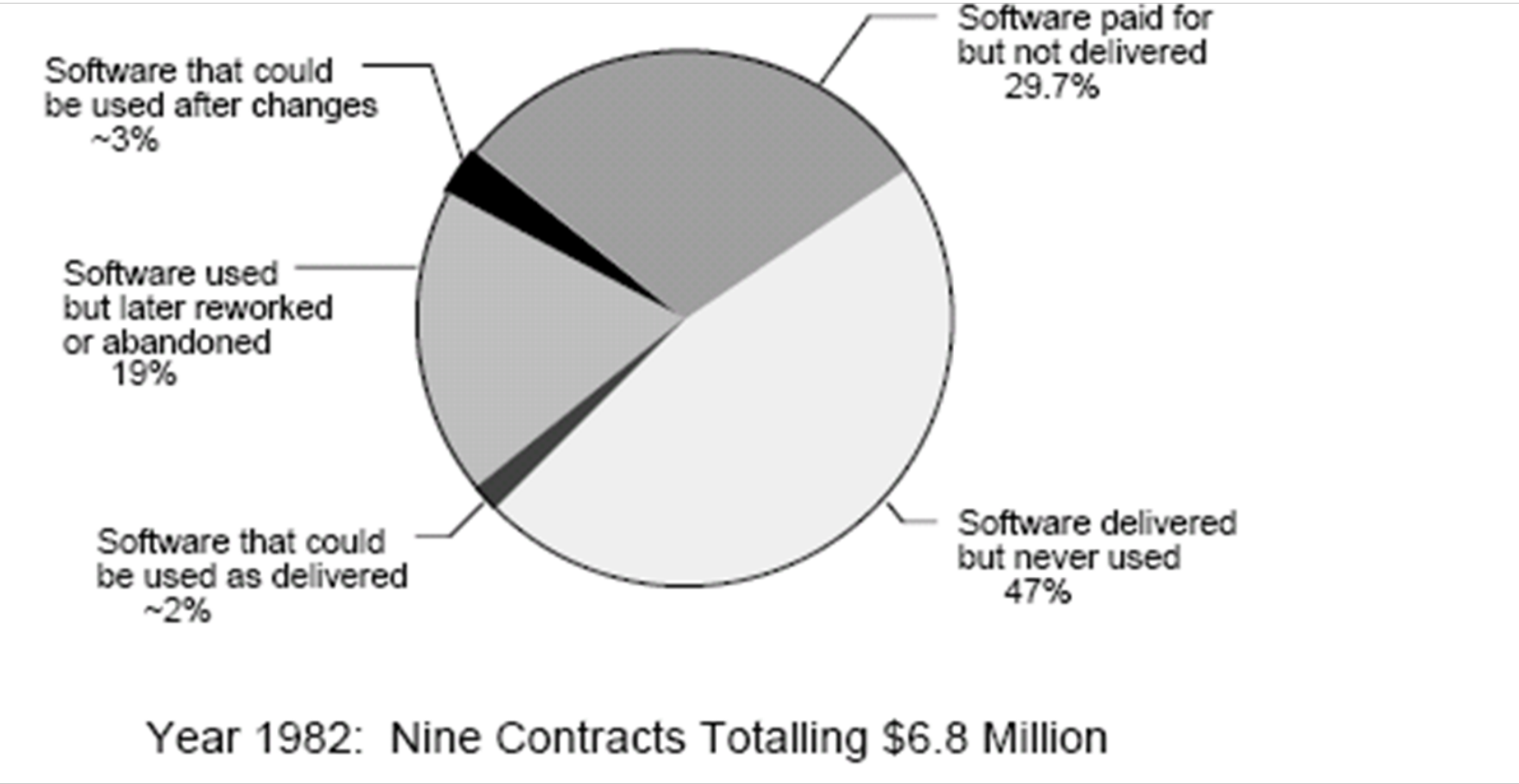
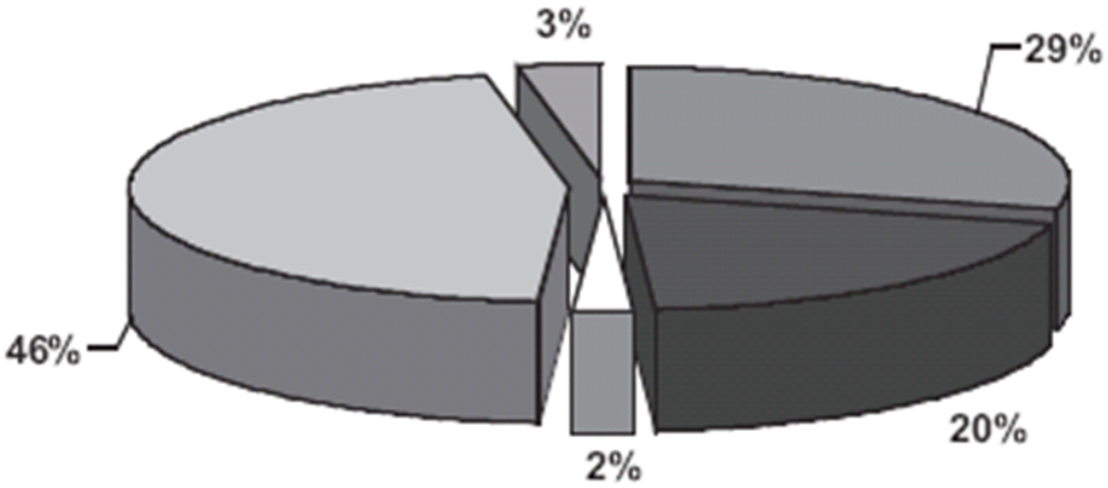
Testing in the Lifecycle

Stuart Anderson

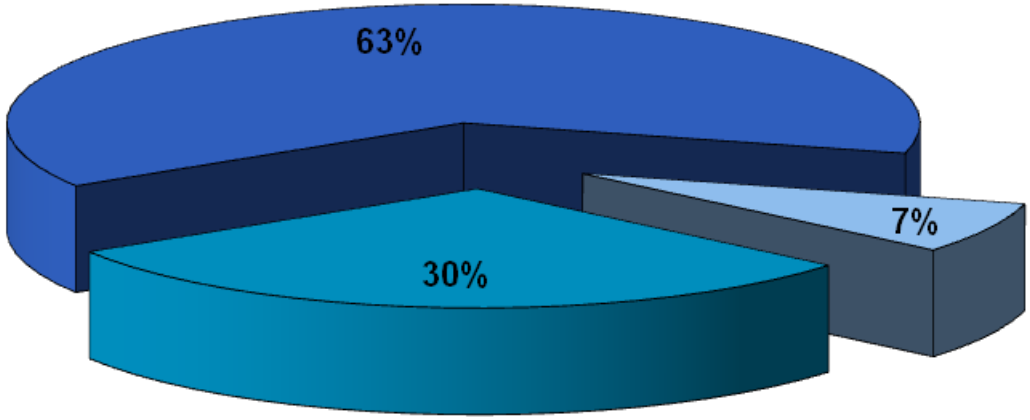
Software was difficult to get right in 1982



It was still difficult in 1995



...and in 2007



- On budget, on time, on spec
- Anything in between
- Never saw the light of day

Source: The Guardian, 18 May 2007
Figures from Department for Work and Pensions spokesman (63%)
And Joe Harley, Chief Information Officer, DWP (30%)

And testing costs are significant

No.	Metric	Project A	Project A1	Project B	Project C
1	Relative Cost of Software Testing (as part of overall project effort)	Test Type: FQT + Unit Testing = 1500 man hrs testing + 2500 man hrs documentation and SQA Total effort = 26000 man hrs Percentage: 15.3%	Test Type: FQT Mainly Testing = 810 man hrs Total effort = 7300 man hrs Percentage: 11%	Test Type: FQT + Unit Testing = 2000 man hrs Total effort = 12000 man hrs Percentage: 16.6%	Test Type: FQT Mainly Testing = 1100 man hrs Total effort = 14000 man hrs Percentage: 8.57%
2	Integration duration as part of overall project development duration	Integration: CSCI = 4.0 mon System =4.0 mon Total: = 24 months Percentage CSCI: = 16.67% Percentage Integ: 16.67%	Integration: = 3.0 mon Total: = 41 months Percentage Integ: 7.3%	Integration: CSCI = 4.0 mon System =9.0 mon Total: = 36 months Percentage CSCI: = 11.1% Percentage Integ: 25.0%	Integration: = 11 mon Total: =36 months Percentage Integ: 30.6%
3	Cost of Requirements Change	16 changes 373 man hrs 23.3 man hrs / change	6 changes 100 man hrs 16.7 man hrs / change	2 changes 26 man hrs 13 man hrs / change	8 changes 70 man hrs 8.75 man hrs / change
4	Functional Coverage	New Video Card, 28 of 35 functions tested Coverage = 80%		----------	----------
5	Code Coverage	According to literature, code coverage is about 55% when blackbox test only is executed [6], [7]			

Cost of Testing vs Cost of Defects

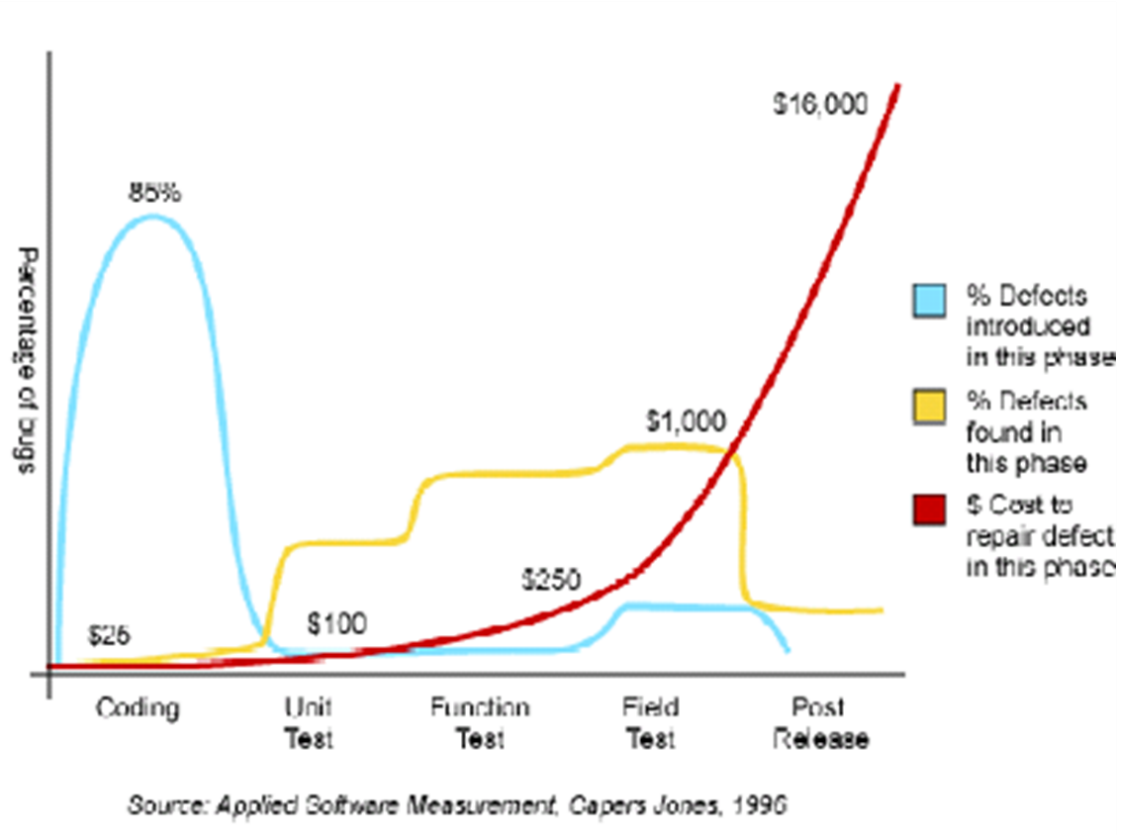
- NIST report (2002): "The Economic Impacts of Inadequate Infrastructure for Software Testing"

  http://www.nist.gov/director/planning/upload/report02-3.pdf

- Notes that "developers already spend approximately 80% of software development costs on identifying and correcting defects".

- "Identifying and correcting defects" not necessarily the same thing as the cost of testing, but still...

Costs of fixing defects found at different stages



Source: Applied Software Measurement, Capers Jones, 1996

Costs of Defects

- Defects in the specification are even more costly to remove if we do not eliminate them early.
- Different software lifecycles distribute testing (verification — *'building the thing right'*, and validation – *'building the right thing'*) differently.
- The different distributions of test activity can have an impact on where bugs are discovered.
- We consider three representative lifecycles and consider where testing is located in each:

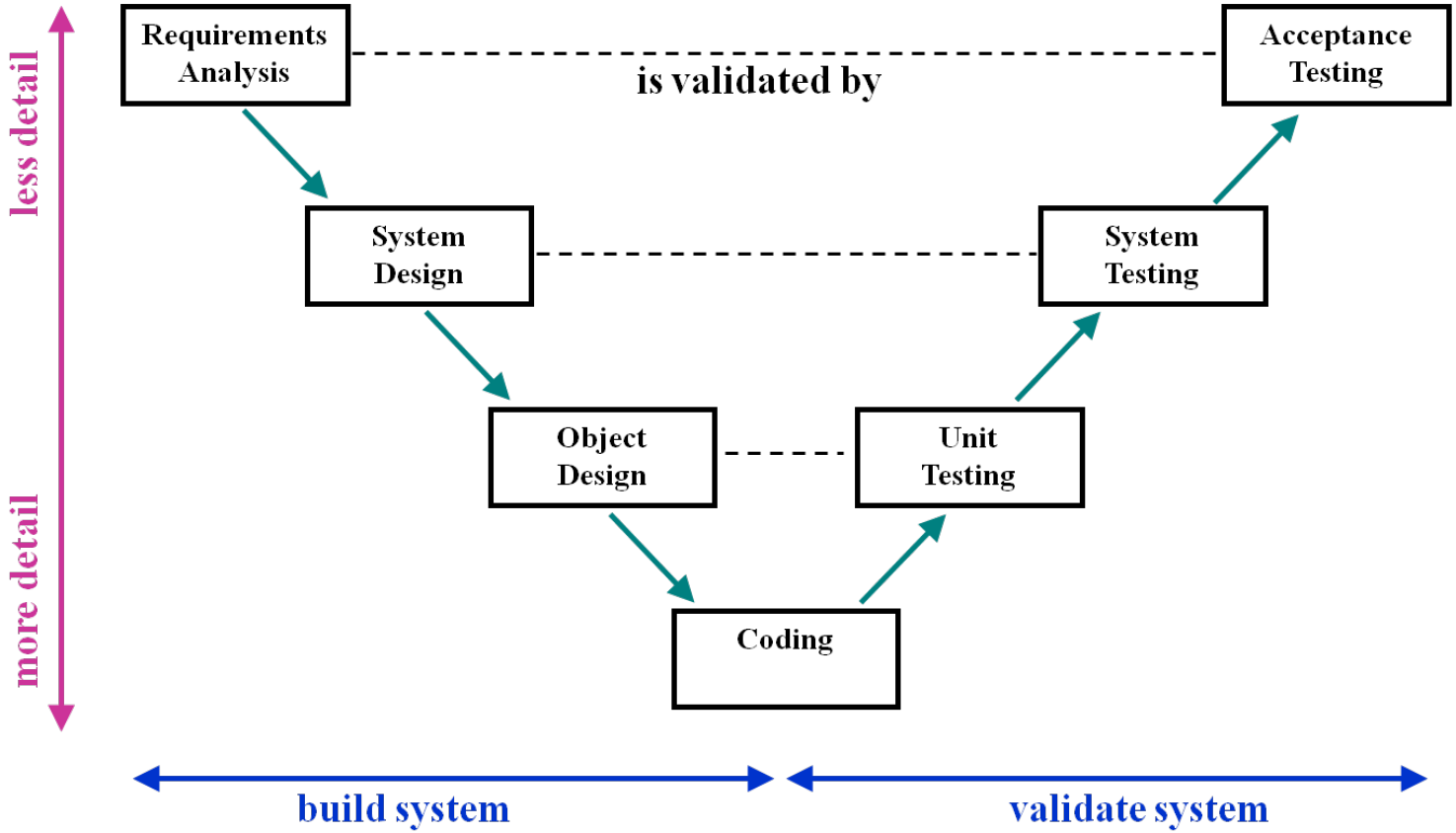
  The V-model, *Boehm's spiral model* and *eXtreme Programming (XP)*

Recap: Waterfall model of software development

1. Requirements
2. Design
3. Implementation
4. Testing
5. Release and maintenance

Sequential, no feedback — Ironically its "author", Royce, presented it as an example of a broken model

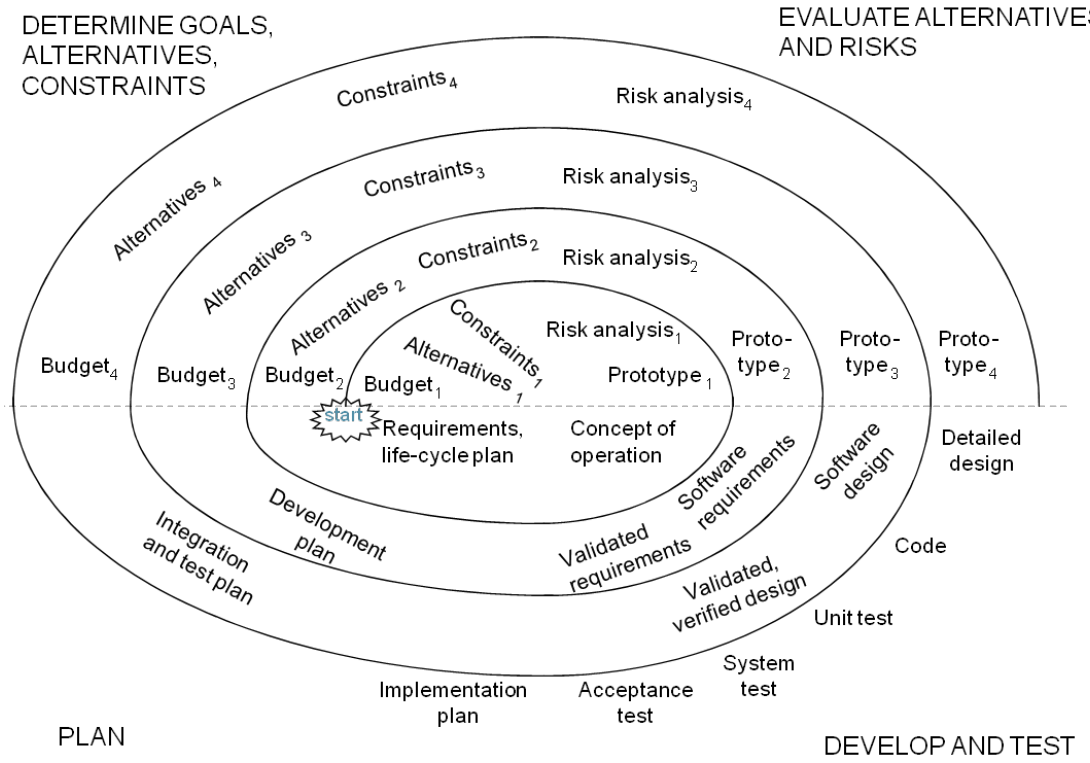
V-model


Requirements Analysis
is validated by
Acceptance Testing
System Design
System Testing
Object Design
Unit Testing
Coding
less detail
more detail
build system
validate system

V-model Rationale

- This is a modified version of the waterfall model.
- Tests are created at the point the activity they validate is being carried out. So, for example, the acceptance test is created when the systems analysis is carried out.
- Failure to meet the test requires a further iteration beginning with the activity that has failed the validation.
- V-model is focused on creating tests in a structured manner.
- It is popular with developers of systems that are highly regulated because it is well suited to creating evidence that can be used to justify a system to a regulator.

Boehm's Spiral Model



Spiral Model Rationale

- The spiral model is focused on controlling project risk and attempting formally to address project risk throughout the lifecycle.

- V&V activity is spread through the lifecycle with more explicit validation of the preliminary specification and the early stages of design. The goal here is to subject the early stages of design to V&V activity.

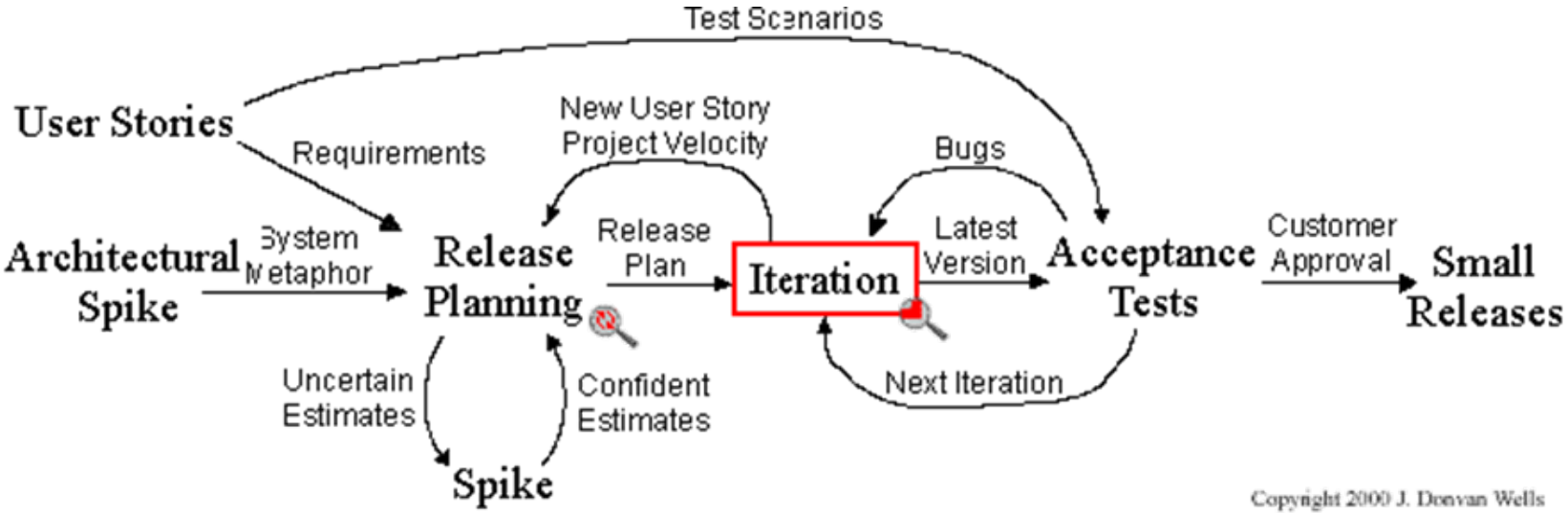
- At the early stages there may be no code available so we are working with models of the system and environment and verifying that the model exhibits the required behaviours.

XP principles

- eXtreme Programming advocates working directly with code almost all the time.
- The 12 principles of XP summarise the approach.
- Development is test-driven.
- Tests play a central role in refactoring activity.
- "Agile" development mantra: Embrace Change.

1. Test-driven development
2. The planning game
3. On-site customer
4. Pair programming
5. Continuous integration
6. Refactoring
7. Small releases
8. Simple design
9. System metaphor
10. Collective code ownership
11. Coding standards
12. 40-hour work week

eXtreme Programming (XP)



http://www.extremeprogramming.org/map/project.html

Summary

- We have considered three different approaches to the lifecycle and have seen how testing fits in the lifecycles.
- Each approach will have a different testing cost and cost-profile through the lifecycle.
- Lifecycles are often dependent on the type of product and how well we understand project risk.

Required Readings

- **Textbook (Pezzè and Young):** Chapter 4, Test and Analysis Activities within a Software Process

- **Textbook (Pezzè and Young):** Chapter 20, Planning and Monitoring the Process

Suggested Readings

- Gregory M. Kapfhammer, Software Testing, In A. Tucker (Ed.), Second Edition, The Computer Science and Engineering Handbook, Chapter 105, CRC Press, 2004.
- Mary Jean Harrold. 2000. Testing: a roadmap. In Proceedings of the Conference on The Future of Software Engineering (ICSE '00). ACM, New York, NY, USA, 61-72.
  DOI: http://dx.doi.org/10.1145/336512.336532
- Winokur, M.; Grinman, A.; Yosha, I.; Gallant, R.; Measuring the effectiveness of introducing new methods in the software development process, Euromicro Conference, 1998. Proceedings. 24th , vol.2, no., pp.800-807 vol.2, 25-27 Aug 1998.
  DOI: http://dx.doi.org/10.1109/EURMIC.1998.708105