Requirements and Testing
as Risk Minimisation

Emmanuel Letier
http://letier.cs.ucl.ac.uk

London, February 2013
Engineering has fundamental laws

“Every body perseveres in its state of being at rest or of moving uniformly straight forward, except insofar as it is compelled to change its state by forces impressed.”

Newton’s First Law of Motion
Law #1: Boehm’s cost-to-fix curve

- Hasty generalisation?
- Misrepresentations?
Influence on Requirements and Testing

**Goal:** Reducing defects

It is critical to **minimise requirements defects** as these are the most costly.

Over time, we lost sight of the ultimate goal!

It is critical to **find the maximum number of bugs at the least possible cost**.
What is the ultimate goal of software engineering?
The ultimate goal of software engineering is ...

A. To deliver software on time
B. To deliver software on budget
C. To deliver software with low number of bugs
D. All of the above
E. None of the above
The ultimate goal of software engineering is ...

A. To deliver software on time
B. To deliver software on budget
C. To deliver software with low number of bugs
D. All of the above
E. None of the above
F. To deliver software that provides value to its clients
   (or no software at all if there are better ways to provide value)
Beware of local optimisations

Delivering on time, on budget, with low defect rate doesn’t necessarily provide value (e.g. UK police mobile handsets)

Minimising requirements defects (ambiguity, incompleteness, etc.) doesn’t necessarily yield the most valuable system
Law #2: Wieger’s Law of Requirements Ambiguity

“The requirements may be ambiguous but the product will be definite”
When are requirements good enough?

When is testing good enough?

When the code coverage target is achieved
When are requirements good enough?

When the risks of building the wrong product are acceptable

When is testing good enough?

When the risks of software failure are acceptable

Requirements and testing are about understanding and minimising risks (the risks of failing to deliver value)
Question: What comes next in the talk?

A. Testing as risk minimisation
B. Requirements as risk minimisation
C. All of the above
D. None of the above
Testing as Risk Minimisation

“All bugs are equal, but some bugs are more equal than others”

Severity of failure caused by fault
- Safety or business critical failure
- User annoyance

Likelihood that fault will cause failure
Take testing out of its boxes

Optimising testing for code coverage or bug counts

Optimising testing for bug severity by looking at impact of bugs in the World
Requirements as Risk Minimisation

“All requirements defects are equal, but some requirements defects are more equal than others”

Severity of problem caused by defect
- Safety or business critical failure, *architecture breaker*
- User annoyance

Likelihood that defect will cause problem
Ideas

Requirements risks tend to be severely underestimated

Good RE techniques exist but are not applied

Make requirements risks more visible

Make requirements risks more actionable

We need a **scientific (evidence-based)** approach to requirements risks management
An evidence-based approach

- Early Requirements Risk Warning System
- Requirements Risk Decision Support System
- Predictive models
- Economic & goal models
- Historical data
  - Software Repositories
  - Project Mgmt Repositories
  - Business Data
Progress

First Step: Failure Prediction in Feature Requests
(Fitzgerald, Letier, Finkelstein @ RE’11, REJ 2012)
• Explored feasibility in 6 open-source projects
• Successful predictions but project-specific and no clear causality
• Only considered very basic predictive attributes (discussion lengths, basic word analysis, etc.) and not impact on project-specific goals

Next Steps
• Extend scope to agile & iterative development in brownfield projects
• Improve risk warning and decision support systems
• Main case study: UCL Information Systems Projects
  – representative of other large organisations, e.g. Gvrt Dprt
What's the ultimate goal of software engineering?

Business Value over effort and defect

Beware of local optimisations

Requirements and Testing as Risk Minimisation

Take testing out of its boxes

Make requirements risks more visible and actionable

Moving away from counting bugs

Time to consider bugs severity

Moving away from documentation-centric perspective

Time for a scientific approach to requirements risks
References

