W4156

Debugging & Defect Management

Agenda

Lots of activities and practices. Feels like learning to juggle.

- ❏ Metaphors for *learning* Software Engineering
- ❏ Metaphors for Software Engineering Projects
  - ❏ How activities fit together
- ❏ Recap / Walk through key learning outcomes

Learning Software Engineering

Learning Software Engineering

+

=

Lots of skills that need to come together.

Overall Software Engineering Projects

(I have used manufacturing analogies to refer to the limited scope of written code to production. This is the ‘production line’. However, this does not apply to the design phase / wider project)

Problem Domain

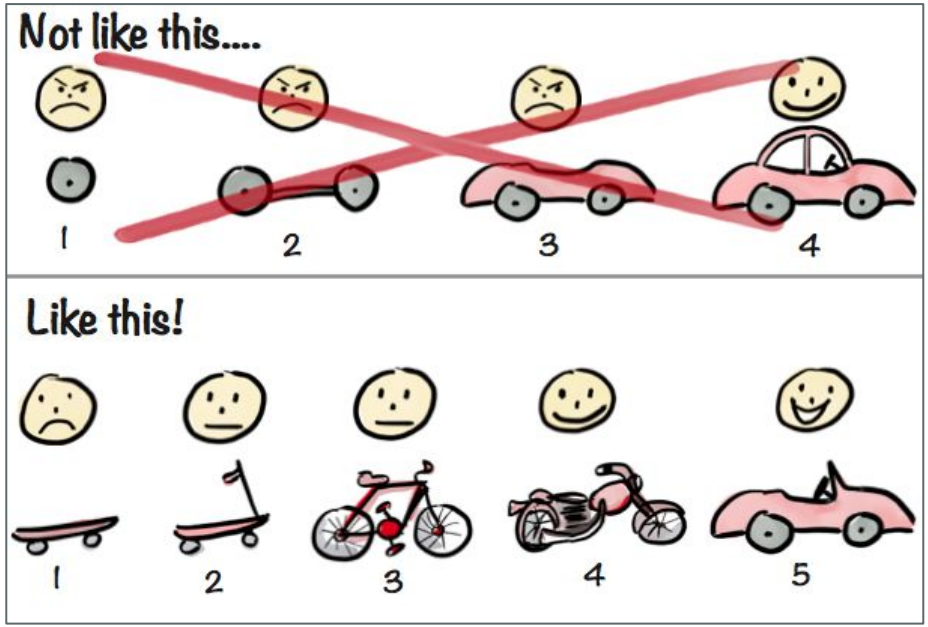
Software is not the same as house building, See rebuttals here (XP is method under agile umbrella), here, here, here, here.

The semantic collision of ‘software architect’ with ‘building architect’ is problematic as it implies that software can be built in the same ways as houses:

- How novel are houses?
- Can you add a floor once it is started?
- Can you run 10 copies of a house?
- Can you model a house before it is built? Can you do the same with software?
- Can you ‘test’ a house works before it is built?
- Are most houses fraught with civil engineering unknowns?

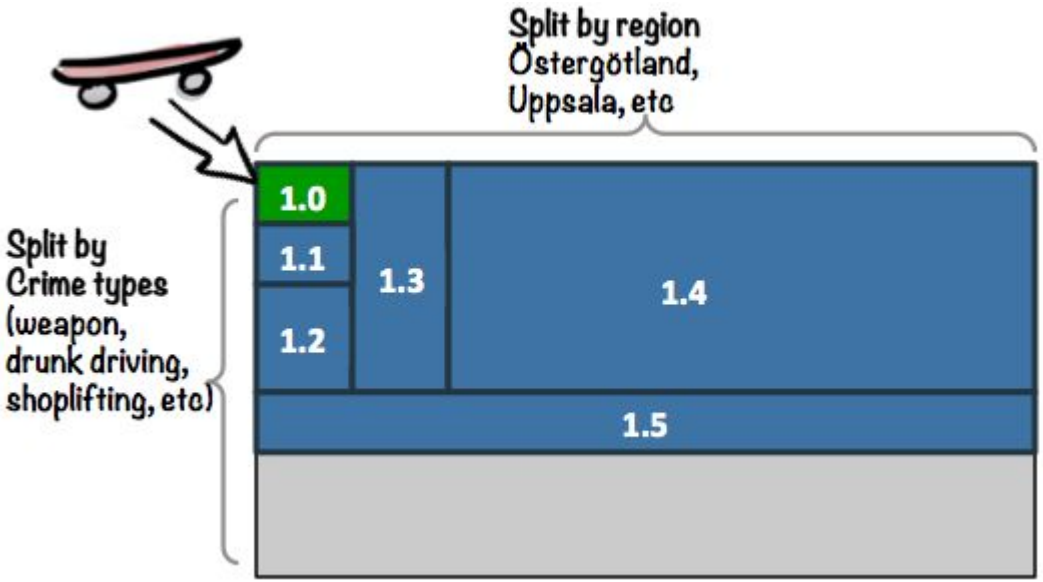
Hence .... Software is generally not built sequentially ...

The Agile Car



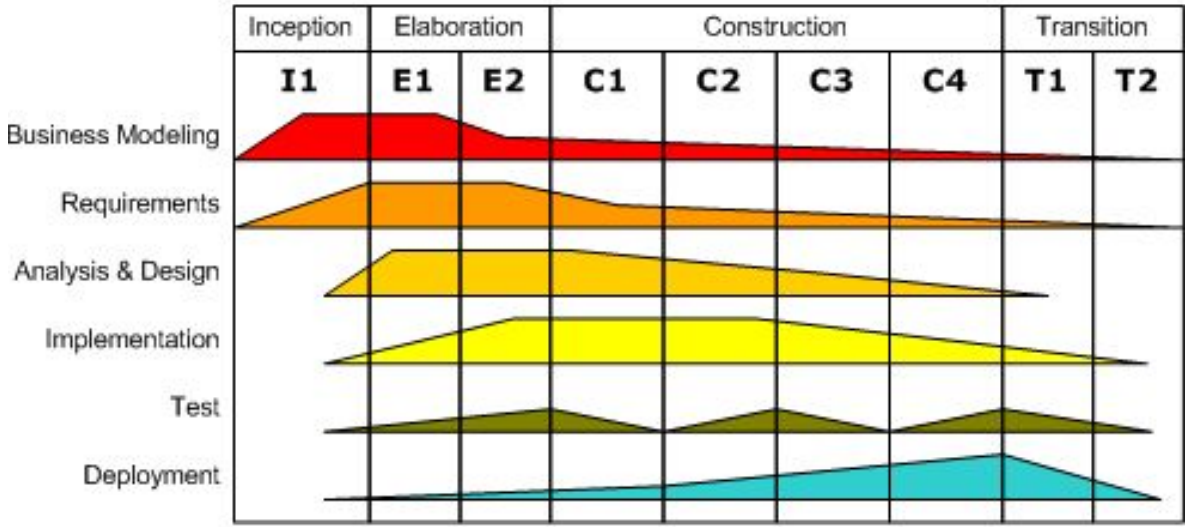
http://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp

Agile Police Project



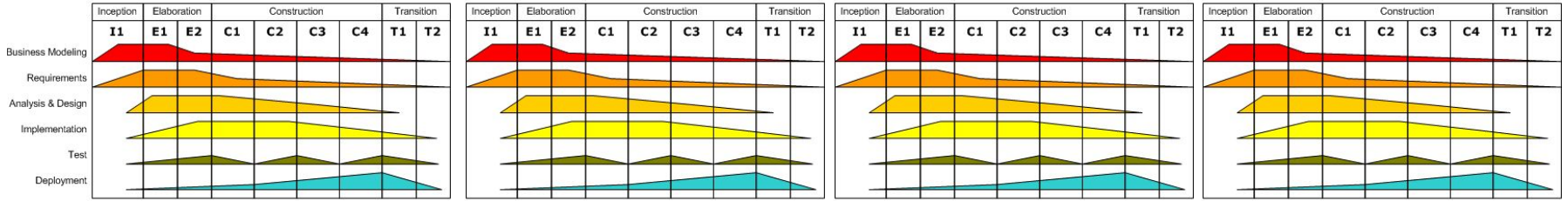
http://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp

Even Within an iteration activities are overlapping

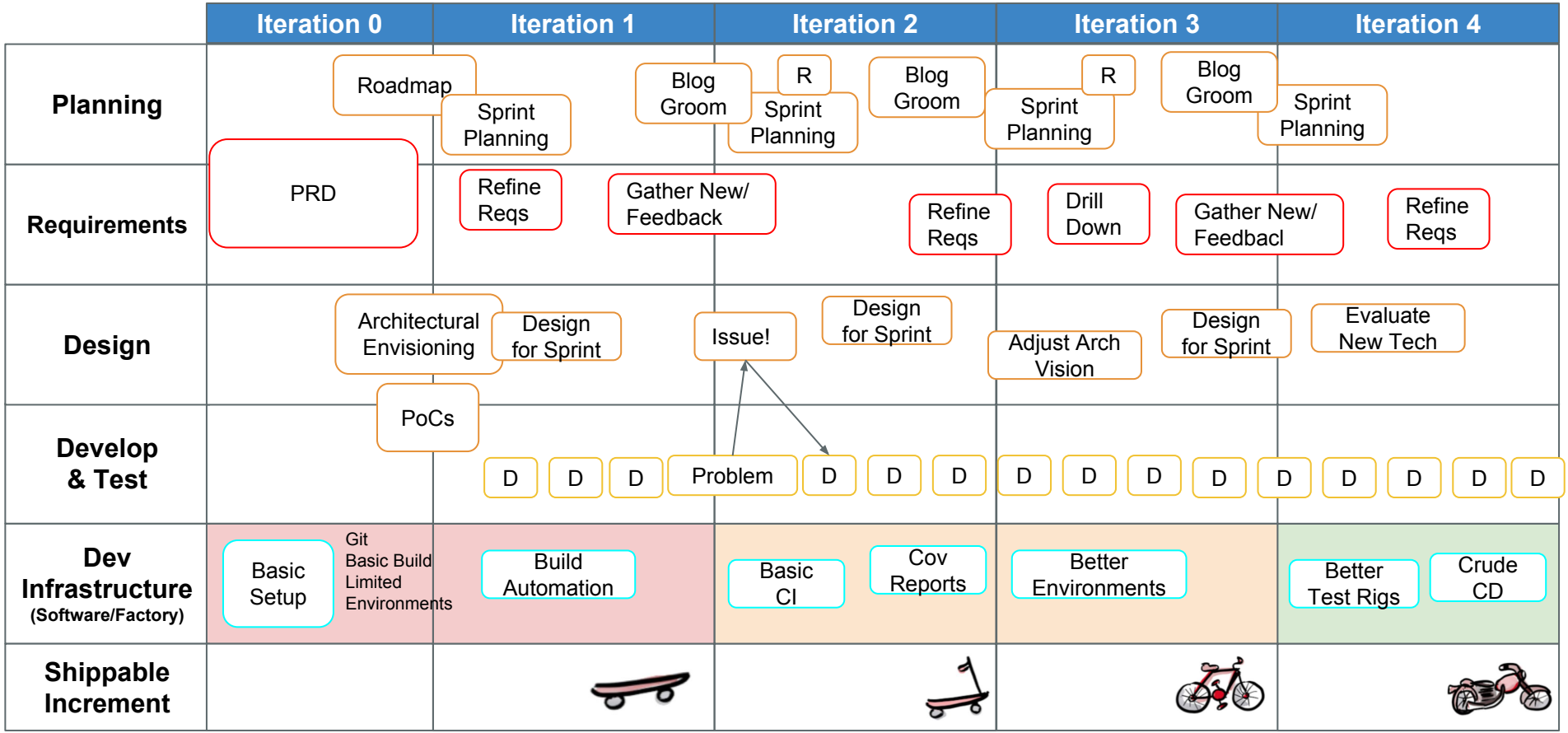


(Diagram from RUP, yet another agile method)

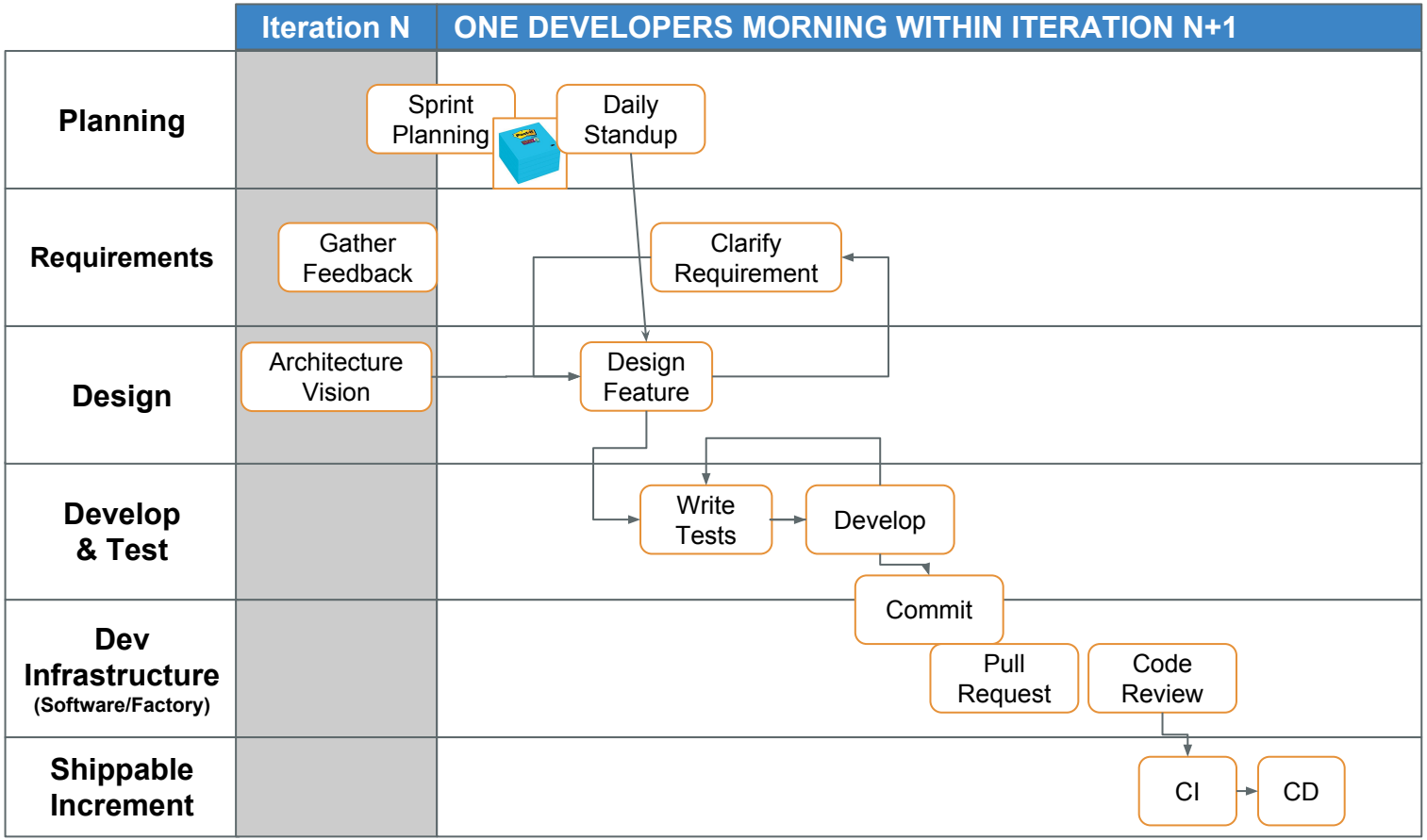
Repeated Across Iterations



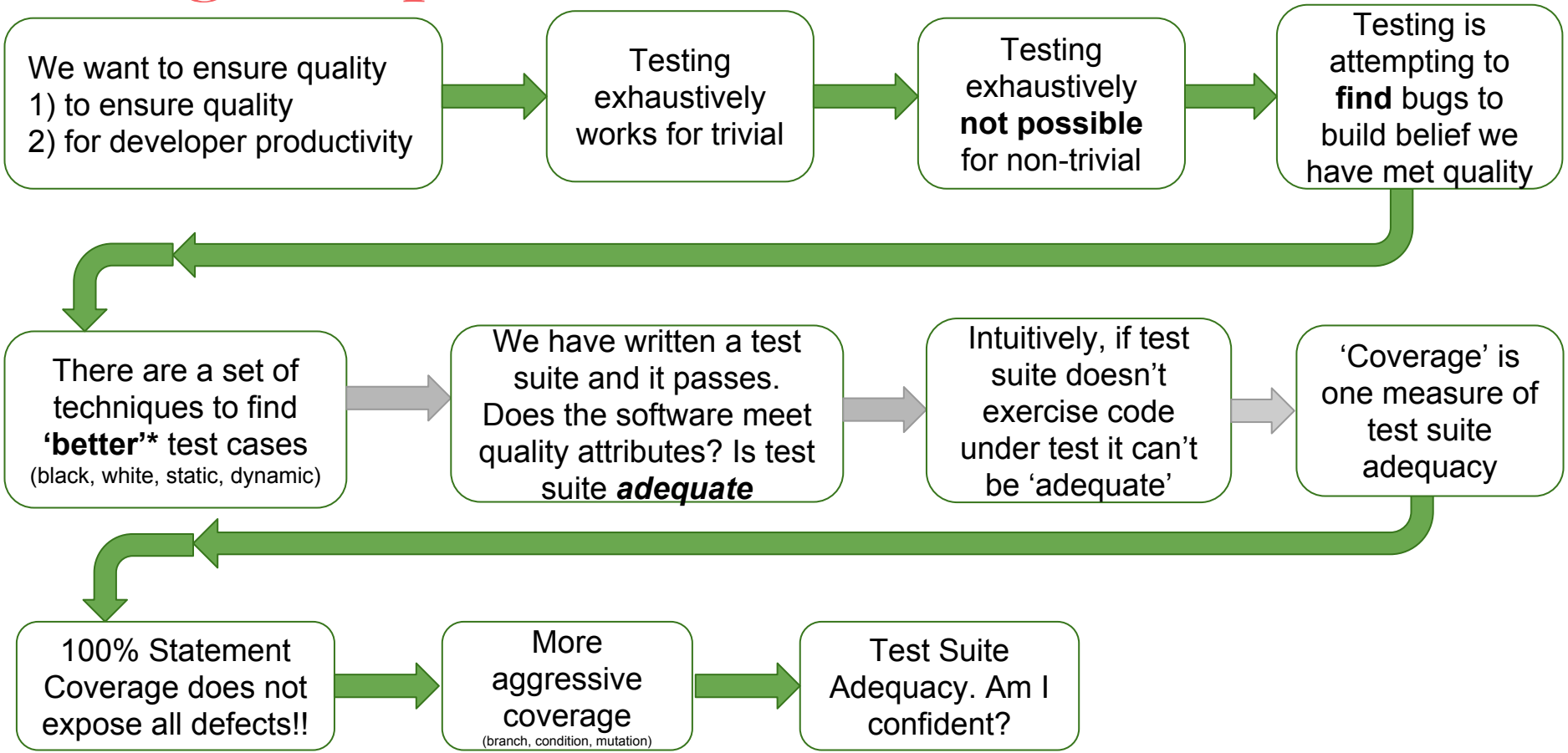
‘Planes of Activity’



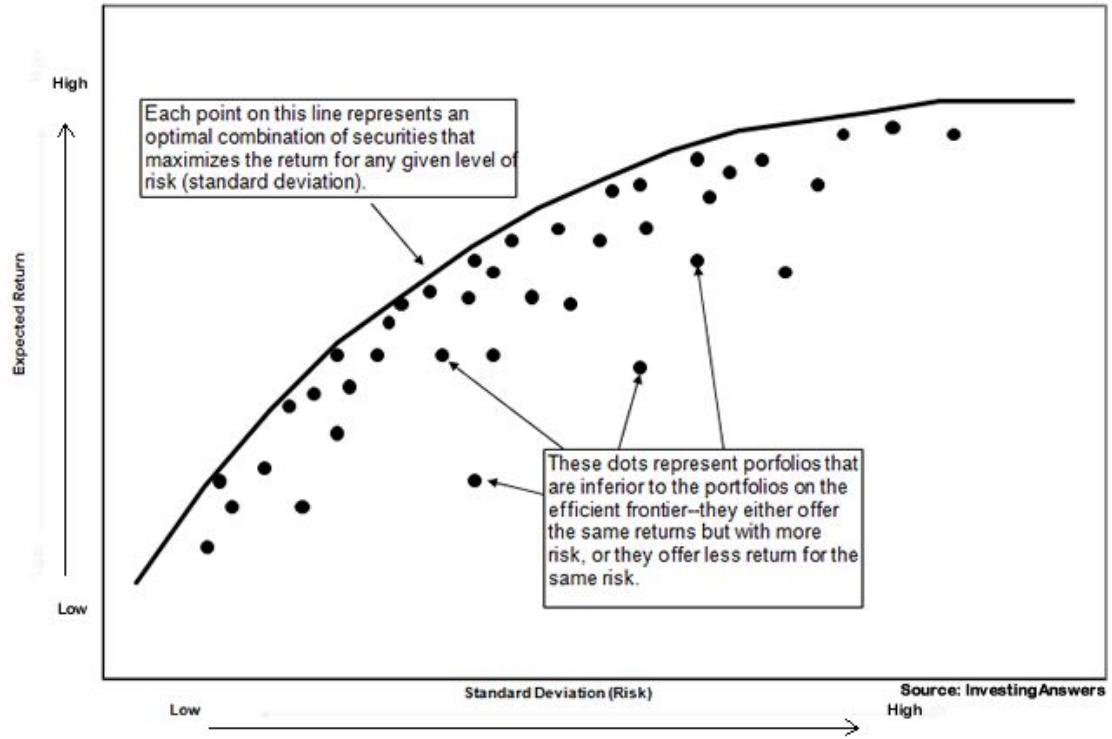
Sprint to Production



Testing Recap

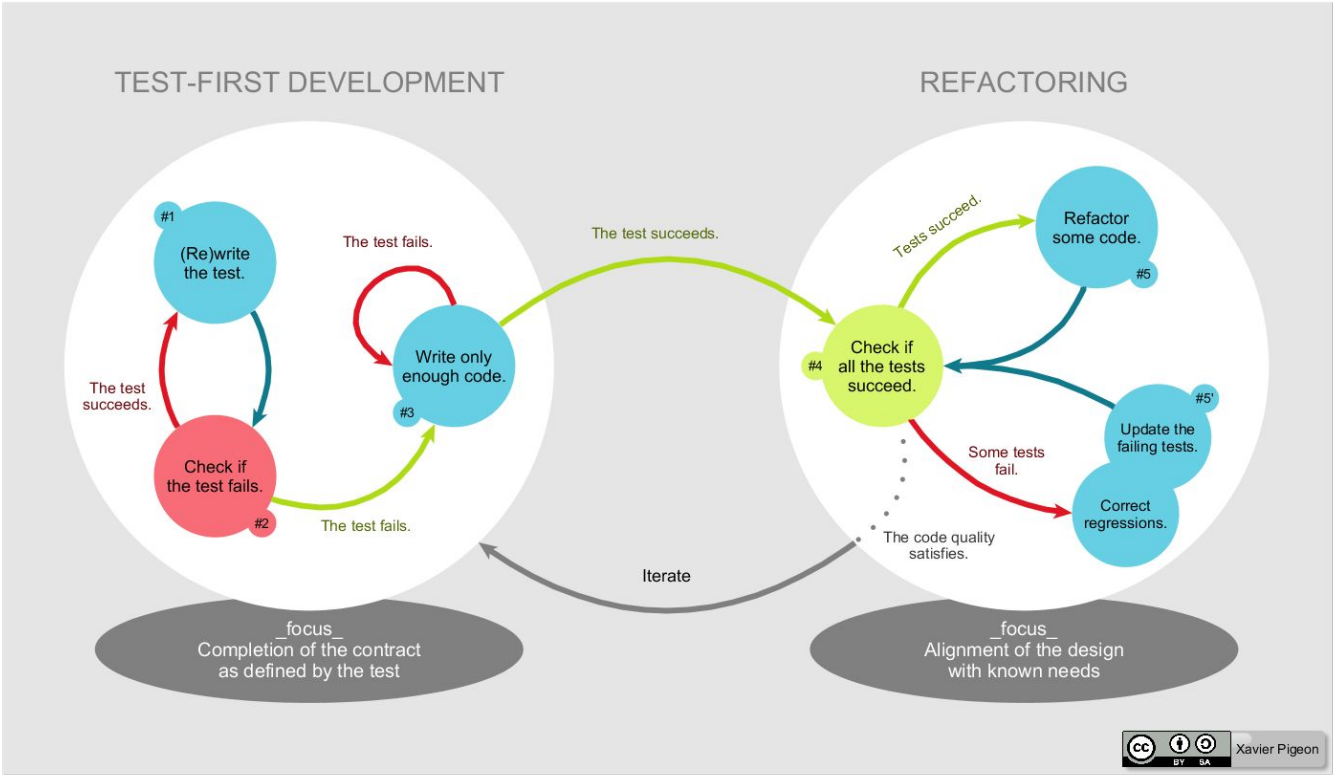


Efficient Frontier - Test Cases?



http://www.investinganswers.com/financial-dictionary/investing/efficient-frontier-1010

Test Driven Development



Testing Techniques

		Am I allowed to execute the code?	
		Static	Dynamic
Box / Perspective	Black	• Specification walk-through	• Equivalence Partitioning • Boundary Value Analysis
	White	• Code standards • Code reviews • Compilers	• Flow Graph • Coverage (statement, branch, condition) • Mutation Testing

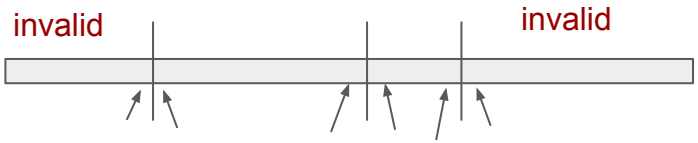
We will dive into dynamic black and white. Remember the context/framing

Technique: Black Box EP & BVA

Equivalence Partitioning

- Domain can not be exhaustively explored
- However, behavior is often the same for ranges within the input domain
- *Partition* input domain into ranges of value with similar behavior to reduce number of test cases

Boundary Value Analysis



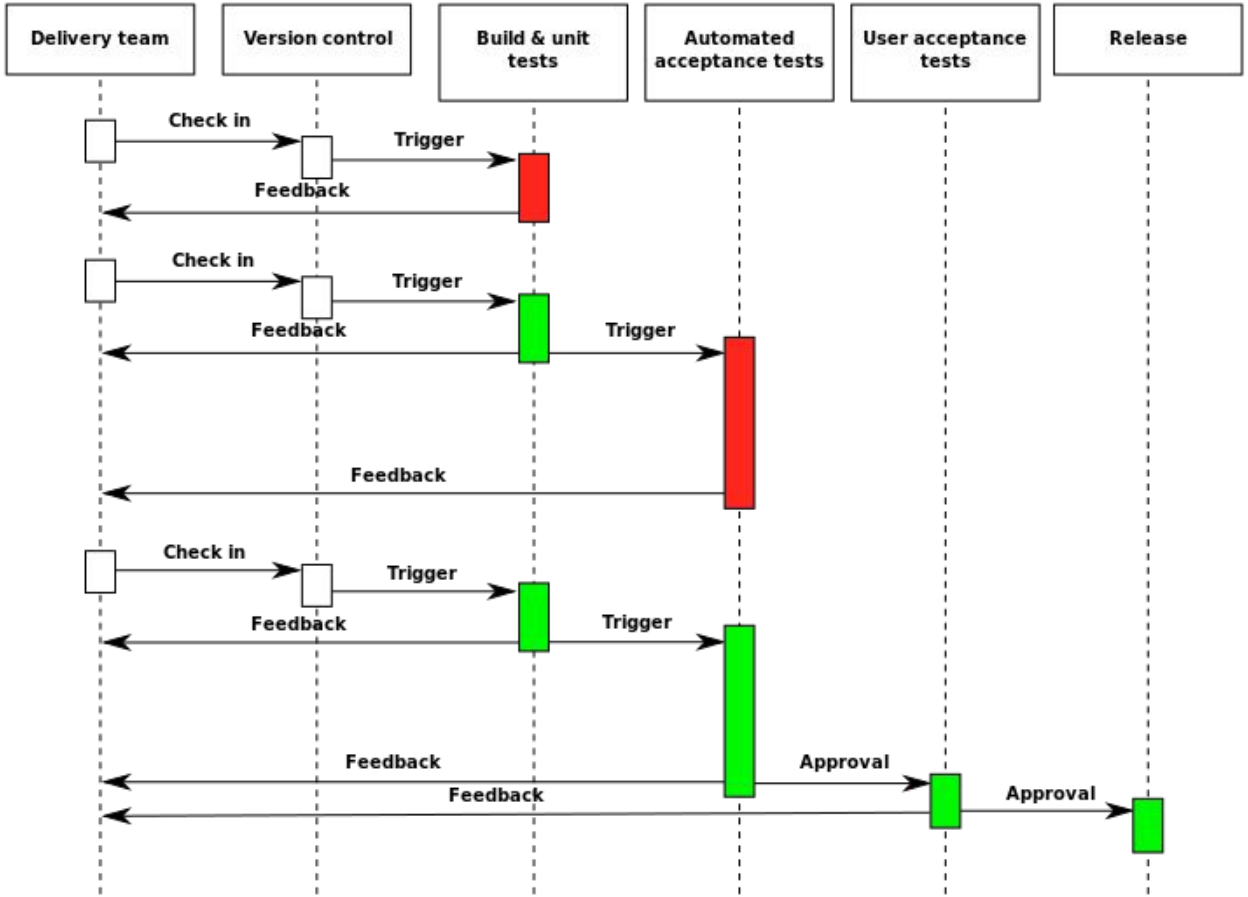
- Amplifying technique is BVA
- Write test cases on the boundaries of equivalence partitions
- Exposes logical and coding errors (classic example is < vs <=)

Are my tests adequate?

Rigour defined by higher % coverage and more aggressive measures (statement, branch, condition)

		Test Suite Adequacy	
		Rigorous **'Good Coverage'**	Lousy **'Poor Coverage'**
Test Suite	Passes	**Higher confidence in quality Lower defects likely**	**Lower confidence in quality. Higher defects likely**
	Fails	At least one bug	

Do not be confident in your code if your tests pass unless they are rigorous



https://en.wikipedia.org/wiki/Continuous_delivery