

Lean Software Development

Doing what works - not what *should* work

Intro

- Who is James Hall
- What is Nulka
- What is my point?
 - Defence and Aerospace do the most rigorous conventional development
 - There are better ways than conventional development
 - Make the case for investigating Lean Development in Software because it makes sense
 - Wastes
 - Countermeasures

Lean Product and Process Development, Ward, Allen C

Nulka - a crucible of innovation

- 2 years from clean sheet of paper to flight trials
- Original development offspring:
 - Fire Control & Launch Subsystem (RAN and CN),
 - ESSM workshare (Guidance & Control),
 - AASD division of Aerospace (UAV & UGV)
 - Ship Air Defence Model (SADM)
- All these development successes had at their heart Lean principles

Lean & our history

- We didn't set out to be Lean
- But Lean fits what was done
 - But not all of what we do!

- Now investigating how to formalise the way we succeeded

Core Principles

- Goals of Lean
 - Best Quality as perceived by the End User
 - Lowest System Level Cost
 - Shortest Lead time

So What?

- Eliminating Waste gets you to those goals
- But what is waste in a development environment?

Knowledge Wastes

- Scatter
 - Barriers to Communication
 - Poor Tools
- Hand-Off
 - Useless Information
 - Waiting
- Wishful Thinking
 - Testing to Specification
 - Discarded Knowledge

Countermeasures

- Value Focus
- Entrepreneur System Designer
- Set-Based Concurrent Engineering
- Teams of Responsible Experts
- Pull, Flow, Cadence/Heartbeat

Scatter

Where does all the time go?

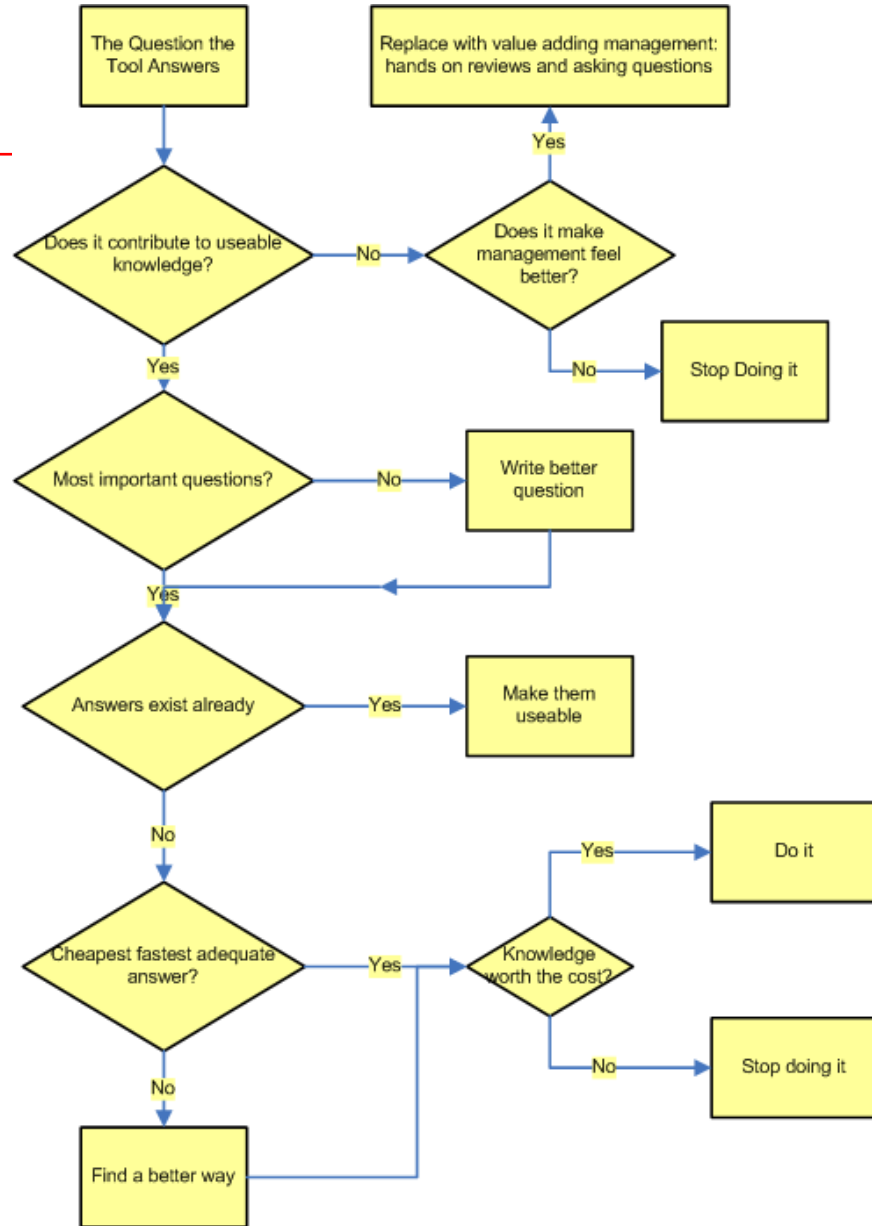
	Problem	Conventional Response	Scatter Effect	Lean Response
1	Things going badly	Reorganize	Obsoletes interaction knowledge	Find Root cause
2	Project Slipping	Add more developers	Disrupts communication	Supervisors help
3	Faults in development tools	Call the vendors more often	Distracts developer – Makes excuses for slow progress	Find limitations and fix
4	Keep having product failures (inc repeats)	Add more checks and gates to the development process	Distracts Developers	Find root cause and fix
5	The customer wants something new	Tack onto current development project	Over loads developers, causes mistakes & inconsistency	Scheduled continuous innovation
6	Problems with software support / installs	Keep developers on support until system is running properly	Software developers not available for the next development, problem repeats	Set Based concurrent Engineering, planned rotation of people from product support to development

Barriers to Communication

- Social Barriers
 - Informal rules of engagement
 - Formal rules of engagement
 - Waste expertise of shop floor
 - Good Engineers moved to Management
- Physical Barriers
 - Distance
 - Time zones
 - Language
- Technology Barriers
 - Incompatible Formats
 - Proprietary formats
- Skill Barriers
 - How to put what you want into an understandable useable format

Poor Tools

- Does using a tool add to useable knowledge?
- (Does it make management feel better)
- Is the tool asking the most important question
- Is there an answer already?
- What is the cheapest fastest adequate answer?
- Is getting that knowledge worth the cost?



Pg 39 Reference A.

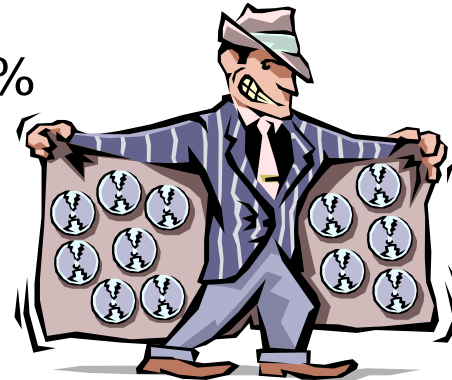
Hand Off – The mother of all death spirals

- Who has responsibility? Who has Knowledge? Who does the work? Who manages feedback?
- It's destructive because you have people making decisions without enough knowledge
- A good teacher can only teach 30% of the intended knowledge

Knowledge



Responsibility



Feedback



Action

Useless Information

- (Most) Power Point Presentations!
- Status Reports
- Proforma analysis
- Adhoc demands for information re problems (seagull management, tail wagging dog, inverted outcome priorities)
- Checklist reports (what are you going to do with the result?)
- Un-needed Optimization
- New Technology that is not better than the old!
- Normally to maintain the illusion of control

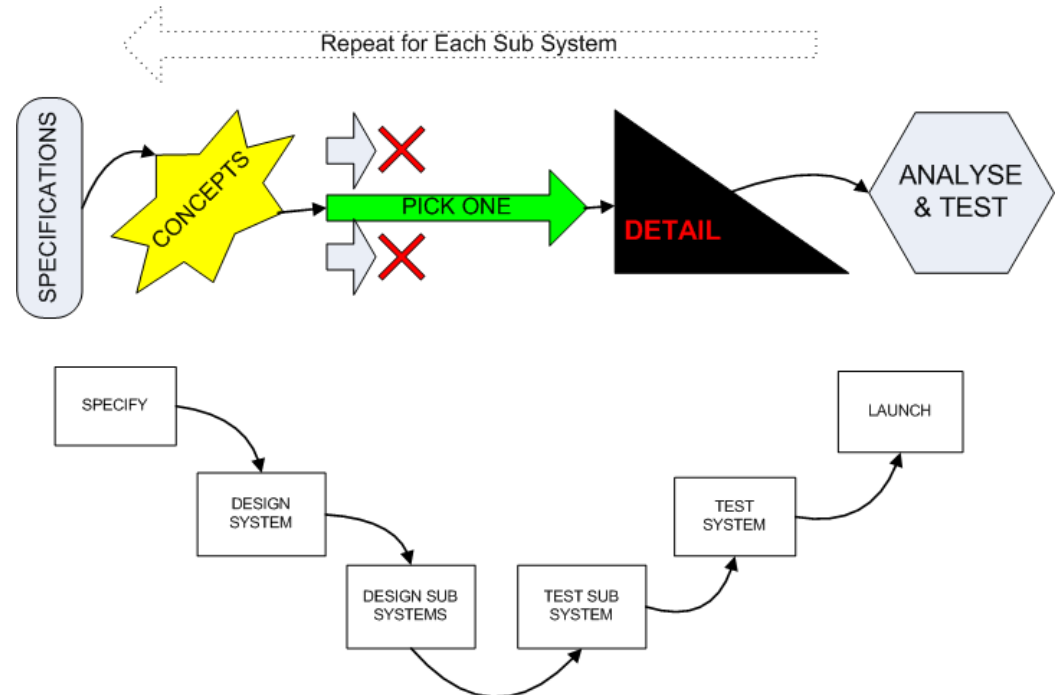


Waiting

- Developers waiting for approvals
- Sign off Turn Around Time
- Sequenced work, PERT, Gantt, Phase Gates
 - Primary cause of delay: this has to finish before we start that
 - Spec -> System Design -> Module Design -> Integration
 - You can't Schedule Development Tasks like pouring concrete
- This isn't construction, you can start developing different modules at the same time (Different dependencies)
- Noticed that when Developers are allowed to break the rules (eg not do all the extraneous tasks) they go so much faster.

Wishful Thinking

- Making Decisions without Data Operating Blindly
- Bid processes
- Fixing final specifications at the start
- Specs are set on the basis of what was possible, not what will be
 - Do you really know at the start what your Memory and Processor usage rates are going to be?
 - How do you know you aren't going to push your GUI to it's limits but the back end guys get a free ride? If one slips the whole program slips.



Testing to Specification

- When you test to spec, you do not learn about how far you exceed a spec by
 - Bottlenecks and weak link remain hidden
 - Candidates for future upgrades for capability improvements remain hidden
 - You do not learn what will break the system
- Bill Gates and The Blue Screen of Death
- It might be able to run with more users than specified (Free upgrade??)
- Find which parts of the Program break – any improvements can then target those issues

Discarded Knowledge

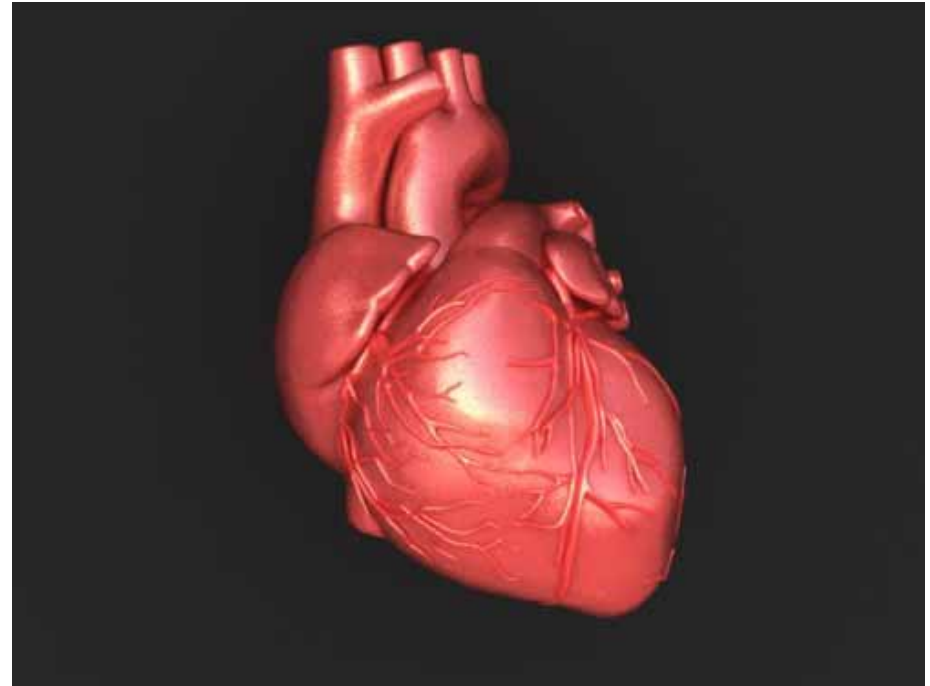
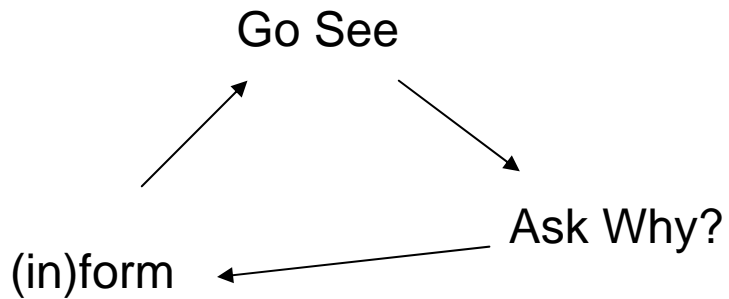
- What happens after the development program finishes and the Software is delivered?
- What about all the decisions that were made along the way and why?
- What if you took everything that was learned and put it into a USEABLE form?
 - We keep our people around forever so we don't lose their learning in the meantime!

Countermeasures

Requires cultural change to be sustained

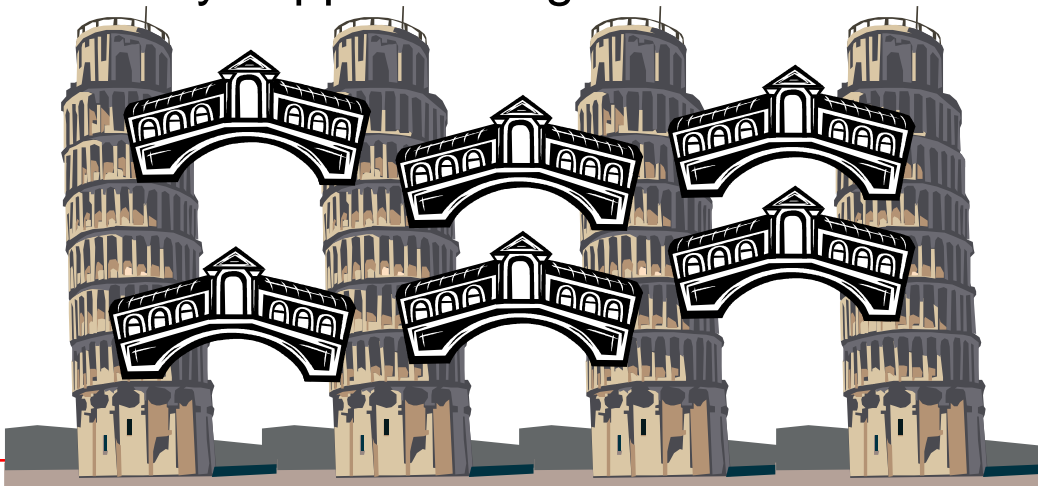
Value Focus

- Ask
- Why?,
- What's the point?,
- Does this add value/knowledge?
- Who is this for?



Entrepreneur System Designer

- It's his/her system
- Has the vision which functions support
- Ultimate decision responsibility
- Works across functions
- Faster Decisions
- The central person to make sure things get done
- Bureaucracy supports the goals of the ESD



Set Based Concurrent Engineering

- Start Everything at the Start
 - Start from a Set of Solutions rather than point Based single solution which you have to make work.
 - Identify when down selections have to be made
 - You know then where trade offs have to be made
 - Low risk of failing to find a solution, increased likelihood of providing and innovative solution
- E.G. GUI and Back End requirements, if the back-end requires less processor or memory than originally thought, that can be released to the GUI developer which may enable them to use better graphics
- E.G. When starting a new module, the SADM team will sit down and begin prototyping at the start multiple solutions, the most innovative and likely to succeed gets down selected.

Flow, Cadence, Pull

- Flow: knowledge and material are available when needed
- Pull: Responding to the needs of the next downstream customer
- Heartbeat / Cadence: Repetitive rhythm of ongoing development levels the load on resources
- The Effect
 - Value Creating Management: Supervisors directly create value designing systems and spreading knowledge supporting front line developers

Teams of Responsible Experts

- Lets think about going for a ride in an Unmanned Aerial Vehicle
- Behaviours
 - Responsibility
 - Teamwork
 - Expertise
- Discipline
 - External
 - Internal

Summary

- Go See -> Ask Why -> in Form ->Go See
- The developers goal is to get Learning and Knowledge as fast and as cheap as possible
- Lean Countermeasures will help you to overcome the limitations and frustrations of Conventional Development
- More Checks and Gates wont get you a better product
- Use or design your own tools that help you achieve generating value

- You will
 - create pull, vision and leadership
 - understand customer needs, have effective communication & governance
 - be agile

Questions

- james.e.hall@baesystems.com

Reference

Lean Product and Process Development

Ward, Allen C.

2007 The Lean Enterprise Institute, Inc

ISBN 978-1-934-109-13-7