Software Testing -

A Project Manager's Secret Weapon

Donna O'Neill, IV&V Australia

Overview

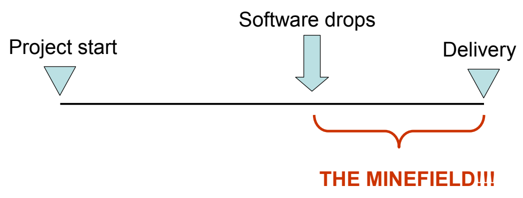
- Moving beyond the "testing is good" mindset
- How can testing be the PM's secret to success?
- How to turn testing into an effective secret weapon

Moving beyond the "testing is good" mindset

Test maturity starts with…	immaturity – relying on ad-hoc and isolated testing processes
Then progresses to…	the desire to "fix" testing – *but don't bother the developers while you're doing it*
Then grows into…	the realisation that you can't "fix" testing in isolation – and that it isn't just a *testing* problem in the first place
And comes of age when…	testing is effectively incorporated into the development lifecycle, and so projects are more controllable and product quality increases

What can happen to testers…

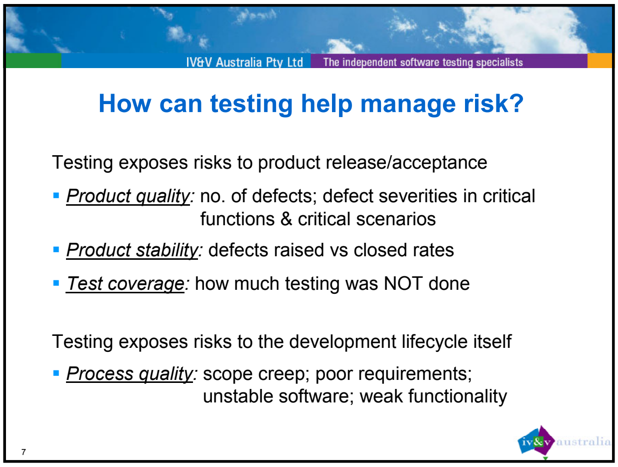
Why does that happen?



How can testing be the PM's secret to success?

Testing is a powerful risk management tool

Used well, it can provide visibility and control of the project, and increase your chance of a positive business outcome



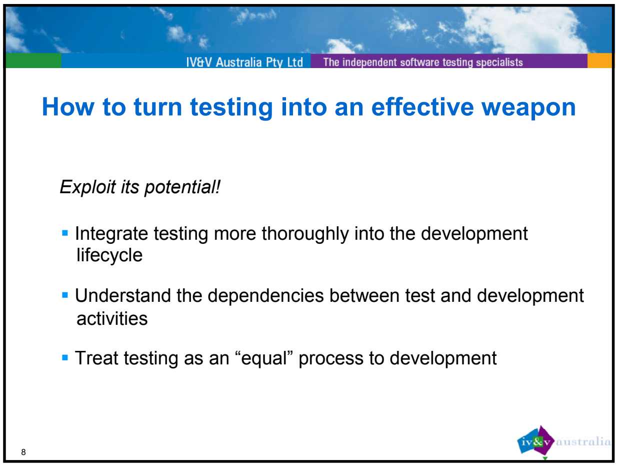
How can testing help manage risk?

Testing exposes risks to product release/acceptance

- *Product quality*: no. of defects; defect severities in critical functions & critical scenarios
- *Product stability*: defects raised vs closed rates
- *Test coverage*: how much testing was NOT done

Testing exposes risks to the development lifecycle itself

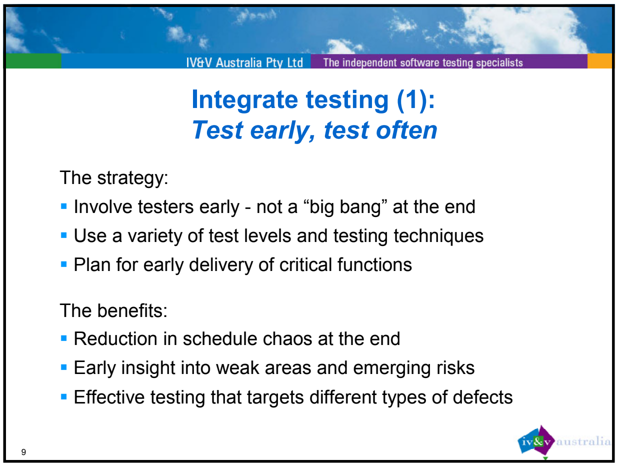
- *Process quality*: scope creep; poor requirements; unstable software; weak functionality



How to turn testing into an effective weapon

Exploit its potential!

- Integrate testing more thoroughly into the development lifecycle
- Understand the dependencies between test and development activities
- Treat testing as an “equal” process to development



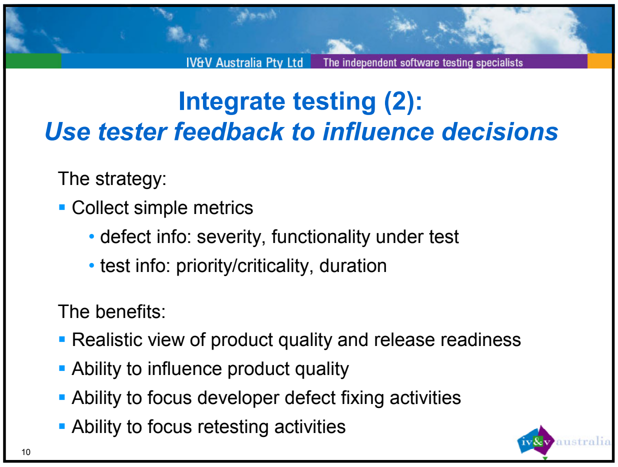
Integrate testing (1): *Test early, test often*

The strategy:

- Involve testers early - not a “big bang” at the end
- Use a variety of test levels and testing techniques
- Plan for early delivery of critical functions

The benefits:

- Reduction in schedule chaos at the end
- Early insight into weak areas and emerging risks
- Effective testing that targets different types of defects



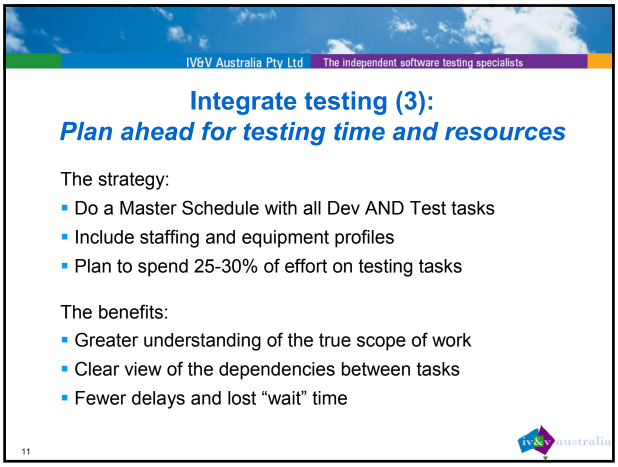
Integrate testing (2): *Use tester feedback to influence decisions*

The strategy:

- Collect simple metrics
  - defect info: severity, functionality under test
  - test info: priority/criticality, duration

The benefits:

- Realistic view of product quality and release readiness
- Ability to influence product quality
- Ability to focus developer defect fixing activities
- Ability to focus retesting activities



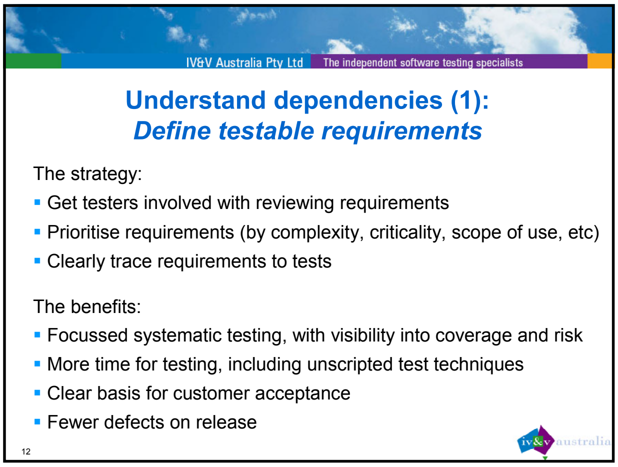
Integrate testing (3): *Plan ahead for testing time and resources*

The strategy:

- Do a Master Schedule with all Dev AND Test tasks
- Include staffing and equipment profiles
- Plan to spend 25-30% of effort on testing tasks

The benefits:

- Greater understanding of the true scope of work
- Clear view of the dependencies between tasks
- Fewer delays and lost “wait” time



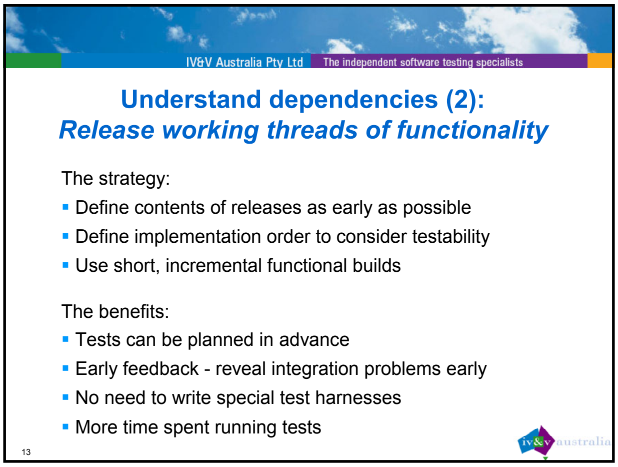
Understand dependencies (1): *Define testable requirements*

The strategy:

- Get testers involved with reviewing requirements
- Prioritise requirements (by complexity, criticality, scope of use, etc)
- Clearly trace requirements to tests

The benefits:

- Focussed systematic testing, with visibility into coverage and risk
- More time for testing, including unscripted test techniques
- Clear basis for customer acceptance
- Fewer defects on release



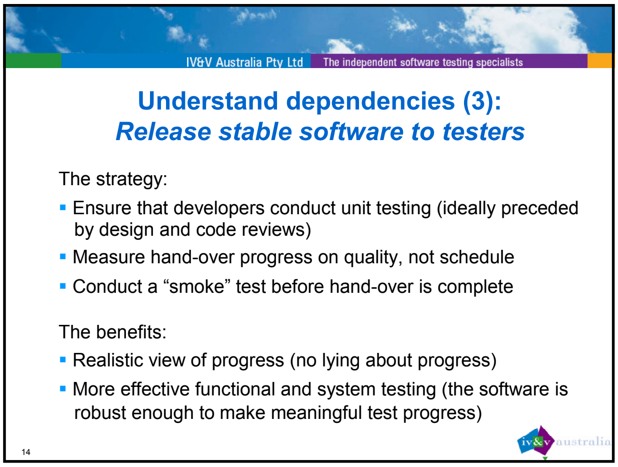
Understand dependencies (2): *Release working threads of functionality*

The strategy:

- Define contents of releases as early as possible
- Define implementation order to consider testability
- Use short, incremental functional builds

The benefits:

- Tests can be planned in advance
- Early feedback - reveal integration problems early
- No need to write special test harnesses
- More time spent running tests



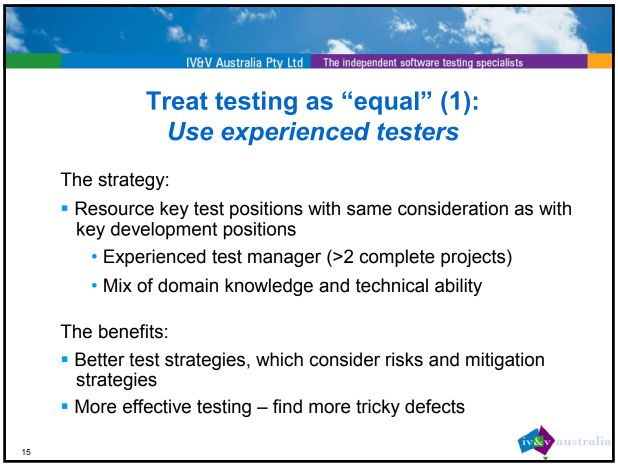
Understand dependencies (3): *Release stable software to testers*

The strategy:

- Ensure that developers conduct unit testing (ideally preceded by design and code reviews)
- Measure hand-over progress on quality, not schedule
- Conduct a “smoke” test before hand-over is complete

The benefits:

- Realistic view of progress (no lying about progress)
- More effective functional and system testing (the software is robust enough to make meaningful test progress)



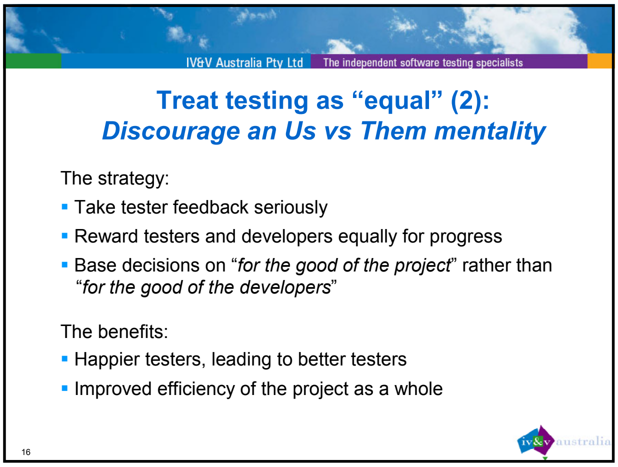
Treat testing as “equal” (1): *Use experienced testers*

The strategy:

- Resource key test positions with same consideration as with key development positions
  - Experienced test manager (>2 complete projects)
  - Mix of domain knowledge and technical ability

The benefits:

- Better test strategies, which consider risks and mitigation strategies
- More effective testing – find more tricky defects



Treat testing as “equal” (2): *Discourage an Us vs Them mentality*

The strategy:

- Take tester feedback seriously
- Reward testers and developers equally for progress
- Base decisions on “*for the good of the project*” rather than “*for the good of the developers*”

The benefits:

- Happier testers, leading to better testers
- Improved efficiency of the project as a whole

How will Project Managers benefit from being smart about testing?

A successful project!

The ability to use testing, and the ***visibility*** and ***control*** that testing activities can bring, to their best advantage in running a successful development project and achieving a positive business outcome

How will Testers benefit from being smart about testing?

More interest, less frustration!

Technical interest: The chance to put our testing knowledge to good use

Social interest: The chance to be an equal part of the project team and make a valuable contribution