inspearit

Aligning for Customer Value

Pitfalls of Agile transformations

Speaker: Alan Graham

Expert Consultant

Self introduction

Alan Graham

- I am from Ireland and have been working in software development for 29 years…. Long time ☺
- Programmer analyst (Wang) from 1986, then BPR and project management
- Moved to Telecoms (Ericsson networks) in 1997, SW Project and Development management
- China in 2008, PMO Director in Shanghai (Ericsson)
- Agile coach and consultant in Nanjing (Ericsson)
- Operational development in Nokia phones (Beijing).
- I am a passionate advocate for Agile and Lean ways in software development and the alignment of IT with business objectives

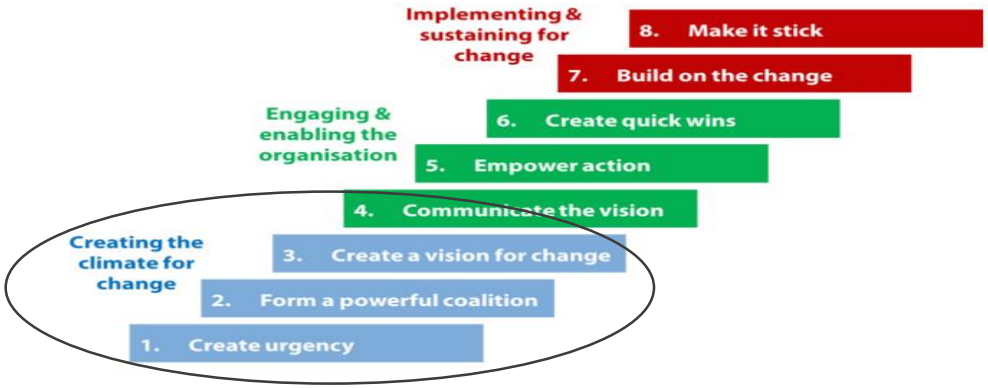
Pitfalls of Agile transformations

A useful Change model - Kotter

Agile transformation is normally a big change

Will give us some clues on how to fail
and then
we will see what to do to be successful

Failure of Executive Leadership



To set the vision, create urgency, build the guiding coalition requires Top Executive leadership and understanding of the journey

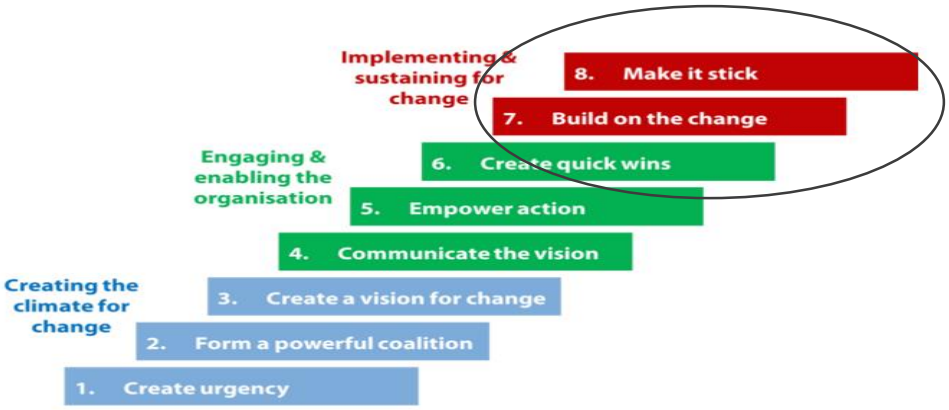
Executive Leader unaware that transformation is a long, challenging journey

- Demands immediate results
- Keeps existing metrics to measure success
- Underestimates the impact of the culture change
  - Underestimates change to Management culture
  - Doesn't realise that there will be major organisational impacts
- Fails to appoint and support a suitable transformation team

→

Attract key change leaders by showing enthusiasm and commitment
Model the trust and teamwork needed in the group
Structure meeting formats that increase trust

Lacking a Transformation Delivery Manager



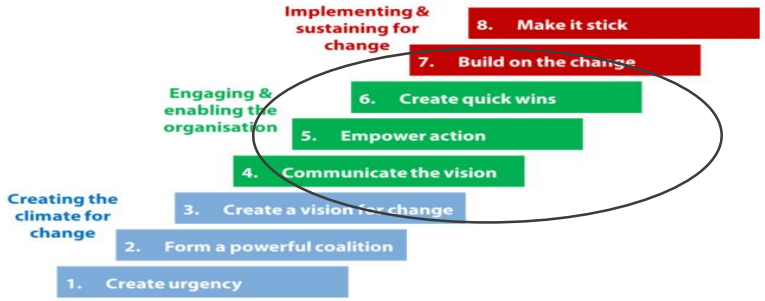
The person who will work to deliver the transformed organisation

Failing to realise that the transformed company is an end-product and must be managed as if it was a product to be developed

Only focuses only on processes, practices, structures and tools

Ensures clear transformation goals in all teams and across teams
Focus on behaviours
Inspect and adapt
Eliminate blaming
Encourage open-ness
Trust
Remove "us" and "them" type of thinking

Failure of Management to change



Transformation will mean changes in management style and behaviour

Action oriented, **command and control** Managers fail to change their behaviour from control to servant leadership

Command and Control

Micro Manager

Risk averse

Allergic to agile values

Drives performance through individual stretch goals

Believes in functional silos

Mentor

Empathy

Invites feedback

Communicator

Team Builder

Conflict negotiator

Adaptable

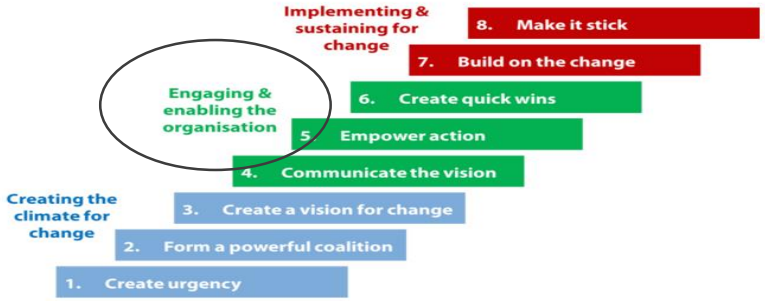
Greater good

Creative

Humility

Integrity

Failure to change organisation structure



Transformation will mean changes in organisation structure to enable End-to-End value delivery

Persisting with the existing control structure (hierarchical, sub optimised, functional organisation) instead of a new collaborative model that supports ultimate value delivery and decentralises control

Silos →

Cross-functional Team-based

Team level rewards and recognition

Removal of functional walls

Measure overall value delivery not individual departments

Business and IT collaboration

Watch the work, not the worker

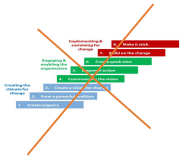
Move decisions closer to the worker

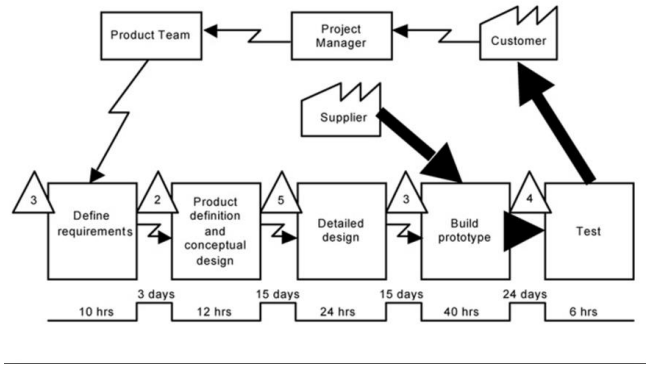
Empower

Enable

Engage

Failure to understand the value stream





Transformation will mean understanding and enabling the End-to-End value stream

Failure to map and understand the End-to-End value stream and to make it lean

Identify queues, bottlenecks, delays

Identify primary constraint, deal with it

Expand the view upstream and downstream

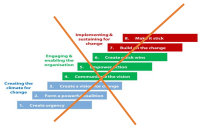
Enhance the value flow

Involve all areas

Establish commitment to the full stream not local parts

Continuous improvement

Persist with Distributed teams



Successful agile transformations do not have widely dispersed teams and team members

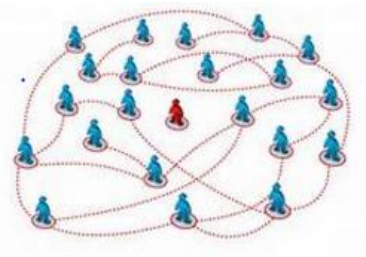
Persisting with teams dispersed across time-zones and with offshore teams in parts of the value stream

Resource utilisation model using teams in different time zones

Reliance on email and documentation handovers

Throw the business requirements over the wall to an offshore development centre

Throw the developed code over the wall to a test group



Eliminate distributed teams

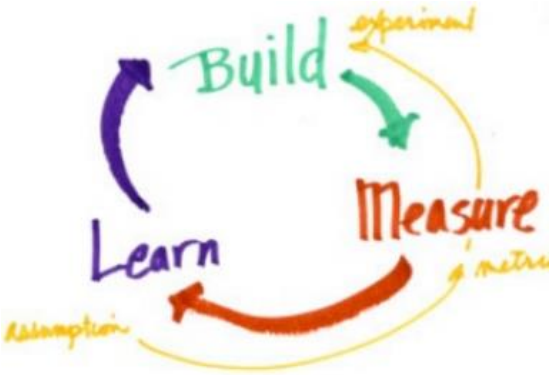
Bring under one roof

Cross functional teams and collaboration

F2F communication

Persist with Late or slow feedback loops

inspearit
Aligning for Customer Value



Fast feedback is the key to both continuous improvement and customer value deliverv

Slow and late feedback inhibits organisational learning, continuous improvement and customer alignment

Stick with traditional project planning methods

Centralised standards

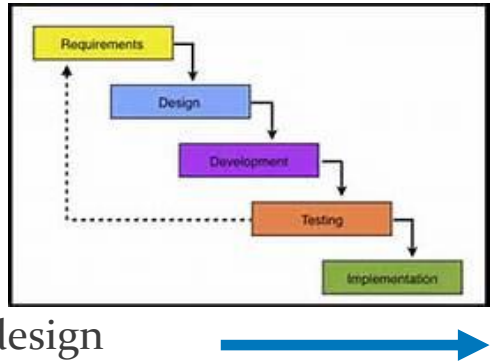
Large batch delivery

Invest heavily in analysis and design phase to reduce risk

End of project lesson learned

Fixed organisation

Improvement programs



Product rather than project approach

Feature delivery

Experimental approach

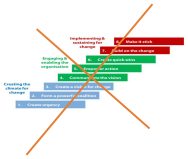
Fast failure

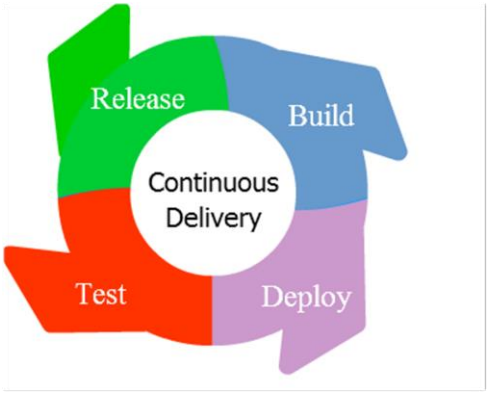
Regular retrospection and improvement

Adaptive behaviours

Learning

Failure to develop a strong technology base





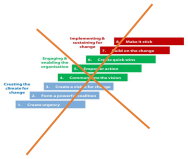
True agile SW development needs continuous delivery technology

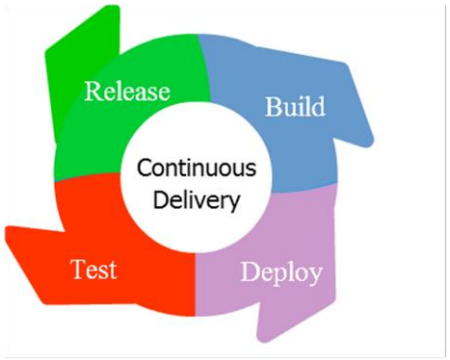
Continuous Delivery is proven to be the safest way for an organization to gain and maintain control of complex software systems

- Acceptance test-driven development
- Automated builds
- Automated testing
- Automated database migration
- Automated deployment

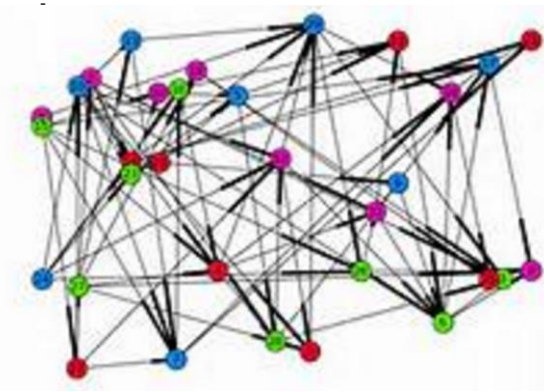
- Code check-in to the mainline at least daily (no branches)
- Mainline is ALWAYS production ready
- Mainline is deployed very frequently

Failure to handle dependencies





Need to decide how to handle SW dependencies in a continuous delivery environme



No matter what kind of system you have, dependencies must be dealt with or else they will eventually haunt you. Technical and architecture people must solve together.

- Big releases
- System testing in the final third of every release cycle
- Automate system test phase
- Focus on defect discovery

→

- Very small releases
- Architecture that isolates dependencies
- Auto-test of complete system of dependent code (using dependency matrix) after every code change
- Focus on defect prevention
- Adopt microservices
- Strangler application for legacy code

And lots more ways to fail …….

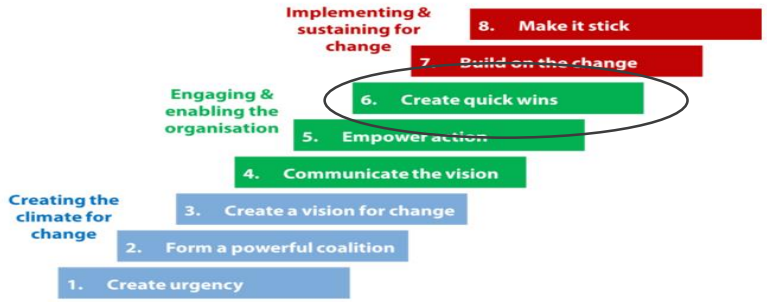
inspearit
Aligning for Customer Value

- Equate self-managing with self-leading and provide no direction to the team. Fail to provide vision and motivation
- Continually fail to deliver what you committed during iteration planning
- Don't create cross-functional teams
- Perpetuate waterfall approach in the sprints or iterations
- Have a Product Owner who does not share product vision with team and stakeholders and who does not evaluate progress on a sprint by sprint basis
- Have one person act as both Scrum master and Product Owner
- Drop or customise important agile practices before doing them "by the book" and fully understanding them
- Don't surface key impediments in retrospectives and don't act on the ones that are surfaced
- Don't continually improve
- Undermine team work and agile values by continuing individual evaluations


Thank you!

www.inspearit.com

Attempting to transform a large complex organisation all at once



Organisation culture, size and product complexity generate big challenges

- Small and deep is preferable to Large and broad

Big bang

Chaos and stress

High dependency, monolithic legacy product not easy to break into deployable parts

Complex technical infrastructure needs to be put in place

Many teams on the same product impacts "self-organisation"

Start small before you scale up

Build organisational change incrementally

Low dependency product architecture is optimal

Multiple independent team preferable to scaling

Greenfield is an advantage