

Welcome from Ian Billingsley

'Feedback during the Software Development Cycle'




Welcome from Ian Billingsley

- Over 16 years developing LabVIEW applications
- Aerospace, Formula One, Automotive, Subsea, Government.
- Certified LabVIEW Developer 2006-2013
- Certified LabVIEW Architect since 2013



Presentation Overview

- Traditional software development models
 - The cost of feedback delay
 - Reducing feedback cycle time
 - Practical applications in LabVIEW
 - Conclusions
- 

A traditional software model

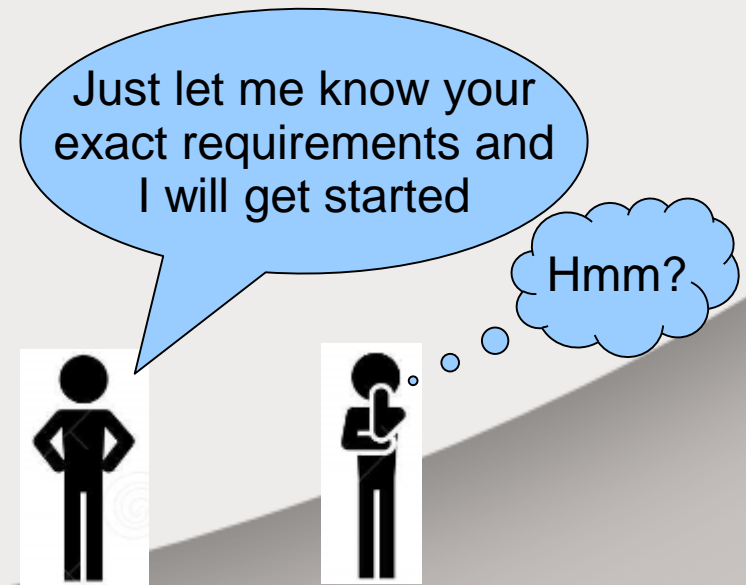
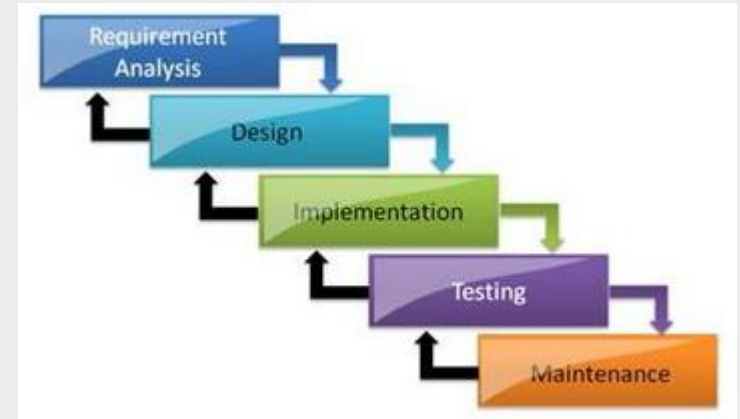
The Waterfall Software model

Useful when:

- Requirements are clear
- Requirements do not change
- Customer is happy to wait

Problems:

- Long lead times
- Requirements evolution
- Waste
- Slow feedback cycle time



Feedback is awesome!

Computers are great because when you're working with them you get immediate results that let you know if your program works. It's feedback you don't get from many other things.

Bill Gates

meetville.com

I am still learning.

- Michelangelo
at age 87

KODACOVER.COM

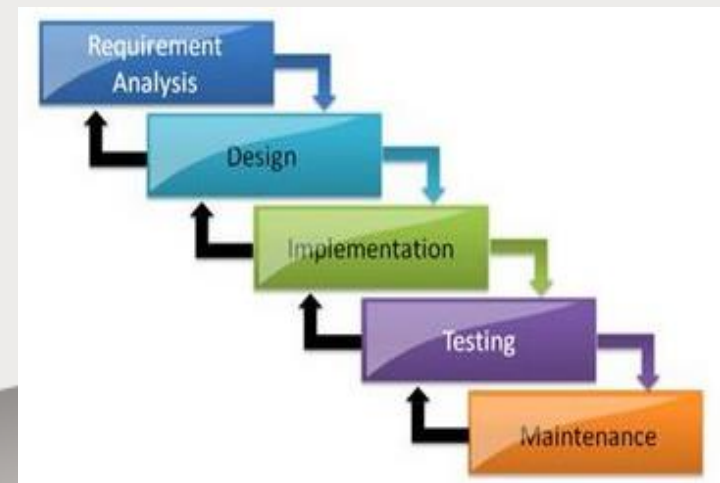
Practice isn't the thing you do once you're good. It's the thing you do that makes you good!

Malcolm Gladwell

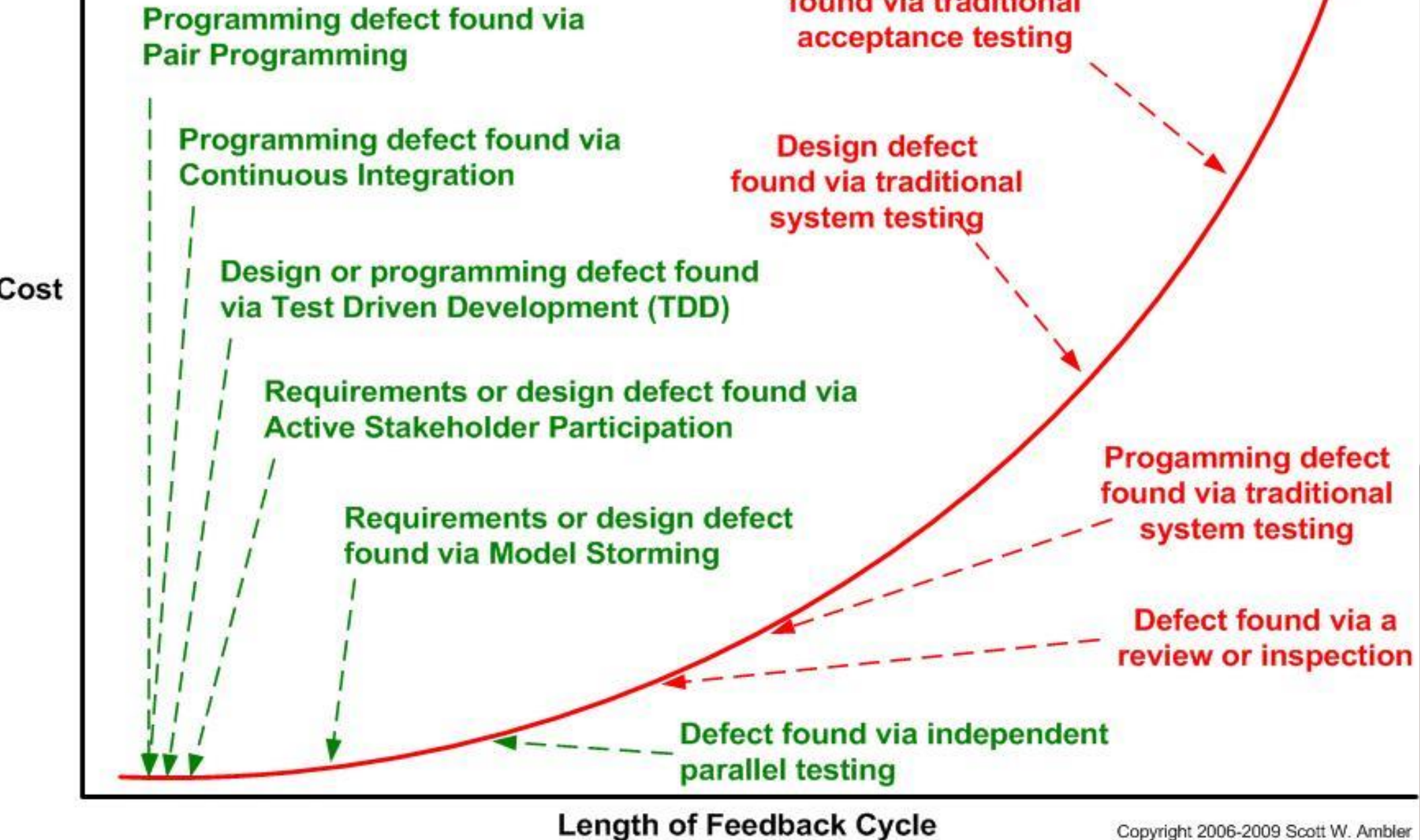


The benefits of faster feedback

- Re-enforces learning
- Reduces project risk
- Requirements can evolve with reduced impact

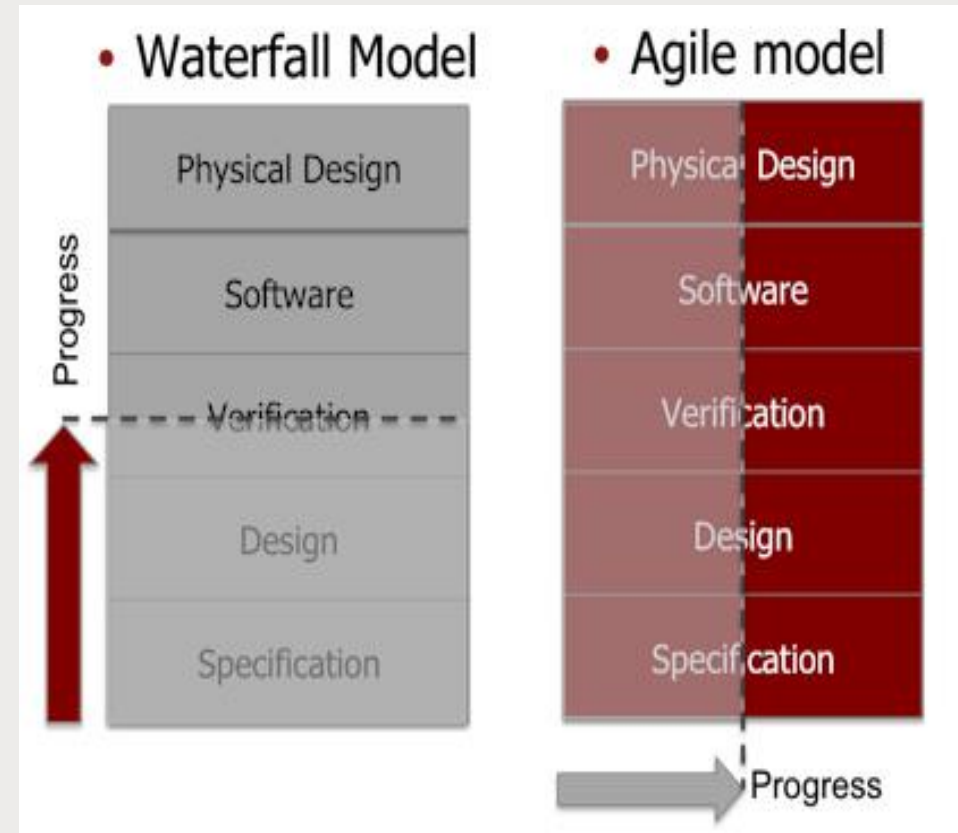


Exponential cost of feedback delay

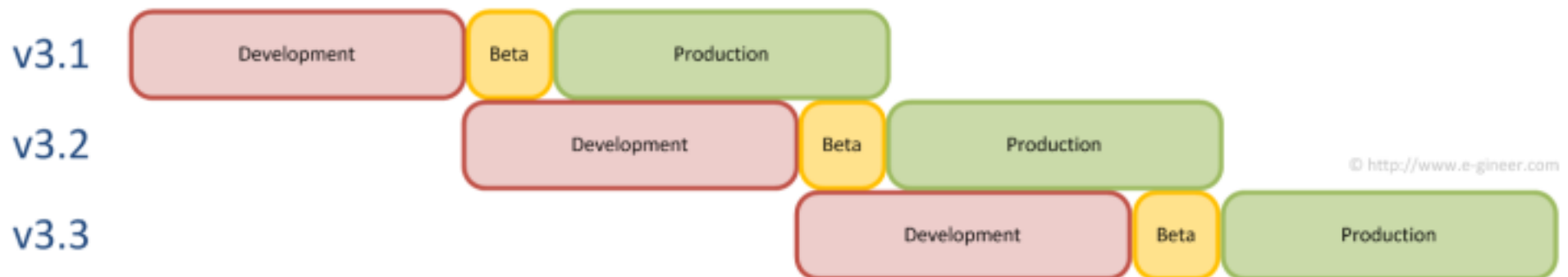


Reducing feedback cycle time

- *Divide tasks horizontally not vertically*
- *Build technology islands*
- *Frequent releases*
- *Requirements lead development*



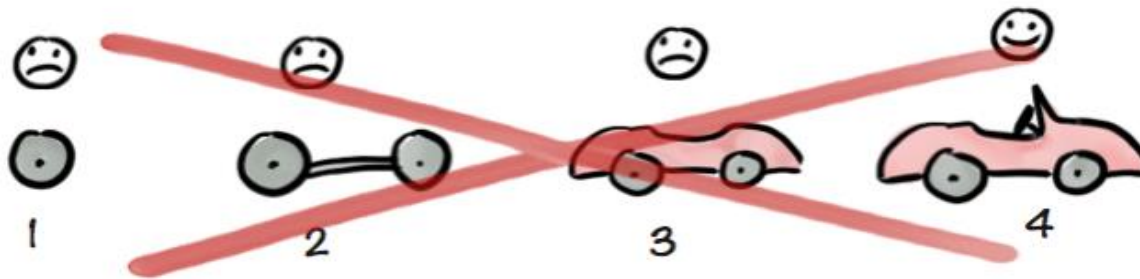
Continuous Application Release Cycle



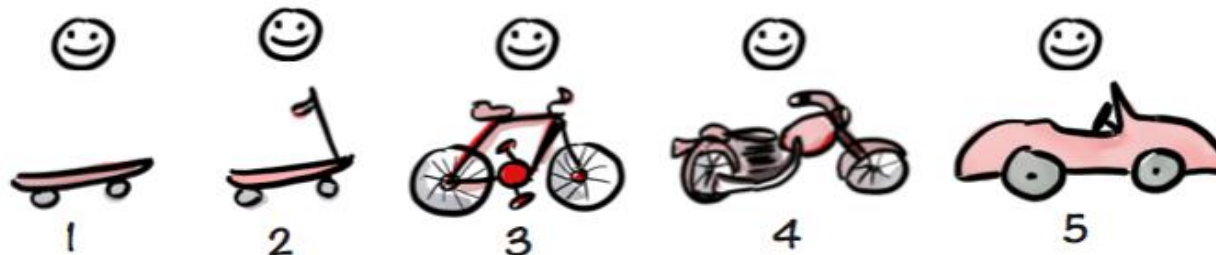
Dividing the tasks example

- Objective – Build transport from A to B
- Release usable application frequently
- Make use of the feedback opportunity
- Iterate until customer is satisfied

Not like this....



Like this!

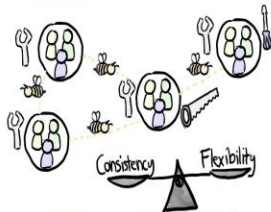


Spotify Engineering Culture

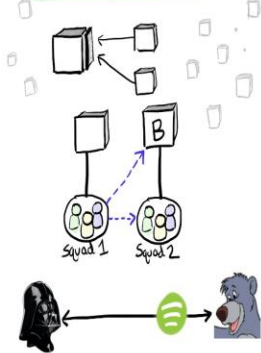
Part 1 of 2

Hank Kniberg
Jan 2014

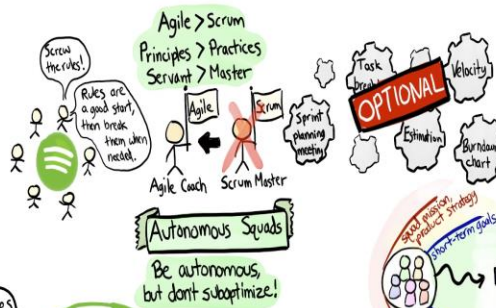
Cross-pollination > Standardization



Internal Open-source model

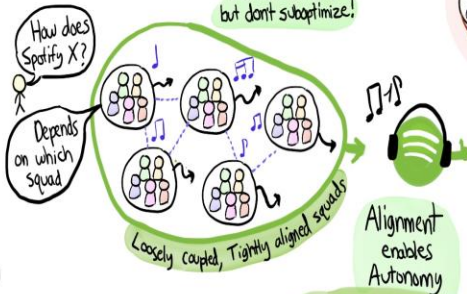


Small + frequent releases

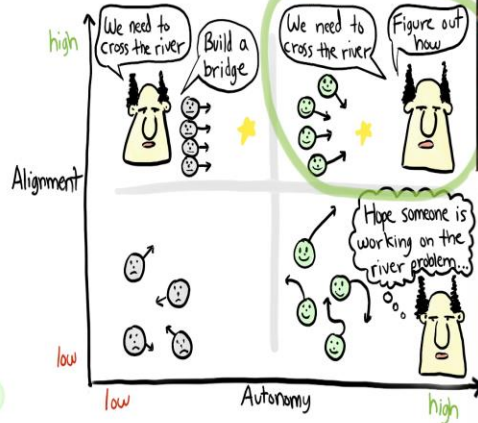


Autonomous Squads

Be autonomous, but don't suboptimize!



Alignment enables Autonomy



Leader's job: Communicate what problem needs to be solved. And why.



Squads' job: Collaborate with each other to find the best solution.



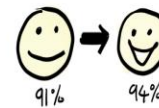
If you need to know exactly who is making decisions, you are in the wrong place.

Focus on Motivation

Hi everyone,
Our employee satisfaction survey says
91% enjoy working here,
and 4% don't.

This is of course not satisfactory,
and we want to fix it.

If you're one of those unhappy 4%,
please contact us.
We're here for your sake, and nothing else.



Trust > Control

Agile at scale requires Trust at scale

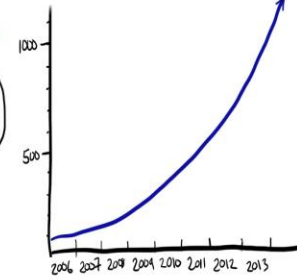
1200+ employees
30+ countries

~~Politics~~ ~~Fear~~

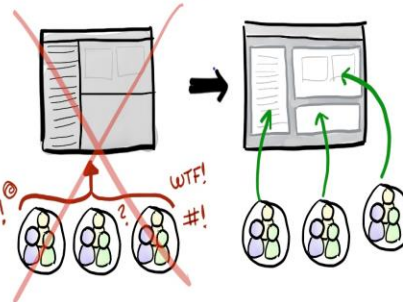
People > *

My colleagues are awesome!

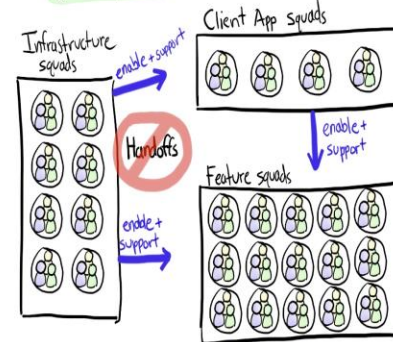
~~Ego~~



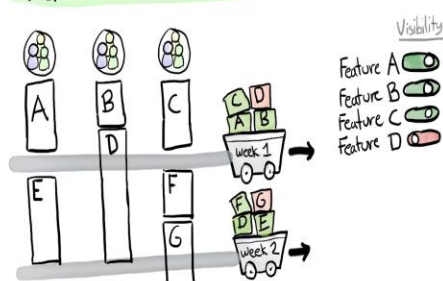
Decoupled releases



Self-service model
enable > serve



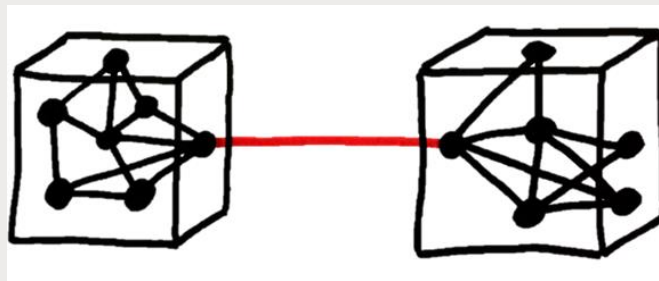
Release Trains + Feature Toggles



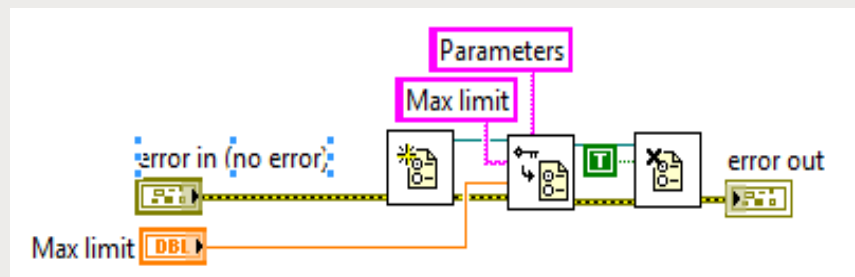
Practical tips for LabVIEW development

Design application to facilitate change

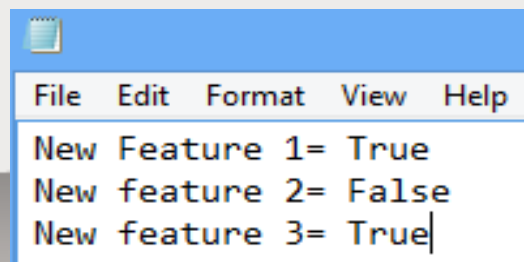
- Low coupling & High cohesion modules



- Abstract variables into a configuration file



- “Feature Toggle” via configuration



Practical tips for LabVIEW development

Maximise opportunities for feedback

- Ensure errors are logged (eyes and ears of the application)
- Start with an engineering / diagnostics interface
- Commit to building an executable at each release
- Formalise feedback at each release
 - Bug sheet, Discussions, tune requirements



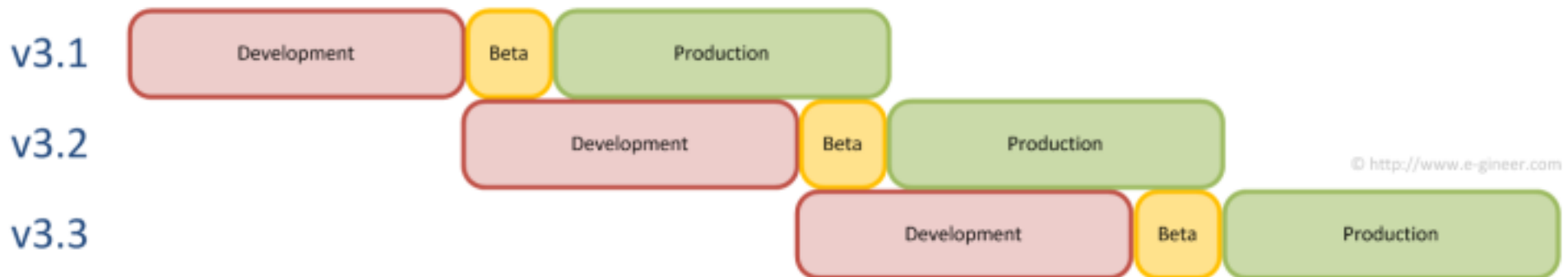
Practical tips for LabVIEW development

Keep control!

- Commit to fixing bugs at each release not to building new features.
 - this leads to code and fix
 - Can appear unprofessional
- New features should be introduced in the next version
- Increment version number each build



Continuous Application Release Cycle



Summary

- Traditional software models are often not optimised for software development.
- Fast Feedback re-enforces learning and reduces project risk
- Feedback cycle time can be reduced by dividing tasks horizontally not vertically
- Develop LabVIEW applications that facilitate change and maximise feedback

Further information

- Examining the Agile cost of change
 - <http://www.agilemodeling.com/essays/costOfChange.htm>
- Spotify Labs - labs.spotify.com

Thank you for listening

Questions?

