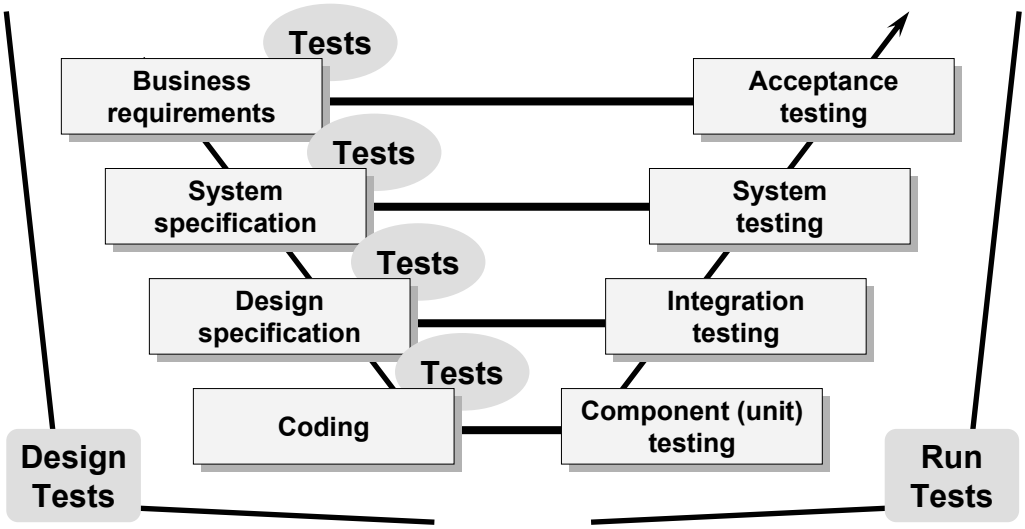
Waterfall model



V-Model



V-Model: early test design

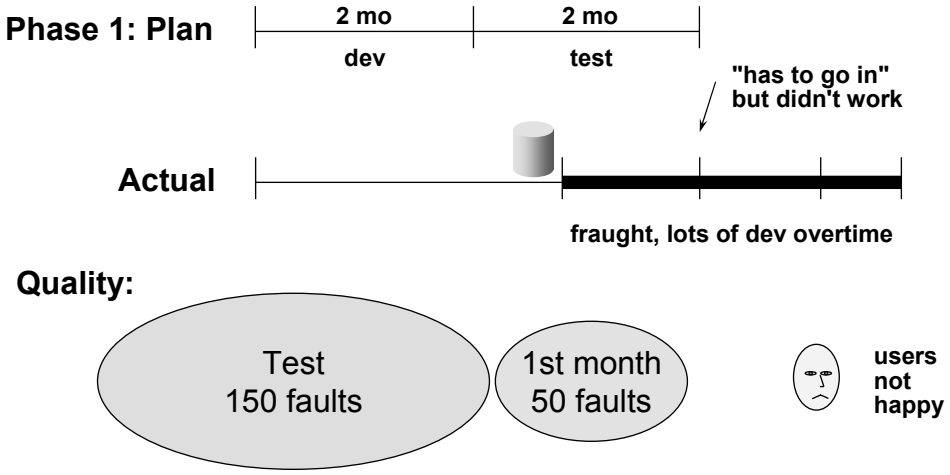


Early test design and defects

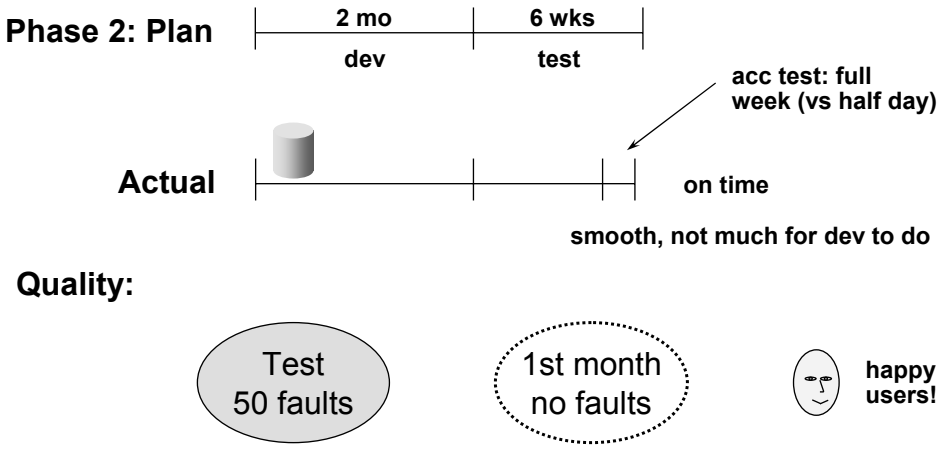
- test design finds defects
- defects found early are cheaper to fix
- most significant defects found first
- defects prevented, not built in
- no additional effort, re-schedule test design
- changing requirements caused by test design

Early test design helps to build quality, stops defect multiplication

Experience report: Phase 1

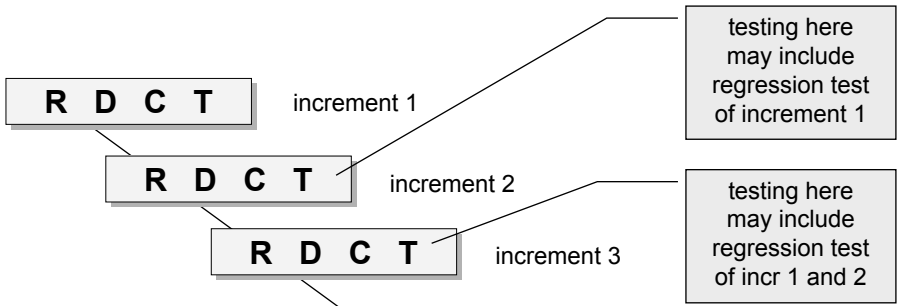


Experience report: Phase 2



Source: Simon Barlow & Alan Veitch, Scottish Widows

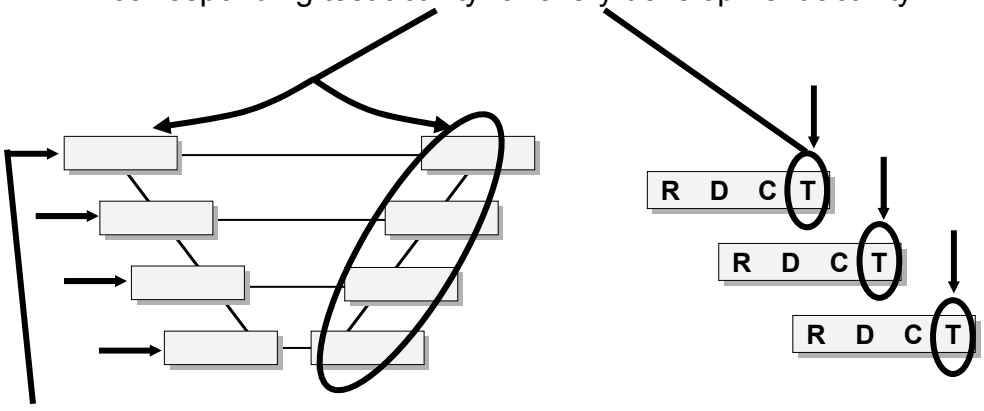
Iterative development



Each increment includes
- **Requirements analysis**
- **Design**
- **Coding**
- **Testing**

Good testing within a lifecycle model

corresponding test activity for every development activity



test objectives for each test level

test analysis and design begins early, testers review development documentation

ISTQB Software Testing Foundation Syllabus, 2005. www.istqb.org

Mindsets

- **requirements engineer**
  - what is needed / wanted?
  - what will help the business?
  - want it to be useful
- **designer / developer**
  - how can I make it work?
  - what's the best way to implement this?
  - want it to be good quality
- **tester**
  - what could go wrong?
  - what exactly does this mean?
  - what if it isn't?
  - what's missing?
  - how could I break it?
  - anti-patterns
  - what would a user do?
  - want it to be useful and good quality
  - if you look for bugs, you are more likely to find them!

Testing and Requirements

Contents

Myth 1: Testing starts at the end

- **requirements come first**
  - "We don't need to think about testing yet – let's just concentrate on requirements"
  - testing is at the end, we're at the start
- **what's wrong with this?**
  - testing can start right at the start
  - thinking about testing early improves requirement specifications early
  - don't have to wait to get benefits of a tester view

Myth 2: Can't test till it's there

- **testing the system needs to have the system**
  - "We can't do any testing because nothing has been built yet."
  - "Testers just play with the system and see what happens"
  - "Anyway, you can't test a piece of paper!"
- **what's wrong with this?**
  - testing is more than testing, and starts before testing
  - misconception: testing = test execution

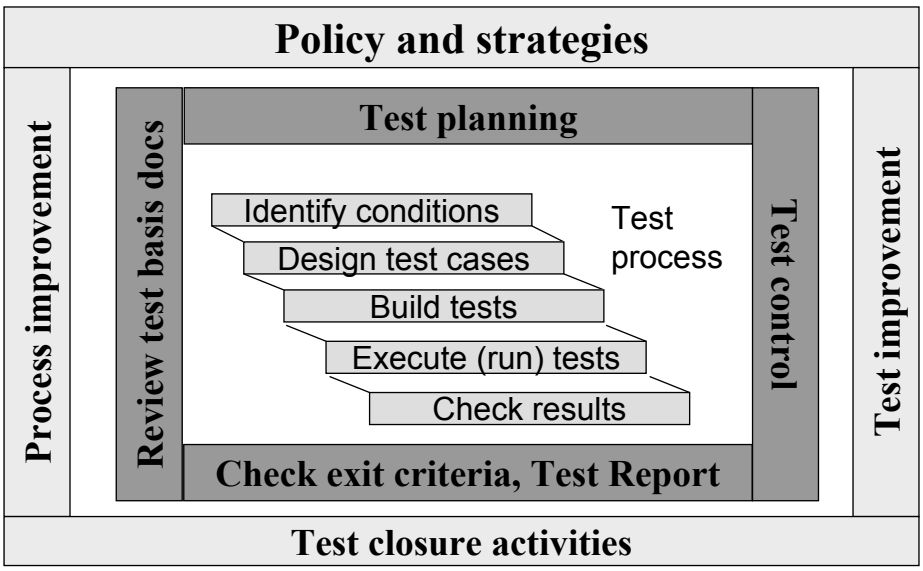
Example requirements

- facilities are required to enable the treasurer to update the account information such as when members pay their subscription fees.
- the system will be required to produce reports giving information about who has paid membership fees, etc.
- the system must be fast. Many people must be able to access the website concurrently.

How would you test this spec?

- **a computer program plays chess with one user. It displays the board and the pieces on the screen. Moves are made by dragging pieces.**

What is testing?



Myth 3: Requirements to test is a one-way street

- **testing uses requirements, not vice versa**
  - "You don't test requirements, you test FROM requirements"
- **what's wrong with this?**
  - thinking about testing raises questions on the requirements
  - test design can lead to improved requirements
    - boundary value analysis
    - decision tables (example ->)

Example requirement

conditions/causes

actions/effects

- Sue has a number of jobs to do on a Saturday but this is dependent on various circumstances.
- if she wakes up early and the weather is sunny she needs to cut the grass. However if she sleeps late and it is sunny then she hangs the washing out.
- if she wakes up early and the weather is not so good and she has some cash in the bank then she will need to go shopping.

- **what if she sleeps in, it's raining, and she has cash in the bank?**

how easy is it to answer this question?

Clearer requirement

how easy is it to answer the question? (sleep in, rain, cash)

Condition/cause								
Up early	**T**	**T**	**T**	**T**	**F**	**F**	**F**	**F**
Sunny weather	**T**	**T**	**F**	**F**	**T**	**T**	**F**	**F**
Cash in the bank	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**
Action/effect								
Cut the grass	**T**	**T**	**F**	**F?**	**F**	**F**	**F?**	**F?**
Hang washing out	**F**	**F**	**F**	**F?**	**T**	**T**	**F?**	**F?**
Go shopping	**F?**	**F**	**T**	**F?**	**F**	**F**	**F?**	**F?**
Tags:	***A***	***B***	***C***	***D***	***E***	***F***	***G***	***H***

"spec" covered only 5 out of 8 combinations!

? = assumption

Myth 4: Tests are for testers only

- **writing good tests is purely a testing concern**
  - "The testers seem to have problems writing tests from our requirements -
  - maybe we should get some better testers!"
- **what's wrong with this?**
  - ambiguous specifications – not testable
  - non-functional quality attributes
    - e.g. "user friendly", "very reliable"
  - if you don't know how to test it, how can you know how to build it?

Non-functional testing

- **testing of software product characteristics**
  - "how" the system works
  - quantified on a varying scale (e.g. response time)
- **performed at all test levels**
- **including the following types:**
  - performance
  - load
  - stress
  - interoperability
  - maintainability
  - reliability
  - portability
  - usability

ISO 9126: Software Engineering: Software Product Quality

Which of the following are testable?

- all help messages are meaningful to the users **Yes**
- context sensitive help available on all fields **Yes**
- all users must like all aspects of the system including reports and screens **No**
- the system must be user-friendly **No**
- the system must be intuitive **No**
- navigation must be consistent across all applications **Yes**
- exit/escape keys must be clearly labelled **Yes**
- entering a new record must be achieved in less than 20 keystrokes **Yes**

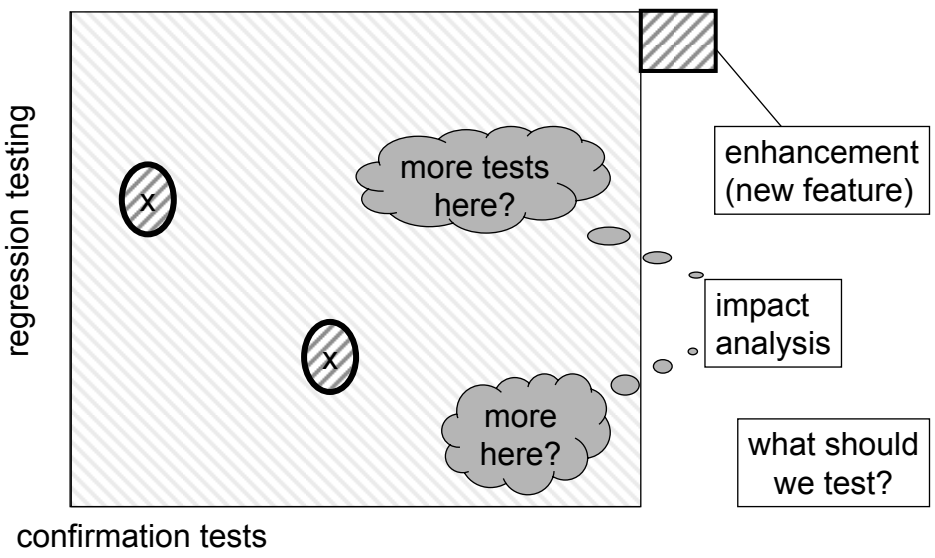
Tom Gilb, Principles of Software Engineering Management, 1988, or gilb.com

---

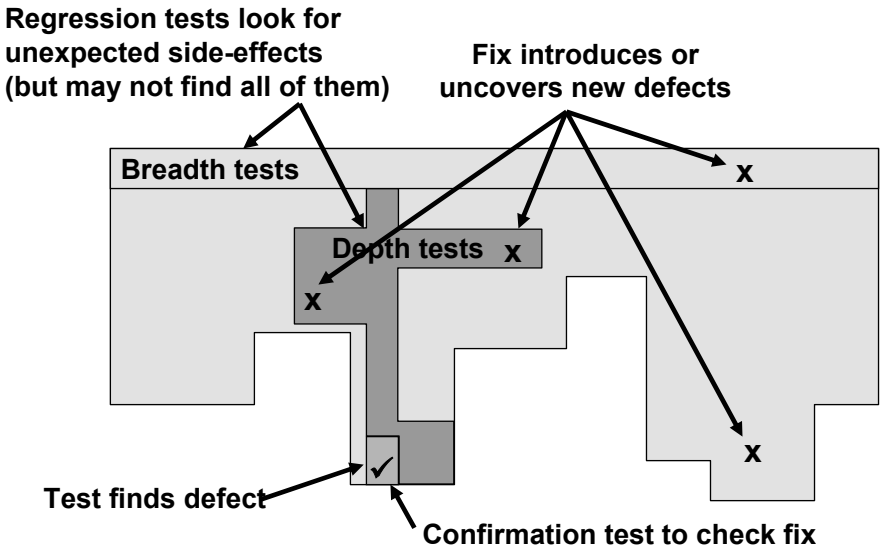
Myth 5: Minor changes are minor

- **minor requirements changes don't matter (much)**
  - "Just add a couple more spaces to this input field. There's plenty of room on the screen."
  - "It's only a minor change; it won't need testing"
- **what's wrong with this?**
  - impact on implementation (e.g. database, checking)
  - impact on testing
    - what unexpected side-effects?
  - size of change NOT = size of testing

Small change ≠ small testing



Confirmation vs. regression testing



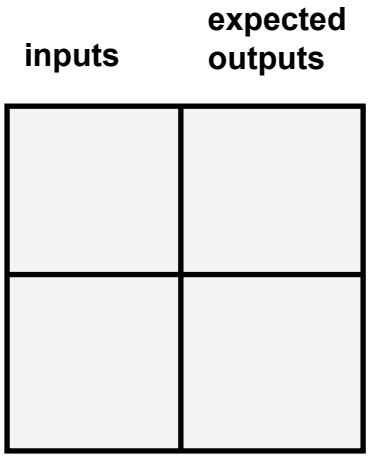
Myth 6: Testers don't need requirements

- **requirements are nice to have but not essential**
  - "We know the requirements aren't great [there], but just test it anyway as best you can."
  - "Just see what the system does."
- **what's wrong with this?**
  - we still need to test somehow
  - what is the test oracle?
  - test that the system does what the system does?
    - not a test! test against what the system SHOULD do

A test

A Program:

```
Read A
IF (A = 8) THEN
   PRINT ("10")
ELSE
   PRINT (2*A)
```



Source: Carsten Jorgensen, Delta, Denmark

Myth 7: Can't test without requirements

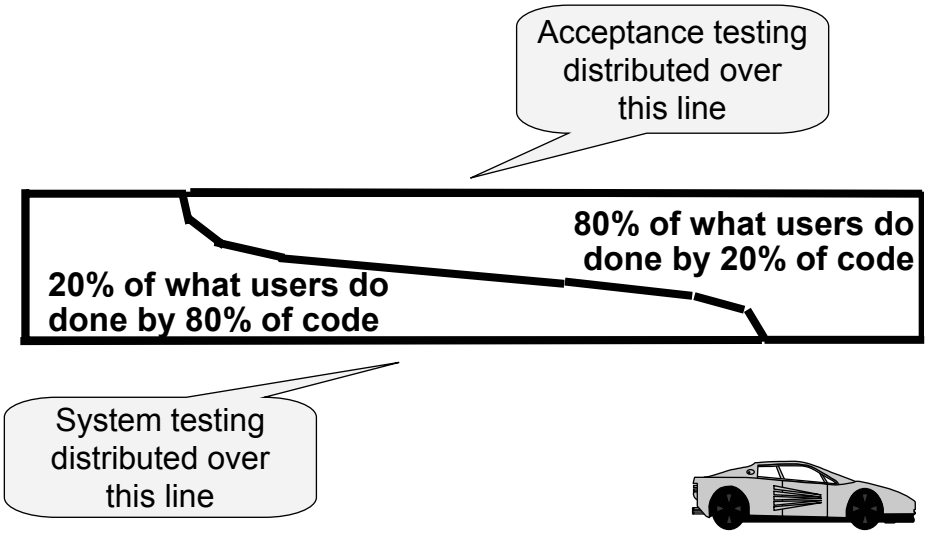
- **testers MUST HAVE requirements**
  - "We can't test until we have decent requirements"
  - the tester's excuse?
- **what's wrong with this?**
  - yes, a test oracle is needed
  - not an excuse to avoid testing
  - more responsibility on the tester
  - exploratory testing is designed for severe time pressure and poor or non-existent requirements

---

Myth 8: Follow the elephant

- **mainstream is more important**
  - "We need to specify what the users do in their normal work."
  - "Of course, there will be exceptions, but these don't happen often, so they're not important
- **what's wrong with this?**
  - yes, normal use is important
  - but exceptions must also work correctly

User Acceptance testing



Acceptance testing is unfair!

purchasers / users	suppliers / developers
no requirement changes	changing requirements
decision pressure	exception details
business needs	psychic specification
technical jargon	no technical understanding
timescales and budgets	delays and overruns
screens still have errors	screen formats
-> acceptance to retaliate	-> acceptance nit-picking

Testing motto

If you don't have patience to test the system

the system will surely test your patience

Testing and Requirements

Contents

Improved requirements through testing

- **get testers involved early**
  - start test activities at the beginning
  - invite testers to requirements reviews
- **use the tester perspective / mindset**
  - with every requirement, ask:
    - what could go wrong? what if it isn't?
  - ask for (and appreciate) feedback from testers
- **technical aspects**
  - include examples, business scenarios, use cases
  - non-functional requirements: measurable & testable
- **communicate with testers: common goals**

Testing and Requirements

Summary: key points

Improving requirements through testing is not only "mission possible" – it's "mission critical" to understand stakeholders' desires and needs

Good requirements engineering produces better tests; good test analysis produces better requirements

Shameless commercial plug

download IEEE Software article, Sep/Oct 2002 from
www.grove.co.uk (downloads – "paper on requirements")
copy of slides: DorothyRGraham@aol.com or USB stick