From Waterfall to Agile

Do's and Dont's

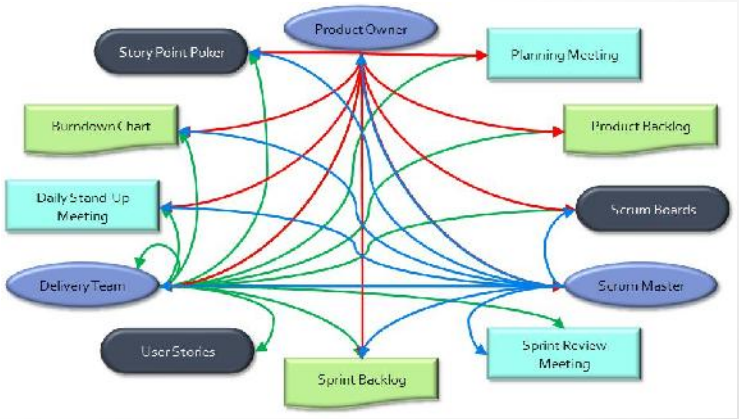
BIO

- Name: **Jorge Monterrosa,** MBA , PMP

- 25+ years in SW dev and Project Management
- UF Graduate (**Go Gators!**)
- Worked across multiple verticals: High Tech, Travel and Tourism, Wireless, Natural Resources, Government, Insurance, Consulting, etc.
- Came into Agile in Aug 2004
- Currently at Carnival Cruise Lines, Manager of Digital Technical Systems (75+ resources), where we launched a Change Management Program in 2010 from Waterfall to Agile (choosing SCRUM)

Today's focus

- What to expect from today's session
- From the top – Waterfall versus Agile
- Readiness - Checklists
- Do's and Dont's
- Continuous Delivery
- Q & A



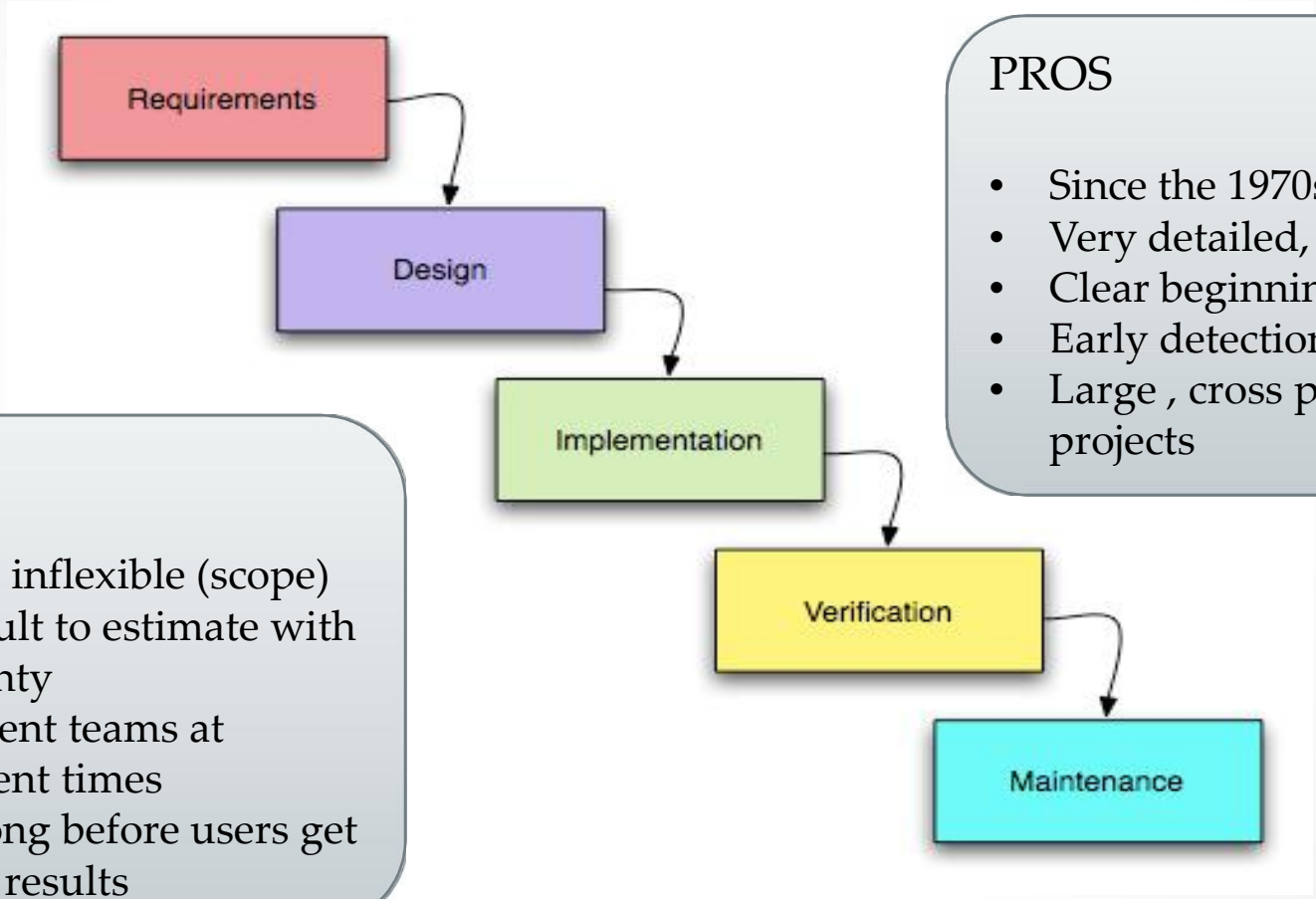
What to expect

- This is not a prescriptive session – no step by step
- I will cover core concepts and dive deeper where appropriate for this class
  - Why are you wanting to change
  - Key dimensions of going to Agile
  - Readiness assessments
  - Planning
  - Execution
  - Measurements
  - Reinforcement
- Open discussion , sharing experiences from others

Waterfall Model

- The **waterfall model** is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance.
- The waterfall development model originates in the manufacturing and construction industries; highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.[1]

Waterfall Model – *Cont'd*



PROS

- Since the 1970s (40+ yrs)
- Very detailed, sequential
- Clear beginning/end
- Early detection of errors
- Large , cross platform projects

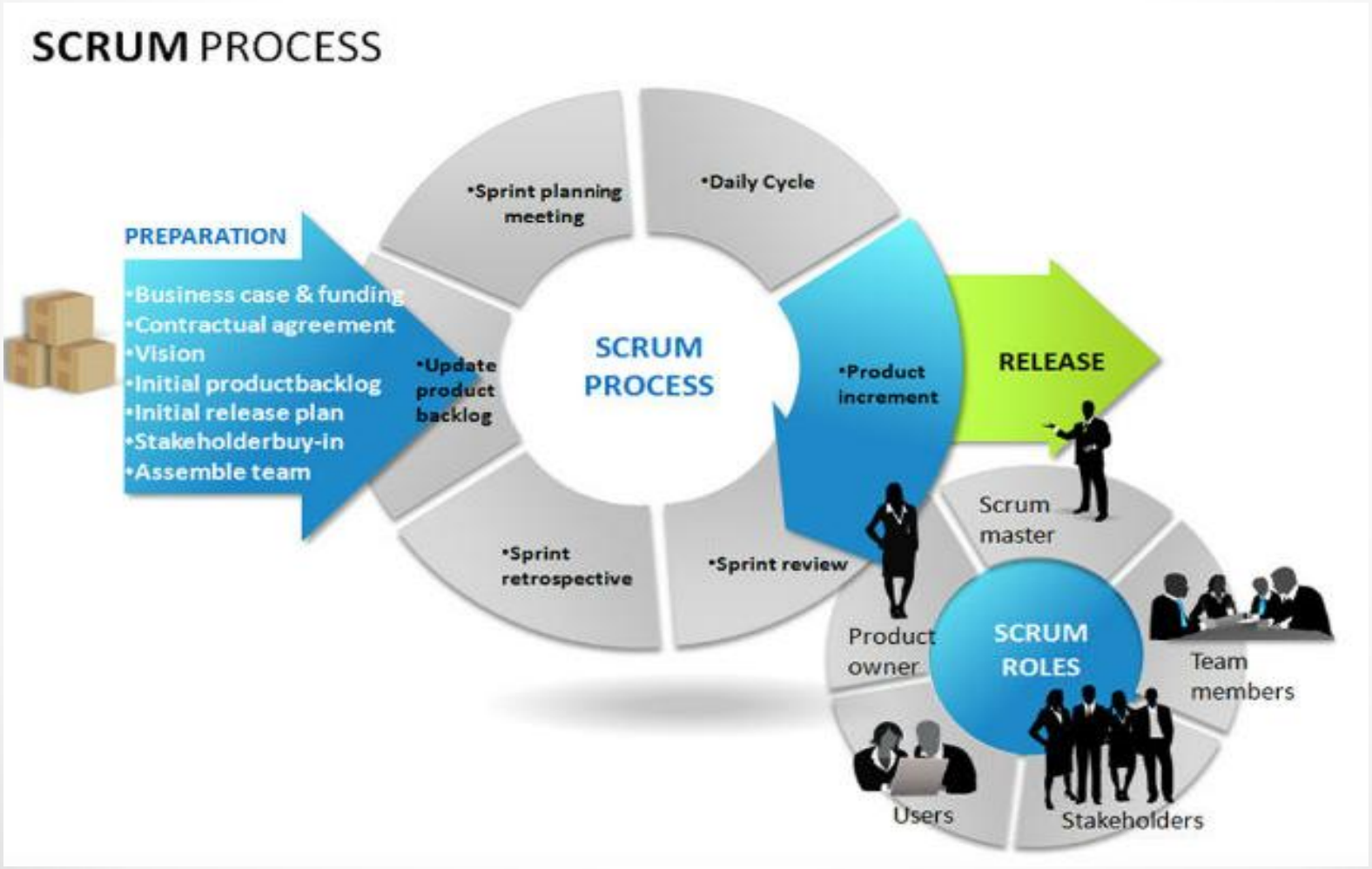
CONS

- Rigid, inflexible (scope)
- Difficult to estimate with certainty
- Different teams at different times
- Too long before users get to see results

Agile Model

- **Agile software development** is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle. The *Agile Manifesto*[1] introduced the term in 2001.

Agile Model – *Cont'd*



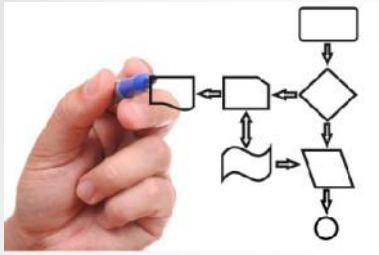
Why Agile Methodology

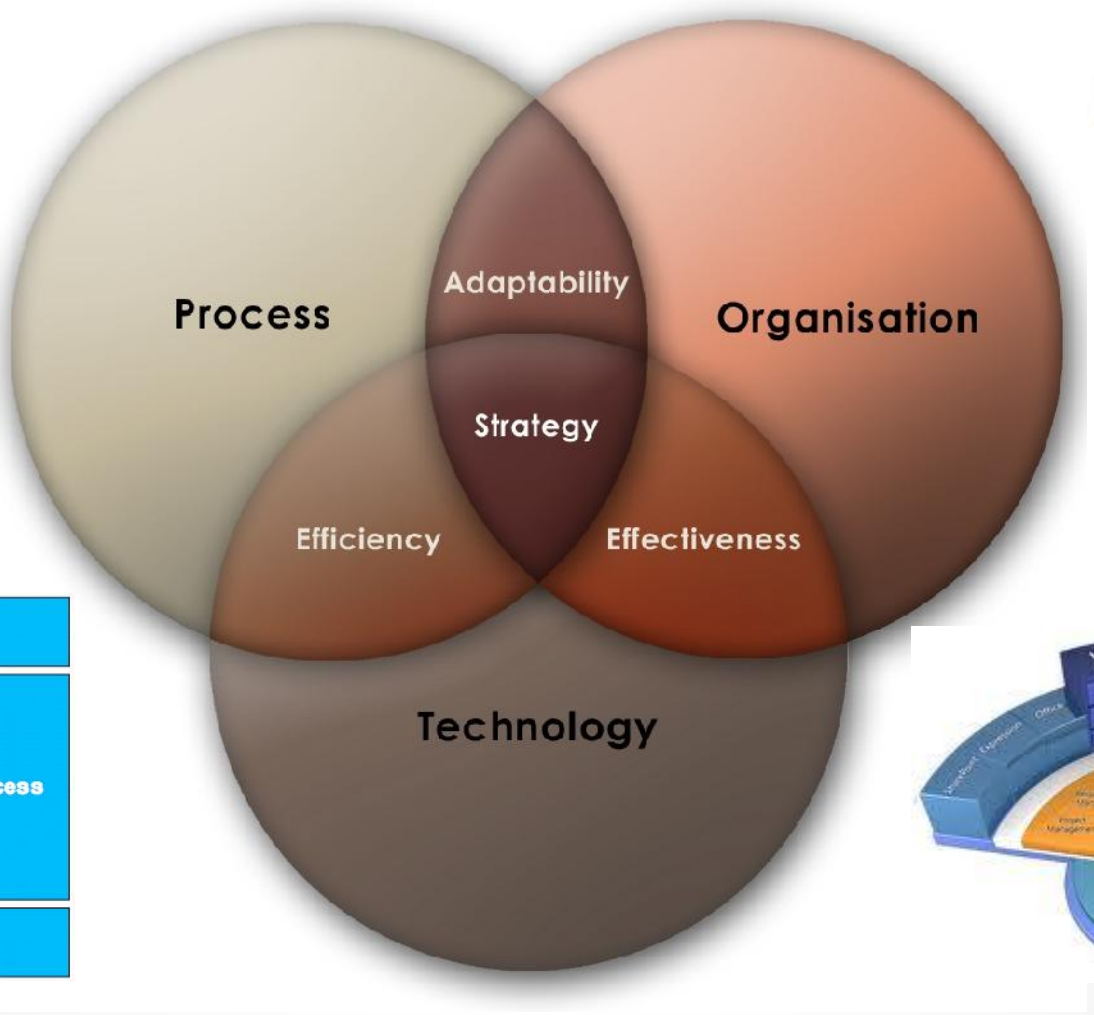
- No documentation or very little
- Development without requirements
- Adhoc processes or barely existent
- Little training required
- Self adjusting teams, no management layer
- Little oversight, I do as I please and in short bursts
- Loose commitments and accountability
- I code half the time and browse the internet half the time
- Pays me more for less

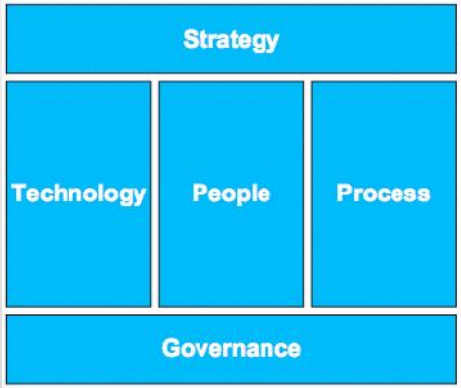
Why Agile Methodology

- No documentation or very little
- Developm out requi
- Adhoc pr hare
- Little training
- Self adjusting te gement layer
- Little oversight, and in short bursts
- Loose comm tability
- I code ha and internet half the time
- Pays me mo or less

Basics




Process
Adaptability
Organisation
Strategy
Efficiency
Effectiveness
Technology


Strategy
Technology
People
Process
Governance


Visual Studio
Ultimate
Premium
Professional
Team Foundation Server

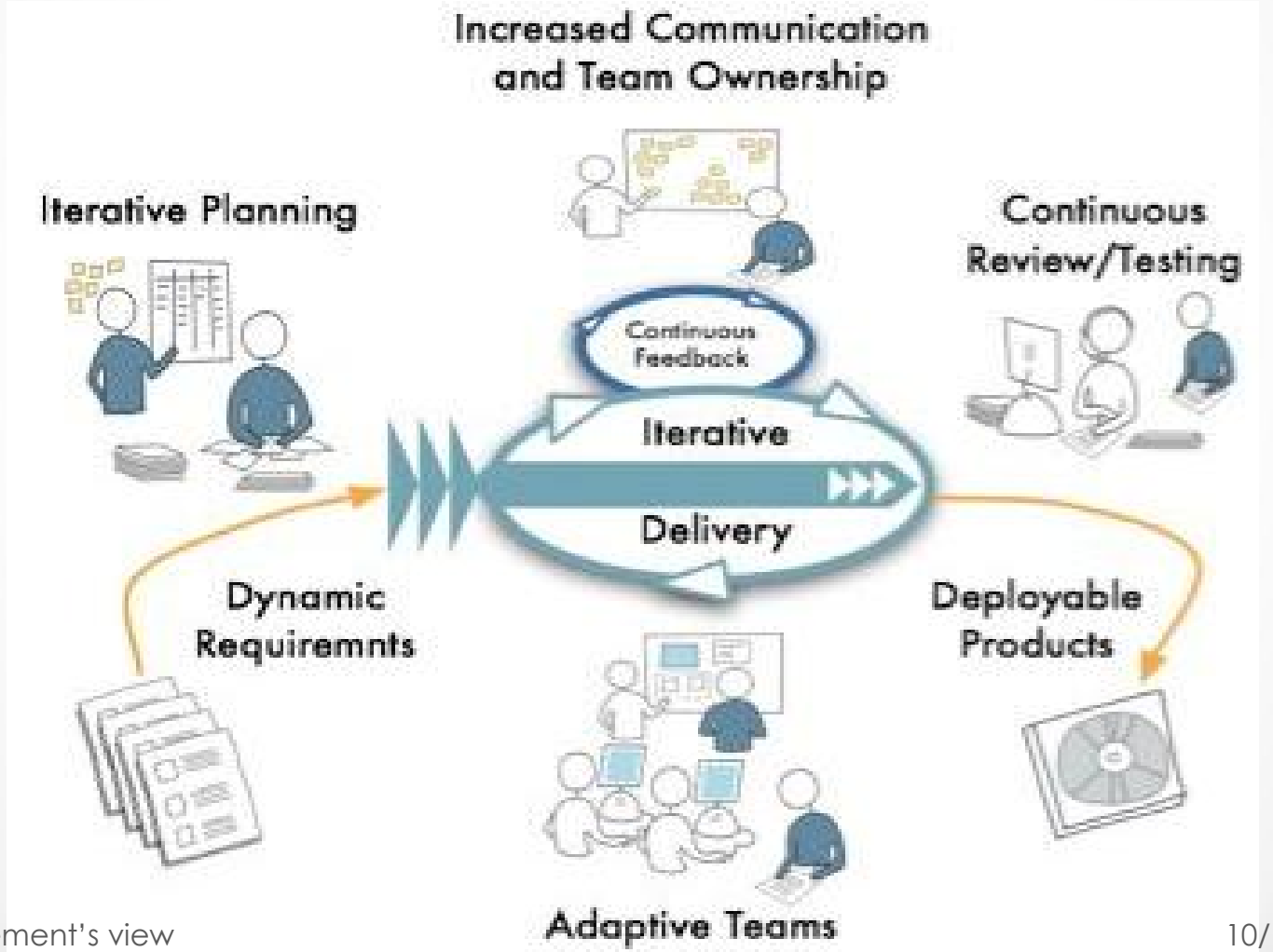
Real benefits in Agile

- Increased delivery of requirements by 300%
  - From 1 year to every 3 months
  - Quicker ROI
- Increased sales by over 120% year of year
  - Consistently grown online presence three years running
  - Now a major pillar in overall sales
- Teams are empowered and energized
  - Self adjusting teams
  - Actively learning from each other and increasing their effectiveness
- Quality increases
  - Small increments of functionality are deployed
  - Better monitoring is in place
- Production stabilizes quicker

From the top

- Make no mistake … going from Waterfall to Agile **IS** a MAJOR project
  - Change Management at the core
  - Six months to two years for first wave
- Options
  - Do it internally (grass roots, POC)
  - Vendor aided
- Senior sponsorship, Partnership and Decision makers
- Funding is key
- MUST HAVE Team Support
- Training
- Other projects don't stop

The promised land

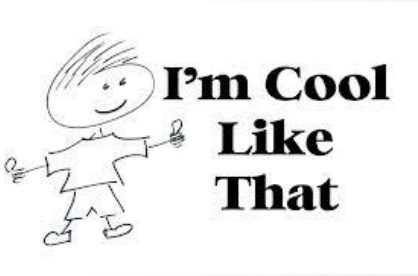


But first – a little pain

- The team has to 'jive'
- Not everyone gets it on the first try
- Three to six sprints to get a usable velocity
- Done definition stabilizes

Readiness

- Being Agile is "**In**", "**Cool**", "**Progressive**", "**Hip**"… but it's more than that !!
- Here are some key things to think about:
  - Company's Culture – Don't try to change it
  - Business landscape – Are there key drivers within the company or economy driving the search for faster solutions?
  - Are you looking to change the entire IT department or just your immediate team – Size DOES matter. No shotgun approach, start small and then scale out
  - How is the money allocated? Can you easily get additional resources as the need arises?
  - Timelines flexible? Can you afford to take a crack at it and if failure try again?
  - Product fits well for agile? Is this a centralized SW scope with one clear owner or multiple owners with variant degrees of control and demands not always aligned with each other?

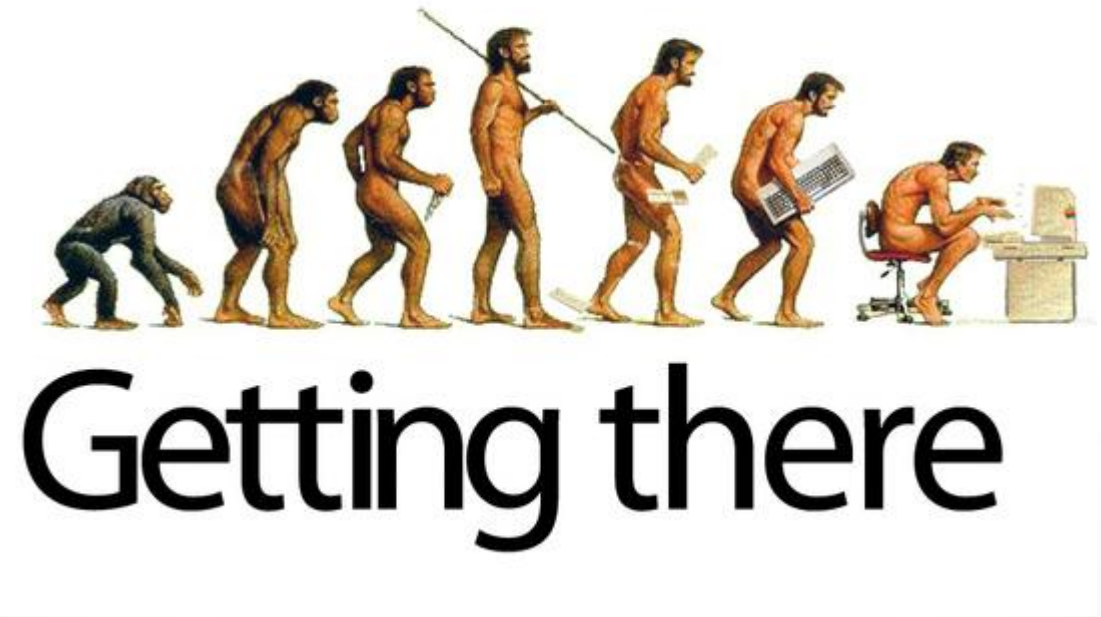


Readiness – *Cont'd*

- Jorge's 10 Point checklist
  1. **Strategy** – Does the business and IT have a clear picture of what's needed (Themes and Epics, prioritization scheme)
  2. **Partnership and Commitment** - Is the business ready to commit dedicated resources and own the results (beginning to end – Product Owner); Dedicated resources in IT: Scrum Masters, BAs, Tech leads, Devs, Testers, Support
  3. **Funding** – Is there money behind this change? Both in support of external help, training, adaptation, tools, monitoring and controlling? Staffing Pattern – cross functional team (5-10) dedicated
  4. **Training** – Ramp up, Coaching, Lessons learned
  5. **Environments and Tools:**
     - Devs, Single/Multiple builds, Systest, TFS
     - Automation, UAT, Security, Staging,
     - Monitoring; Tealeaf, Splunk, Omniture, Other
  6. **Automation** – build faster, scan faster, test faster, deploy faster
  7. **Quality Assurance** – Quality Control and quality gates
  8. **Pipeline management** – Release management, scrum of scrums
  9. **Reporting** – Metrics and Auditing, Dashboards
  10. **Governance** – Funding, Scope assignment, Standards, KPIs

- Another example by Elena Yatzeck @ http://pagilista.blogspot.nl/2012/12/a-corporate-agile-10-point-checklist.html

Enough about Readiness How do we get there?



An Approach

- Coming up with a Plan – 7 key areas
  1. **Understand your immediate target audience** - Interview team members (get perspectives, readiness level, supporters)
  2. **Alignment with the business sponsors** - Meet with key stakeholders and business sponsors; confirm their understanding on what you are about to launch
  3. **Extract key drivers** (technical and business drivers for change) - common themes and value add dimensions
  4. **Choose an agile methodology** – Need a framework to guide your team
  5. **Prepare documentation** – Rules of engagement, presentations and communication plan
  6. **Execute** – Deploy POC, measure and report; communicate
  7. **Support** – Coach, remove obstacles and reinforce where needed
- Determine key team members – POC should contain **your top** supporters for change (both technical and business)
- Prepare marketing plan – YES… marketing plan; out of sight out of mind
- Ensure timely feedback to key decision makers on progress of POC
- This is a mental state and a Journey

Do's

- Confirm companies readiness
- Ensure at least one business senior management leader (preferably CXO level) is behind you
- Clear budget – Money DOES move everything
- Plan for transition – Org charts (old and new teams)
- Capacity Plan - Both from yourself and targeted team
- Consultants – Don't be a Hero; bring outside help
- Standard operational variables
  - QA Framework and toll gates
  - Quality Control
  - Testing and Automation
  - Builds and integration
  - Configuration Management
  - DevOps

Do's – *Cont'd*

- Get your best resources – POC
  - People who are excited about it
  - People who can drive with little direction
  - Proven performers
- Get a small risk project
  - Choose a project well – start small and scale slowly
  - You'll need at least 6 months before you can claim some victory
- Product Centric Model
  - Get buying from the business to structure a PCM
  - Align your scrums to product lines
- Training – Ensure everyone gets a solid baseline
  - Agile methodologies are inherently lean but everyone needs to see the same picture
  - Everyone should use the same terms and they should mean the same to everyone
- Time bound – Commit to a trial period and then measure - POC
- Give SCRUM a try … it's a proven methodology
- QA / QC – This is VERY important, have a Test Plan
- Automation – In order to achieve your maximum you'll need this
- Technology – Choose your tools carefully, they will make you or break you

Dont's

- Grass roots – May work short term but it won't go far
- Scattered team – No full commitment from team
- Don't start if you are not truly ready
- Not enough funding
- Leadership
- Properly trained resources
  - Product Owner
  - Scrum Master
  - Technical Lead
  - Business Analysts
- Under estimate Scaling out (1 to many) oversight layer
  - Pipeline management
  - Release management

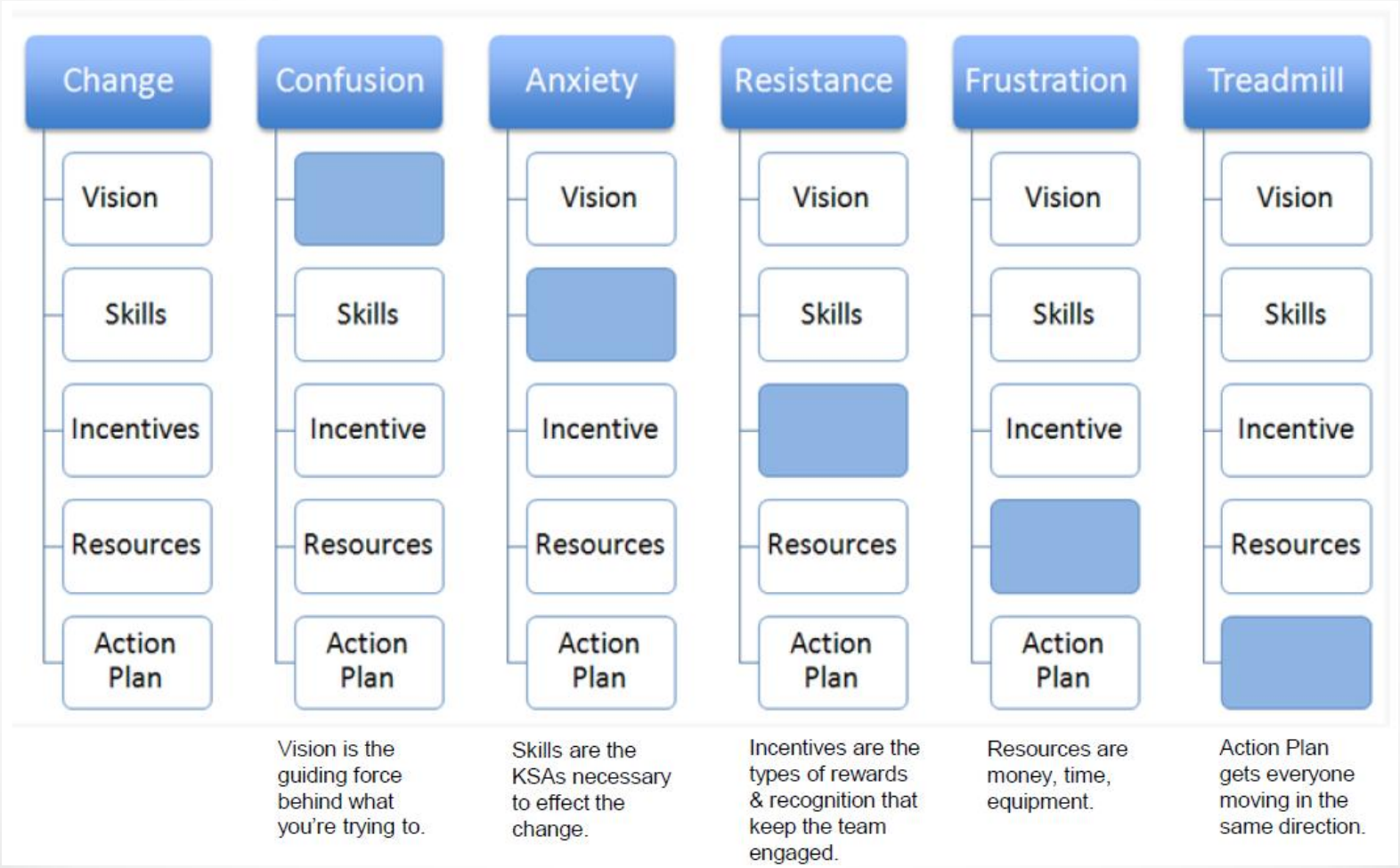
Dont's – *Cont'd*

- Tools – don't underestimate the criticality of tools
- Supporting teams – Don't be alone, it's a family
  - Configuration Management
  - Engineering
  - Production Support
- Under communicate – Key documentation, model relationship of agile teams and business ; other IT teams
- Estimation models – Waterfall and Agile (it's a reality)
- New resources – Plan for adequate training and integration (scrum masters, BAs)

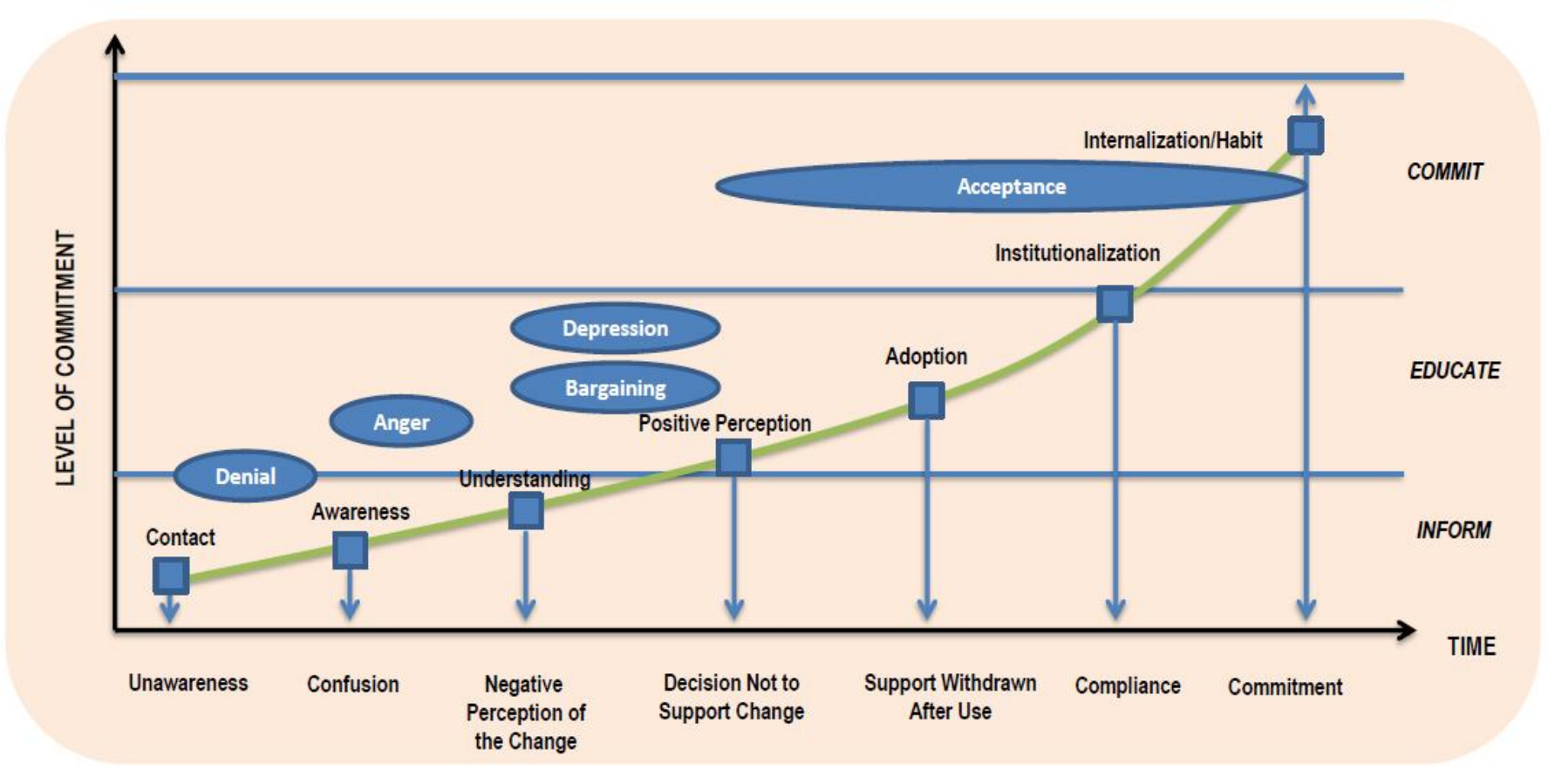
Executing The Change Management Program

(This is not just a dev exercise)

Managing Complex Change



Building Commitment

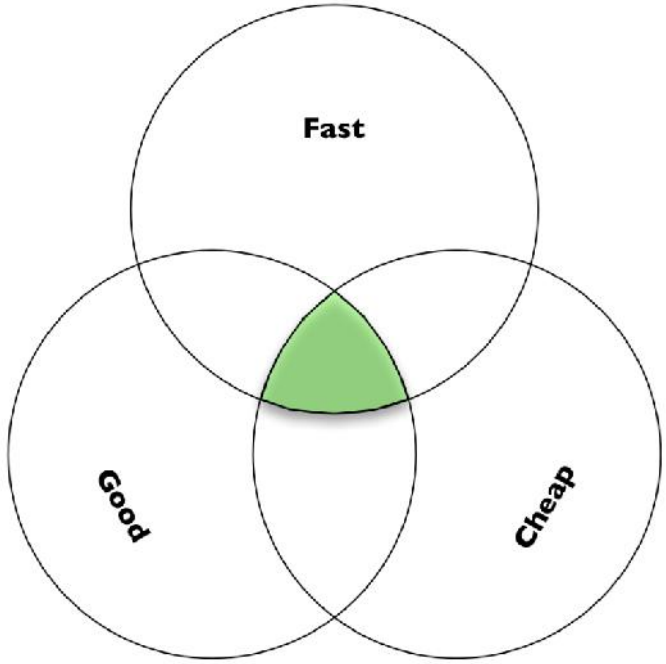


Continuous Delivery

- All forces need to align across all supporting teams to have a well oiled delivery machine
- Continuous Delivery is a core concept to Agile
  - CM, Integration, Automated builds, Scans, Automated Tests, Deployments (DEVOPS)
- Production monitoring
  - Did our deployment succeed? How do you know?
  - Tools – Tealeaf, Splunk, Omniture, Other logs
  - System behaviors – Sales are the same or did they drop/increase? Call center volume?
  - Adaptive monitoring

Take Away


We offer 3 kinds of service:
GOOD - CHEAP - FAST
GOOD service CHEAP won't be FAST
GOOD service FAST won't be CHEAP
FAST service CHEAP won't be GOOD


Fast
Good
Cheap

Q & A