Post-Agile approaches – Agile for the real world

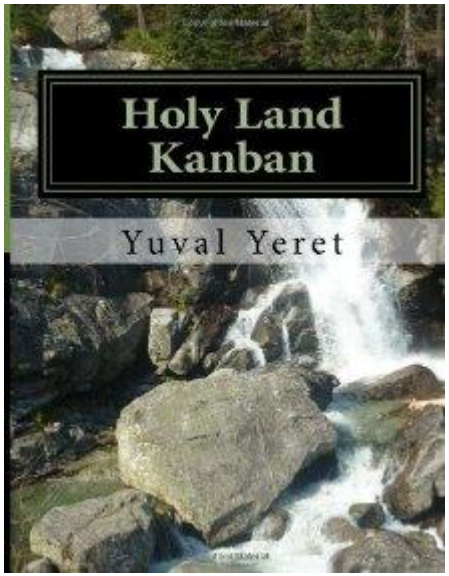
Yuval Yeret
Senior Kanban/Agile Consultant @
www.AgileSparks.com

AgileSparks

FELLOW
Lean Systems Society

YuvalYeret.About()

- Senior Kanban/Agile Consultant @ www.AgileSparks.com
- Blogging at http://YuvalYeret.com
- Author of Holy Land Kanban https://leanpub.com/holylandkanbanbestof



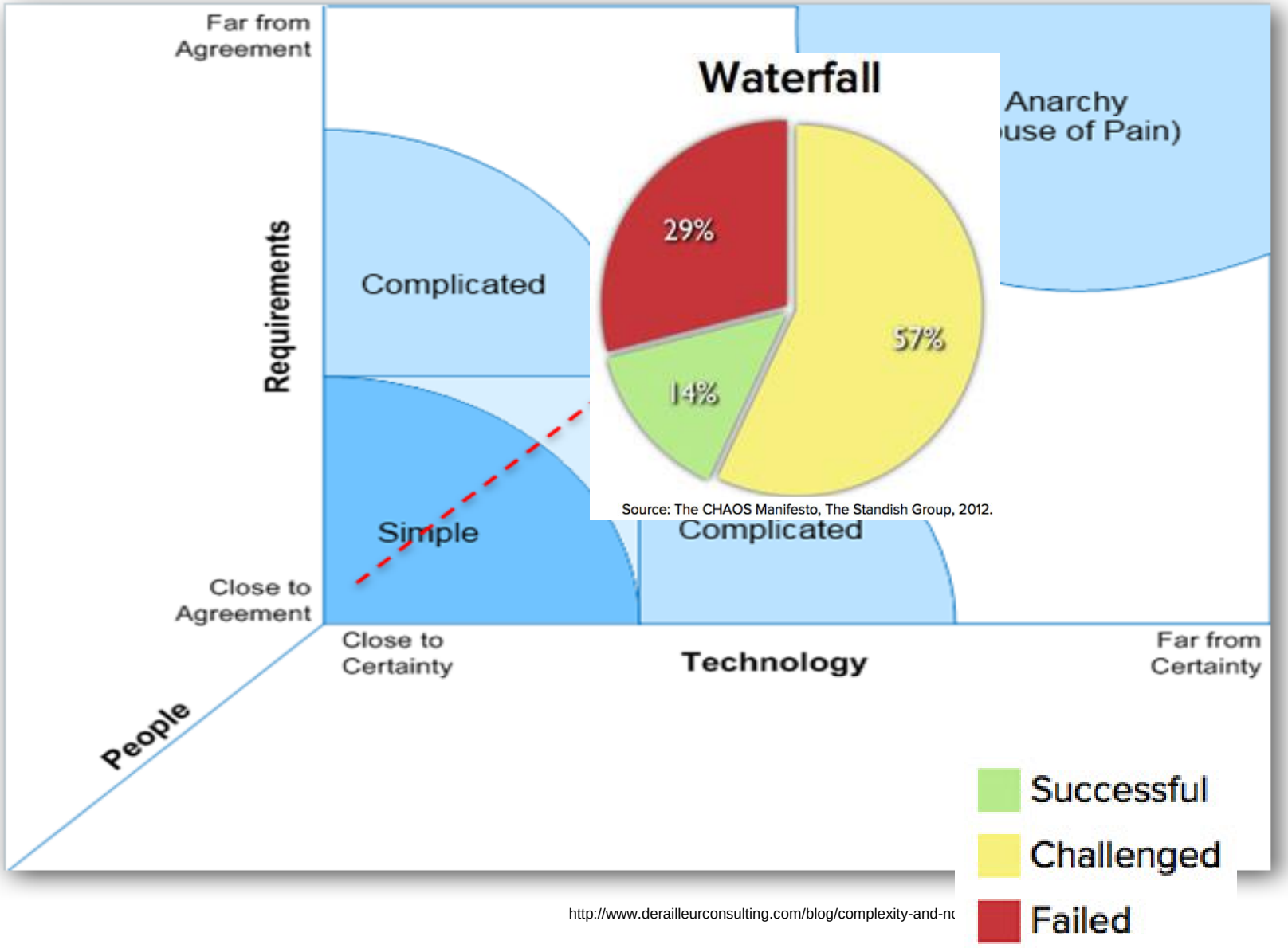
What is Agile all about?

http://www.faculty.umb.edu/gary_zabel

"The RIGHT thing" is a hard to nail moving target

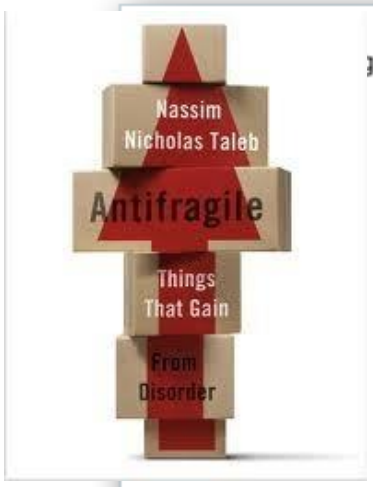
"I'm losing sleep over whether we are actually doing the right thing in this big project and not wasting our time" IT Director in a major telco

We live in uncertain times...



AgileSparks

We need approaches that embrace uncertainty/complexity



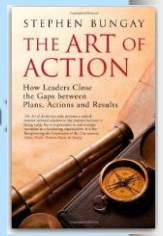
Far from
greement

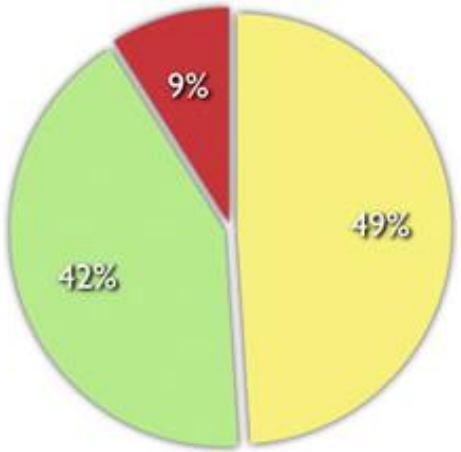
Requirements

Complicated

Agile

Anarchy
use of Pain





Source: The CHAOS Manifesto, The Standish Group, 2012.

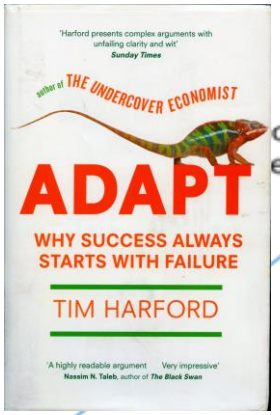
Simple

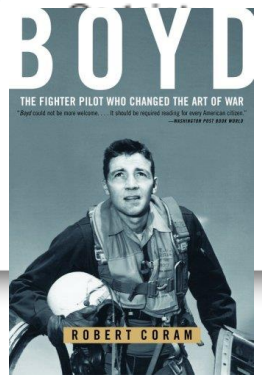
Complicated

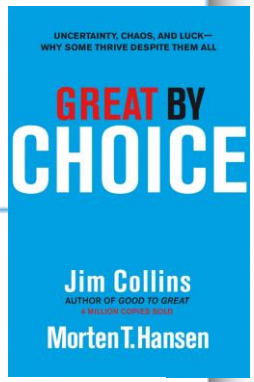
ose to
ement

Close to

Technology







http://www.deraille

ıd-n

Successful

Challenged

Failed

AgileSparks

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value :

Individuals and interactions

over

Process and tools

Working software

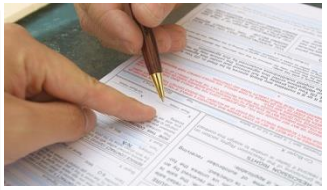
over

Comprehensive documentation

Customer collaboration

over

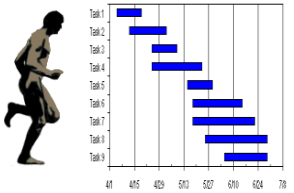
Contract negotiation



Responding to change

over

Following a plan



While there is value in the terms on the right
We value the items on the left more
(http://www.agilemanifesto.org)

AgileSparks

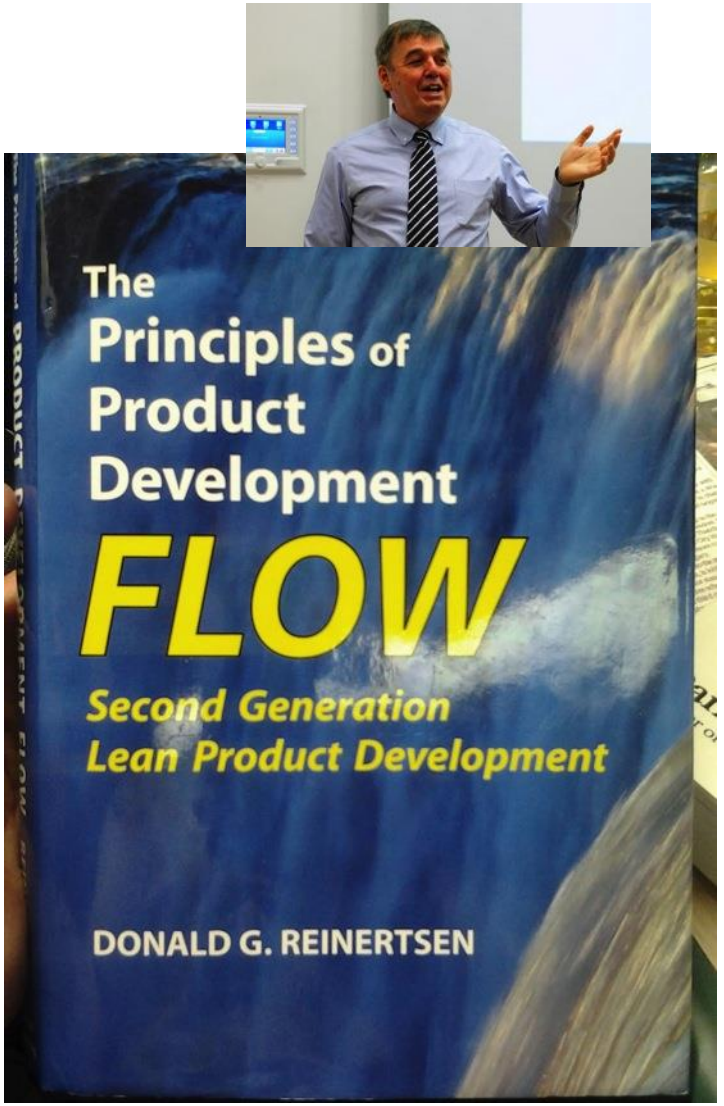
Principles behind the Agile Manifesto

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Principles of Lean Flow

- Time=Money – Quantify cost of Delay
- Reduce batch sizes and work in progress to enable faster feedback and faster time to market
- Focus on flow/response time rather than efficiency
- Reduce transaction costs to enable smaller batch sizes
- Treat plans as hypothesis that will evolve as new information becomes available
- Focus on quick feedback instead of first-pass success



Can you find principles that don't match your reality?

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- aspiring to Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

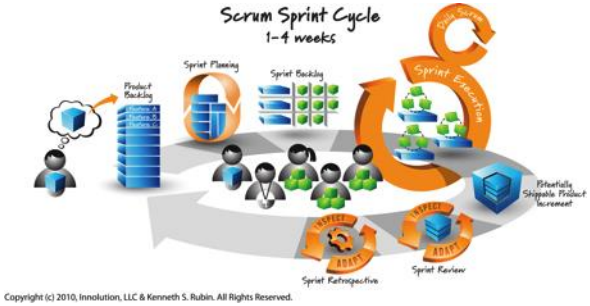
- Time=Money – Quantify cost of Delay
- Reduce batch sizes and work in progress to enable faster feedback and faster time to market
- Focus on flow/response time rather than efficiency
- Reduce transaction costs to enable smaller batch sizes
- Treat plans as hypothesis that will evolve as new information becomes available
- Focus on quick feedback instead of first-pass success

History of key Lean/Agile Frameworks

Kanban - 2010s

Evolve into Lean/Agile

Focus on the flow



Scrum - 2000s

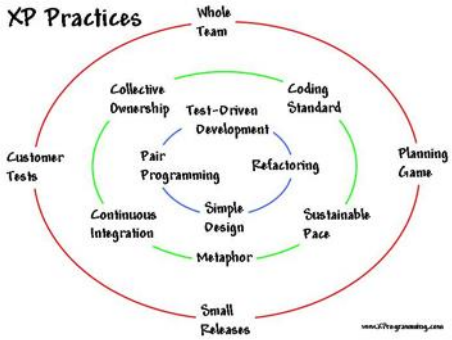
Jump into Lean/Agile

Still the leading classic agile approach

Extreme Programming - 90s

Lean/Agile Engineering practices

Bring in the right practices at the right time, but ignore at your own risk!



AgileSparks

Looking for Failure

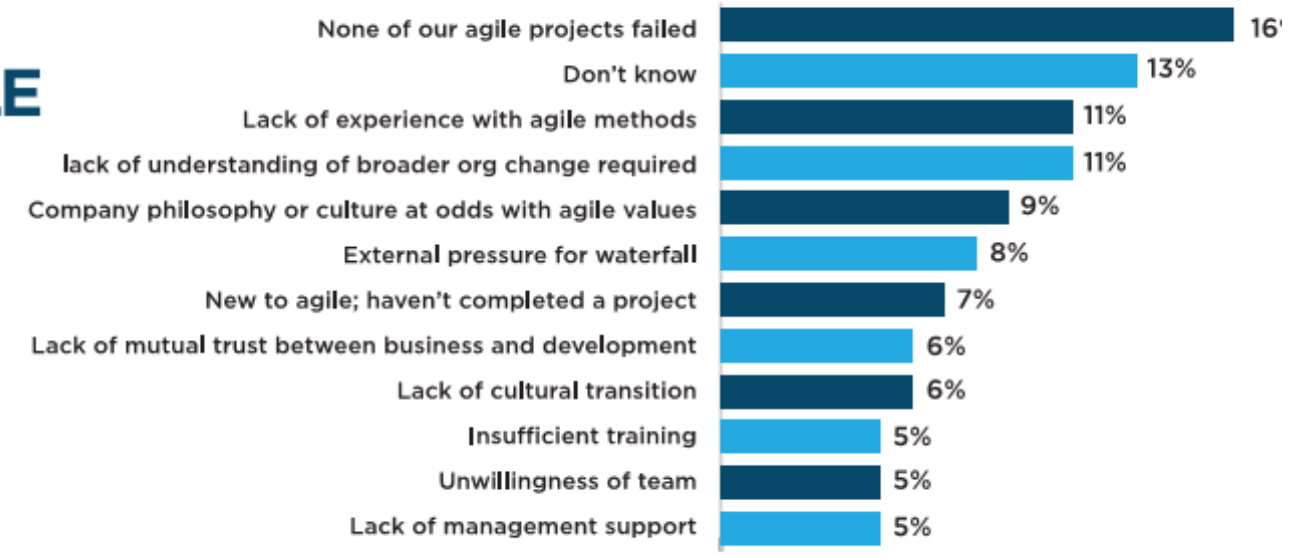
- Do you have situations where Agile DIDN'T/WON'T work?
- Let's try to classify them into groups of root causes for potential/actual failure

What are the challenges?

Why do agile PROJECTS fail?

LEADING CAUSES OF FAILED AGILE PROJECTS

Most respondents said none of their agile projects would be considered unsuccessful (16%). Of those with failed agile projects, most said it was due to either a lack of experience with agile methods (11%) or not understanding the broader organizational change required (11%).



http://www.versionone.com/state-of-agile-survey-results/

AgileSparks

What is stopping further adoption in your organization?

BARRIERS TO FURTHER AGILE ADOPTION

For over half of respondents, the inability to change their organization's culture was the biggest problem. Budget constraints had the lowest impact on further adoption (14%).

http://www.versionone.com/state-of-agile-survey-results/

What is concerning you about Agile?

GREATEST CONCERNS ABOUT ADOPTING AGILE

The most common concerns listed by respondents when they were considering deploying agile were a loss of management control (33%), lack of upfront planning (33%) or management opposition (32%).

*Respondents were able to select multiple options.

http://www.versionone.com/state-of-agile-survey-results/

Early Feedback / TTV – The Goal and the conflict...

Improve – Reduce Transaction Overhead

Ideal WIP
WITH
Leaner more agile
processes

Ideal WIP
With CURRENT
processes

Current WIP

Earlier Feedback/Faster TTV/Cheaper to deal with feedback

Lower process overheads

Changes that Agile will drive:

- Infrastructure Investments
  (release automation, test automation, etc)
- Evolve the organizational
  (new roles, cross-functional teams, etc)
- New skills
  (Vertical story-slicing, retrospectives, agile architecture, etc)
- New habits
  (Frequent customer interaction, frequent release, less specialization)
- Transparency
  (problems and uncertainty painfully visible rather than hidden)

What will happen if we don't do this?

But no need to do it all at once

Try Scrum here...

- 20 different applications (each developer knows 1)
- Build takes 2 days. Dev Sanity is 30 man days effort.
- Hardening/Release cycle takes 600 man days
- System has lots of technical debt – a tough swamp to move through – very hard to finish story design+development in less than 7-10 days.
- Business is built of silos – no single empowered product owner for the entire backlog nor for each MMF.

Scrum SWOT analysis

Strengths

- Simple
- Structure brings safety, comfort
- Strong improvement engine done right
- Very popular, lots of people know the basics
- Many examples of organizations that did it
- Can scale if you build it from real scrums

Weaknesses

- Few understand the real engine that makes Scrum work and therefore abuse it
- Hard to do right and fragile when abused
- Too many think they know how to do it (but don't)
- Requires tech/org revolutions to work
- Scaling requires full pure scrum, not really applicable for most legacy enterprises

Opportunities

- Raise awareness to the patterns/engine that is required for Scrum to really work to increase success with it
- Evolve from Scrum Master to Agile Lead
- Clarify role of management in the picture

Threats

- Growing frustration by people that experienced bad Scrum that don't want to work in an agile env again because of it
- Legacy organizations don't want or cannot change very fast, which makes Scrum less relevant there.
- Lack of clear role for Management

Agile SWOT analysis

Strengths

- Very effective when done right
- Lots of success stories and ways to go about it that fit different scenarios
- Driven by a real business need as well as support by a strong ecosystem these days

Weaknesses

- Classic agile approaches struggle with the legacy enterprise for which the agile “crash diet” is tough to bear
- Like many other great approaches - watered down and became “cargo cult” as it crossed into mainstream

Opportunities

- Emphasize the thinking/principles over the practices
- Find ways to go agile towards agile to accommodate legacy enterprises
- Provide better scale blueprints to clarify the direction for legacy enterprises

Threats

- Agile Zealots are a turnoff for the pragmatists in the mainstream market
- Growing dissatisfaction over bad practices-focused initiatives
- Growing frustration from legacy management abusing agile to serve their micro-management push-based stressful blame-culture

Kanban – EVOLVE towards Lean/Agile

- Smaller Work Units – MVP/MMF/User Stories
- Visualize – Understand **your system**
- Manage Flow - Stop starting start Finishing
- Make Process Policies Explicit – to **enable empowerment/delegation to teams and people**
- Optimize WIP – To start driving improvement, eliminating wasteful waits/multi-tasking
- Implement Feedback Loops – **Own your process**
- Improve Collaboratively, Evolve Experimentally (using models/scientific method)

Kanban SWOT analysis

Strengths

- Easy to start with
- Solid scaling approach even if not purely agile
- Applies end to end not just to the development team
- Accommodates legacy constraints
- Principles focused – liked by people who buy into the principles but want to customize their approach
- Applies to your process, no need to create new process from scratch.

Weaknesses

- Without limiting WIP and focusing on flow will not provide predictability or drive improvement
- Confusion over "is it a tool, is it a methodology, is it a method" leads to people not leveraging it effectively.
- The new boy on the block – has to overcome lots of Scrum propaganda/FUD against it
- Requires patience and determination due to evolutionary nature
- Cadence/Rhythm is optional and ignored by many – weakening flow of feedback, delivery and improvement
- Lack of structure confuses & frightens a lot of people

Opportunities

- Emphasize Cadence/Rhythm by combining with Scrum elements
- Leverage real success stories that are out there
- The favorite approach of the "DevOps"/"SaaS"/"Continuous Delivery" crowd (and for good reasons as it is the best fit for their needs)

Threats

- SAFe is easier to grasp for the common enterprise
- Scrum is quite entrenched at the team level

Add to Kanban from Scrum

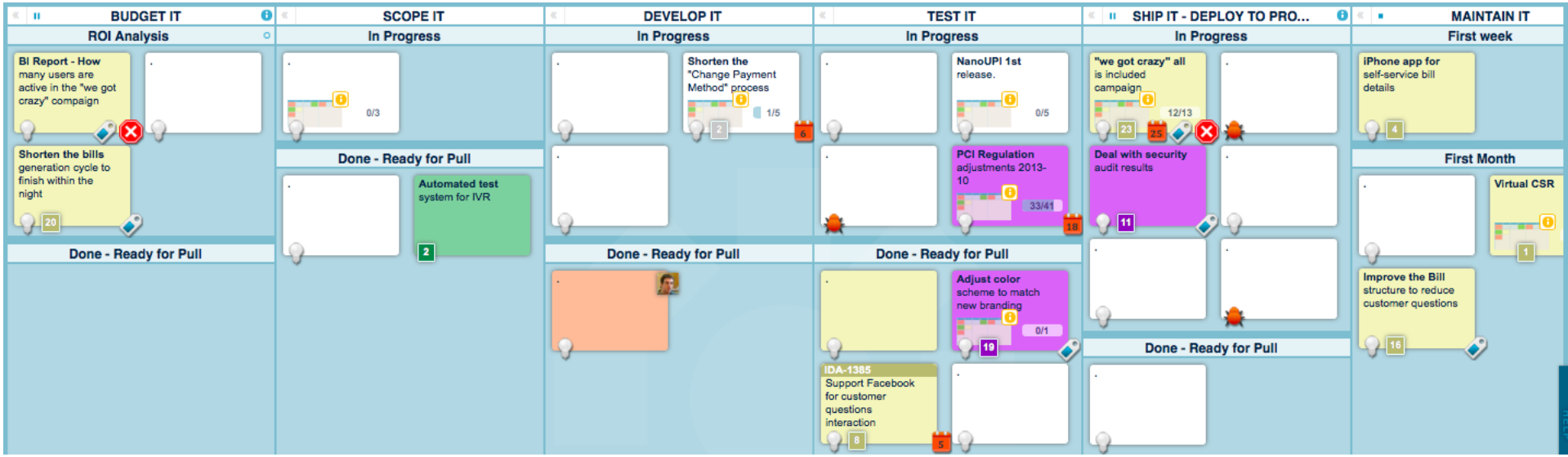
- Rhythm of the Sprints/Iterations – Product Feedback, Process Feedback, Adapt the plan and the process
- Agile Teams – when it makes sense – depends on the context, timing, readiness for change
- "Scrum Master" / "Agile Lead"?
- Product Owner?

ScrumBan – Evolve your Scrum towards Flow using Kanban

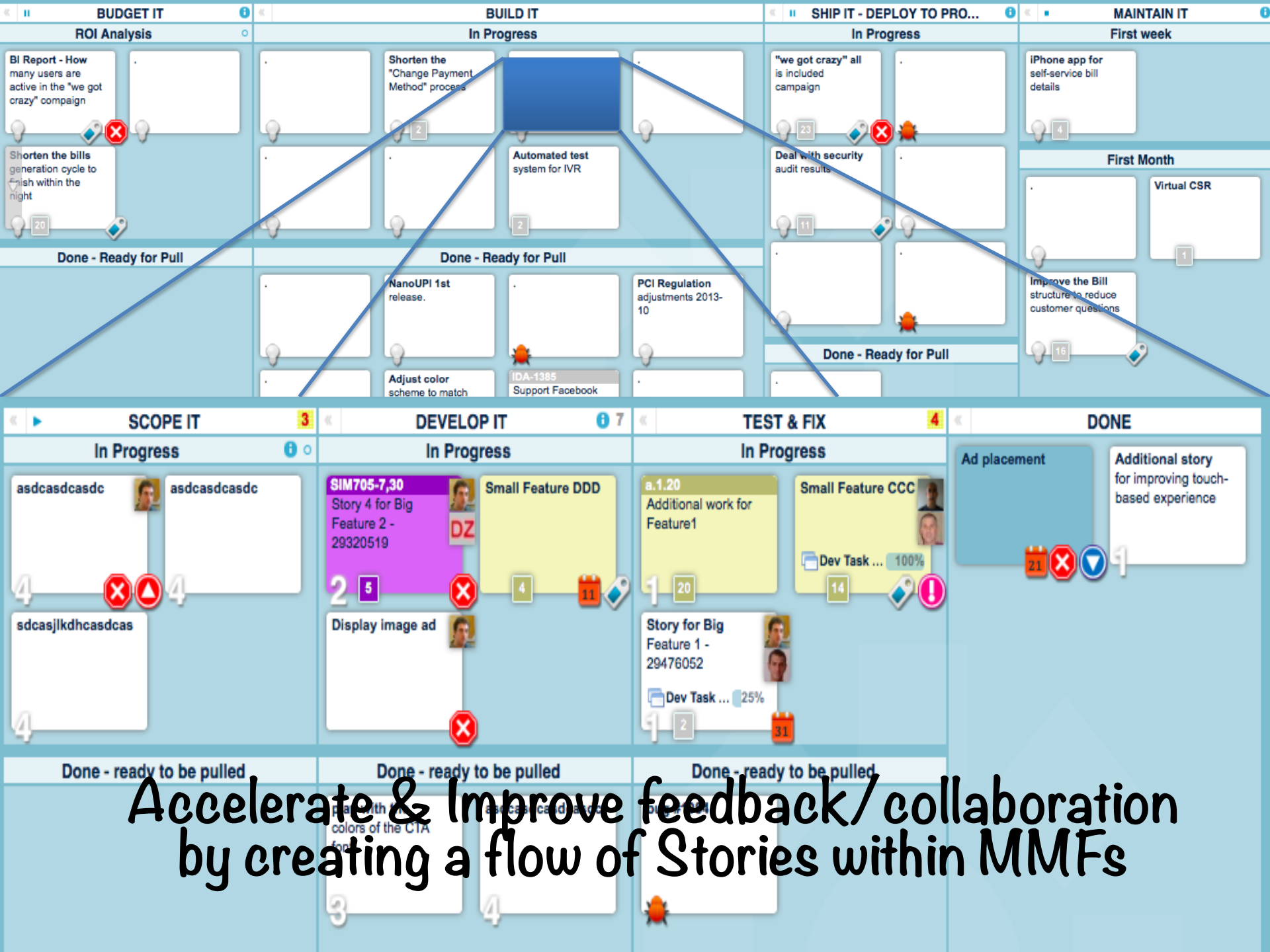
- Visualize using Kanban board
- Manage Flow - Stop starting start Finishing WITHIN a sprint and across the upstream/downstream
- Measure and focus on cycle times within and across sprints
- Allow flow across sprint boundaries, using WIP Limits to drive focus/collaboration/predictability rather than a strict sprint commitment

Scaling using Kanban

Deliver an end to end flow of Minimally Marketable Features (MMFs)

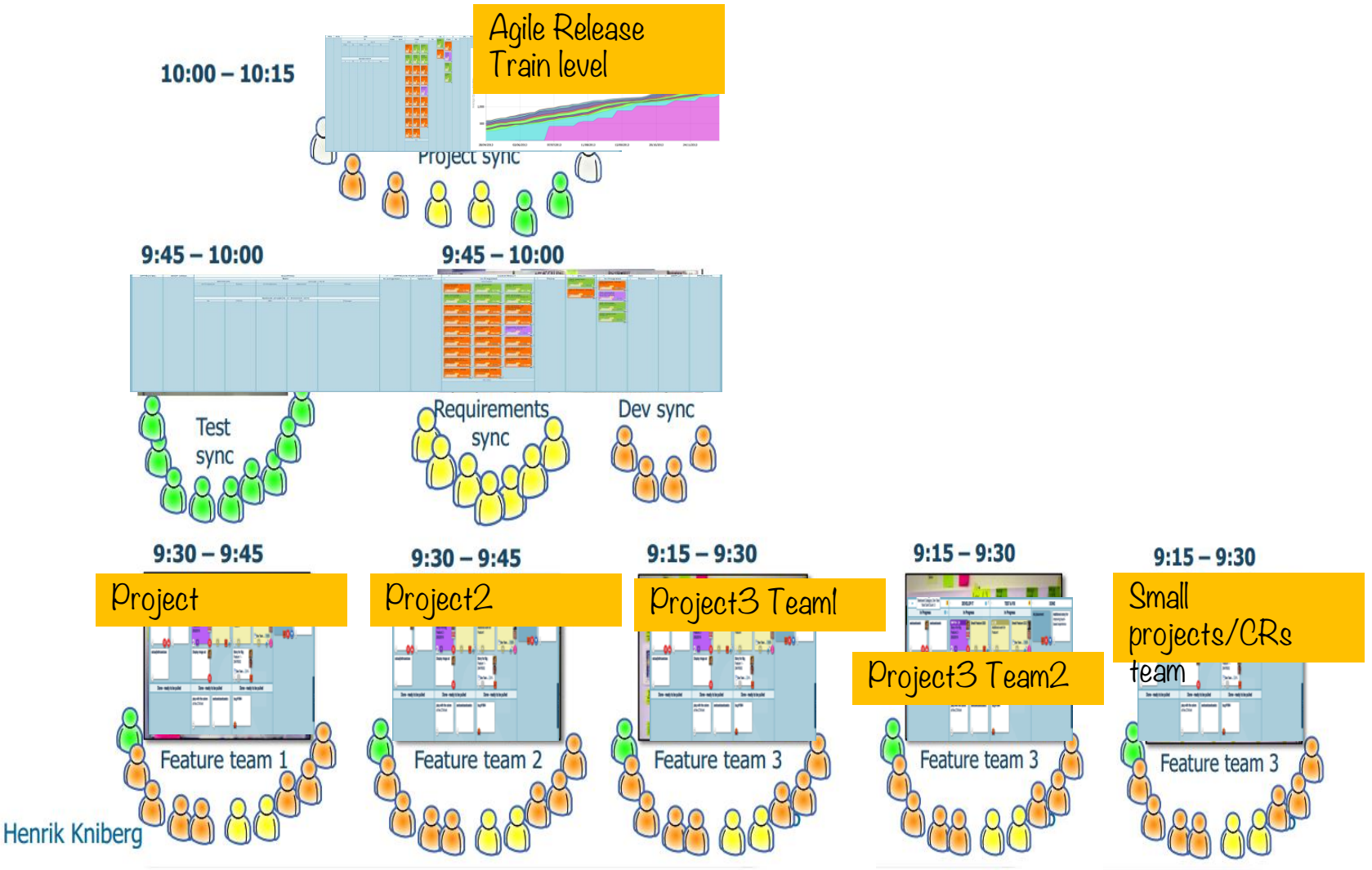


AgileSparks



Accelerate & Improve feedback/collaboration by creating a flow of Stories within MMFs

Managing flow across the program



A different approach to Scaling Agile

SAFe SWOT analysis

Strengths

- Very elaborate – answers many questions in a single place
- the legacy enterprise can see itself doing this
- Built on good proven principles and practices
- Integrates well with Scrum at the team level

Weaknesses

- If treated as a "take it and adopt it" might kill evolutionary engine towards good fit for the organization
- Relatively new – not much knowledge and experience out there
- Copyright/IP nature frightens/confuses practitioners/consultants.

Opportunities

- Rides the wave of the confused mainstream legacy enterprises that have to become more agile these days. The elaborate prescriptive guidance to scaling agile is what they want… (even if not always what they need…)
- Emphasize its nature as a guide book rather than a guided tour
- Make it easier to work with from a copyrights/IP perspective

Threats

- Oriented towards "Slow agile" not the future of "Continuous Delivery" so if "DevOps" strengthens it becomes less relevant
- When organizations build the right decoupled agile teams no need for it (see Spotify, Amazon for example)

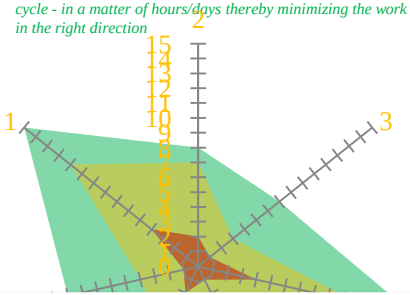
http://scaledagileframework.com/

1. Visualize & Manage the Flow

1. **Visualize main Work types (using Kanban Board or similar) to create flow awareness**
2. **Definition of what Done (Working Tested Software) means is clear and adhered to ("DoD") so real flow is measured and so exceptions drive discussion/improvement.**
3. **Visualize who is working on what in order to be aware of level of multi tasking and dependency on specific people.**
4. **Commitment to finishing work over starting new (eventually reaching a WIP level that "feels OK" for the team) to start to "weakly" constrain and improve flow.**
5. Use flow diagrams/charts (e.g. CFDs) to provide predictability and insight into flow
6. Visualize and focus on blocked work so major flow efficiency issues are addressed
7. Visualize work that is queued/waiting between people/workflow states to start raise to awareness reasons for queuing and identify options for reducing
8. Awareness of Work Types and Work Items and differences in handling, in order to enable expectation setting with different stakeholders for different needs & allow people to make intelligent flow decisions according to the context
9. Some areas in the flow have local work in process (WIP) limits - leading to lower WIP and cycle times and more explicit opportunities to learn from the (lack of) flow
10. Visualize work variability and seek to reduce it (e.g. using Cycle Time Control Charts) so that overall average cycle time is improved and there is less uncertainty about velocity/cycle times enabling more aggressive planning
11. Explicit WIP limit at workflow level - Single workflow full pull – catching more flow problems and driving WIP/cycle time even lower.
12. Next is re-prioritized continuously (no commitment in Next)- Deferred Pull decisions (dynamic prioritization) in order to enable business agility.
13. Definition of what "Ready for work" means is clear and adhered to in order to minimize rework or blocks due to unready work occupying the WIP.
14. *Guidelines for how to pull work (selection from 'Next'/prioritization of WIP) are clear to everyone and adhered to so that most decisions can be decentralized and made faster as well as driving discussion about how to work and resulting in experiments/improvements*
15. *Capacity is allocated to Investment Themes using work in process limits so that it is possible to ensure certain investment in each theme.*

2. Business Value Driven Development

1. **Product owner sees working software frequently and uses the feedback to adapt the scope/timeline plan**
2. **Work items are integrative and testable cross-cutting across the architecture if necessary (e.g. User Stories). Done = Deployable and Performant/Secure, enabling real feedback/learning.**
3. Work items are integrative testable & SMALL - can be delivered in days thereby tightening the internal team level feedback loop
4. frequent feedback from stakeholders/users is used to adapt the scope/timeline closing a real feedback beyond the product owner.
5. Escaping Defects and other kinds of Failure Demand (Waste) are analyzed using Five Whys or another kind of root cause analysis process in order to determine reasons for missing them earlier in the process.
6. Value is delivered in iterative chunks using Minimally Marketable Features (MMFs) thereby achieving business agility – faster time to market and keeping more options open to what will be delivered after the current MMFs.
7. *Requirements that are Hypothesis are validated Using MVP/MVF in a fast learning loop that includes Beta/Early Access programs or Continuous Delivery, in order to enable safe/cheap-to-fail experiments about risky but worthy ideas.*
8. *Feature Usefulness and Successfulness is evaluated as part of the development lifecycle. Learning is applied to improve the feature and future ideas.*
9. *Frequent Delivery to real users - up to 8 weeks apart*
10. *Continuous Delivery - work items are deployed/activated/validated as part of the work life cycle - in a matter of hours/days thereby minimizing the work done without feedback that it is in the right direction*



3. Individuals & Interactions Feedback Loops

1. **All people involved in a work item work on it more or less in the same time period (Developers, Testers, Functional/Product) minimizing the overhead/waste from context switching/recalling past work.**
2. All people involved in a work item (even across silos) can collaborate directly with each other without third parties like team leads in every coordination/communication loop enabling faster decisions and more scalable operation.
3. People working together act as a team with shared accountability to end to end delivery thereby decisions are more value than silo-focused
4. *Significant aspects of goals and rewards are oriented towards team performance/goals (rather than individual performance) driving collaboration not just individualism.*
5. *Team environment is as collaboration friendly as possible*
6. *Individuals are involved in performance feedback of the people they are working with, to encourage teamwork*

4. Engineering Practices

1. **There is a clear definition of what "Coding Done" means and people are working according to it**
2. **People are expected to write SOLID/CLEAN code and estimations reflect it**
3. **Automation coverage is planned and implemented as an integral part of production code implementation**
4. **Defects created as part of new development are fixed as early as possible and in any case before considering that work item as done**
5. There is a Test Automation Pyramid strategy guiding Automation coverage decisions (Preference to Unit Tests>>API tests>>UI tests)
6. de as part of "Coding Done" and
7. n (ATDD/BDD)
8. erified at build time using code
9. proving collective ownership
10. s are closed within hours
11. to do effective SW engineering
12. *ugly code, missing tests, etc.) is*
13. *ducing it*
14. *ownership - most tasks can be pulled by many members of the team without a major effect on efficiency*
15. *Technical Code Design is Test-Driven (TDD)*
16. *Regression cycle costs days at most (due to high level of automation)*

Shallow/Deep Agile

5. Build & Deployment

1. **Continuous Integration – automatic build running at least nightly**
2. **All code and artifacts are versioned using a single scheme**
3. Build is trigged automatically upon code checked in
4. Automated regression tests run as part of build and give a green/red binary answer (no need for analysis to determine success/failure)
5. Frequent check ins to common branch

6. Empowered Teams and Individuals

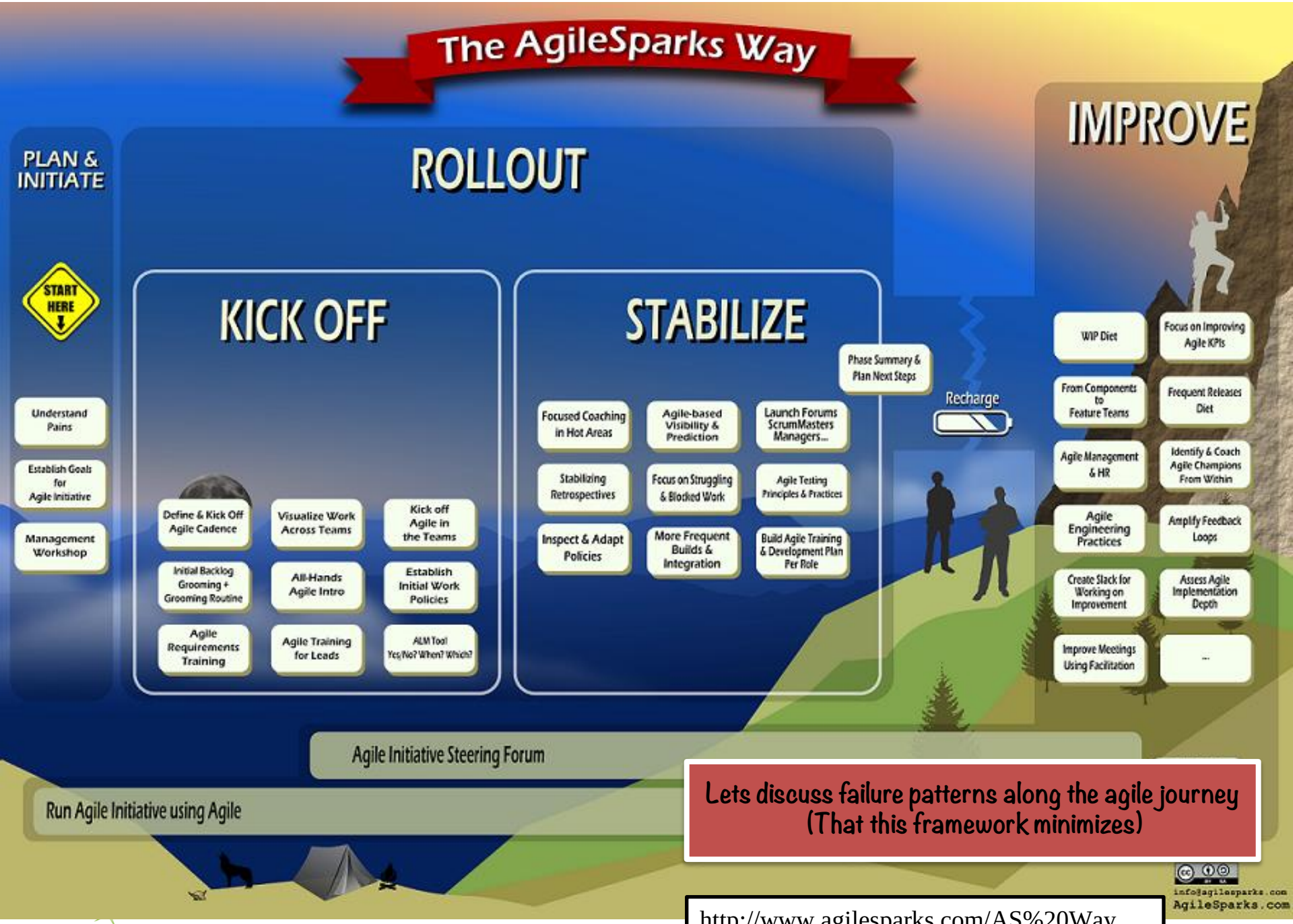
1. **Daily planning meetings (a.k.a. Standups) are used by people to manage their day to day work (instead of work scheduled by supervisors and pushed onto them)**
2. Autonomy - People have a high degree of control over the project day 2 day execution - Choose tasks to pull, where to focus
3. Reason/Intent is communicated as part of every requirement/work item, to increase motivation as well as empower people to do the right thing for the context rather than blindly follow a plan
4. People pull to capacity - by using Team Estimation approaches or just pull to WIP
5. Autonomy - People have a high degree of control over their personal & professional destiny
6. The behavior that is incentivized (formally and informally) is aligned with lean/agile thinking - Flow, Improvement, Trust, Whole Team, Low WIP, Safe to fail experiments, etc.
7. People work in small teams (not more than 10, ideally around 5-7) enabling good communication and direct collaboration as well as effe
8. Managers are pro-actively and methodically seeking w individuals as a way to enable faster decisions as well
9. People are given opportunity to improve their mastery
10. People can shape their work environment – technologi

7. Improve

1. **Regular Lessons Learned eve (frequency of no less than ev weeks) with actionable outco Retrospectives/Kaizen)**
2. People at all levels are highly involved in improvement activ
3. Actionable Improvement Work and managed using "Stop sta finishing"
4. Leaders are aware of the current operational capabilities (may require metrics)
5. Leaders have an operational capabilities goal
6. *Team/Group knows the current process challenge they are targeting*
7. *Team/Group knows what obstacles are preventing them from overcoming the current process challenge, what is currently being addressed and how*
8. *Team/Group allocates capacity/time slots for improvement work*
9. *Team/Group uses models to look at their condition and suggest experiments*

AgileSparks

http://www.slideshare.net/yyeret/leanagile-depth-assessment



Lets discuss failure patterns along the agile journey
(That this framework minimizes)

http://www.agilesparks.com/AS%20Way

To summarize, Agile CAN fail, if you:

- Go for it without a good reason/motivation or without communicating the purpose to people
- Choose the wrong implementation approach for your humane and technological context
- Expect it to happen auto-magically instead of requiring investment, attention, leadership
- Expect it to deliver results without any changes to culture/capabilities/way things are done at all levels
- Expect it to happen in a single linear big planning up front fashion
- Do it without the right knowledge/experience/guidance appropriate for your situation.
  (A legacy enterprise != small fresh technology startup)

Agile WILL succeed, if you:

- Start with the end in mind – Have a real clear purpose that everyone buys into
- Take into account the humane and technological context and choose an appropriate implementation approach and keep inspecting and adapting along the way based on what really happens.
- Prepare for investing time, money, and management attention
- Welcome the need to change how things are done (at all levels)
- Understand that it is an agile journey involving continuous planning and adjustment, safe to fail experiments, successes and small failures, and a lot of learning
- Make sure you have the right people and/or the right help/guidance to challenge you, inspire you, keep you safe, keep you focused, keep you confident you can make it.

AgileSparks

- Sparking Sustainable Delivery and Improvement Approaches that help organizations deliver more value while enjoying the journey
- Inspire others to improve their way of working by helping them invite new exciting relevant and useful ways of thinking and doing into their context in a way that focuses on value and is sticky and sustainable
- Public Training – Exposing the community to exciting new ways to improve work and creating the practitioner community to support introduction and usage of those approaches inside organizations
- Planning, Training, On the job coaching and supporting improvement programs

- Contact us at www.agilesparks.com
  info@agilesparks.com

AgileSparks