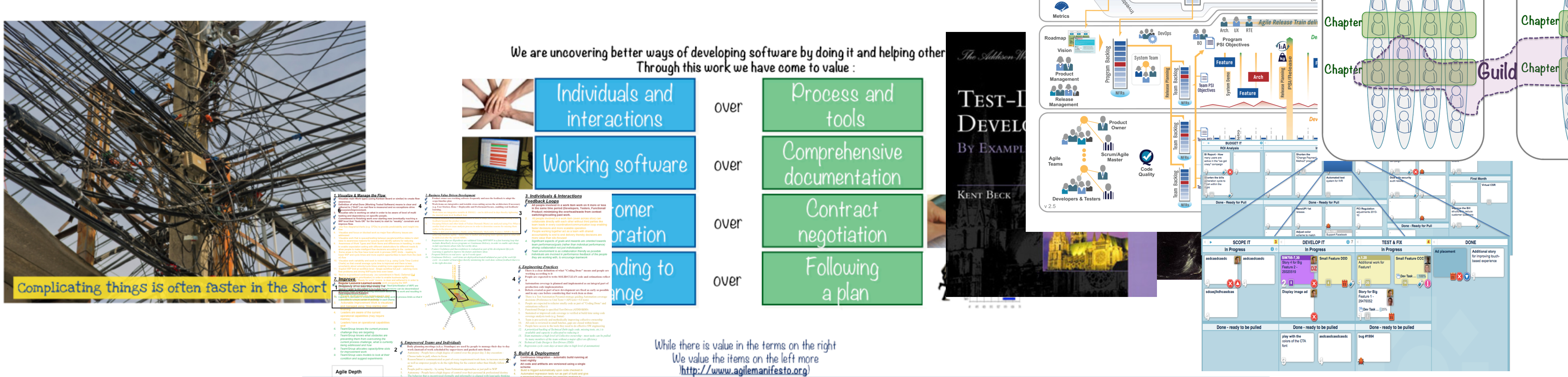


We are uncovering better ways of developing software by doing it and helping other
Through this work we have come to value :

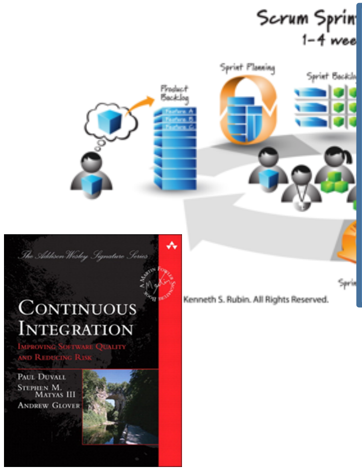
Individuals and interactions over Process and tools

Working software over Comprehensive documentation

omer oration over Contract negotiation

nding to ange over Following a plan

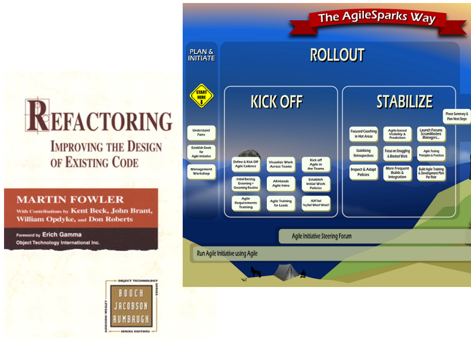
While there is value in the terms on the right
We value the items on the left more
http://www.agilemanifesto.org

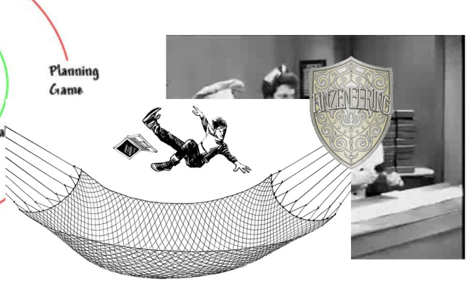


Introduction to Agile

Introduction to Agile - Yuval Yeret - Lead Coach @ AgileSparks

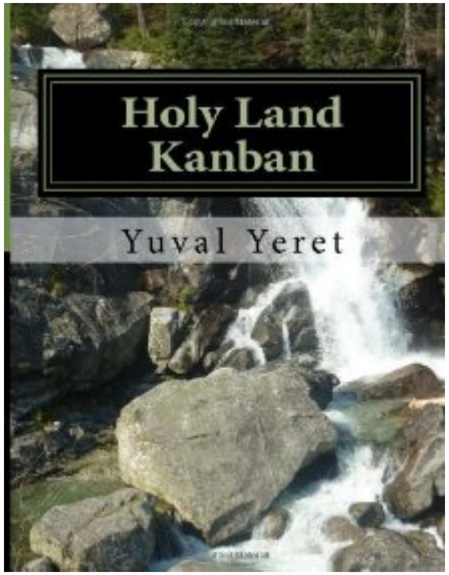
- Principles, history and mindset
- Individuals and interactions
- Value driven development
- User involvement
- Planning and adapting
- Agile Frameworks (Scrum, Kanban, XP)





About Me

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



- Speaking soon at:

Modern Management Methods
Lean Kanban North America
SPEAKER
SAN FRANCISCO • MAY 5-8, 201

2014 Agile Israel
אירוע ה-Agile המרכזי של ישראל

FELLOW
Lean Systems Society

What is Agile all about?

The business winds shift faster and stronger each year

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

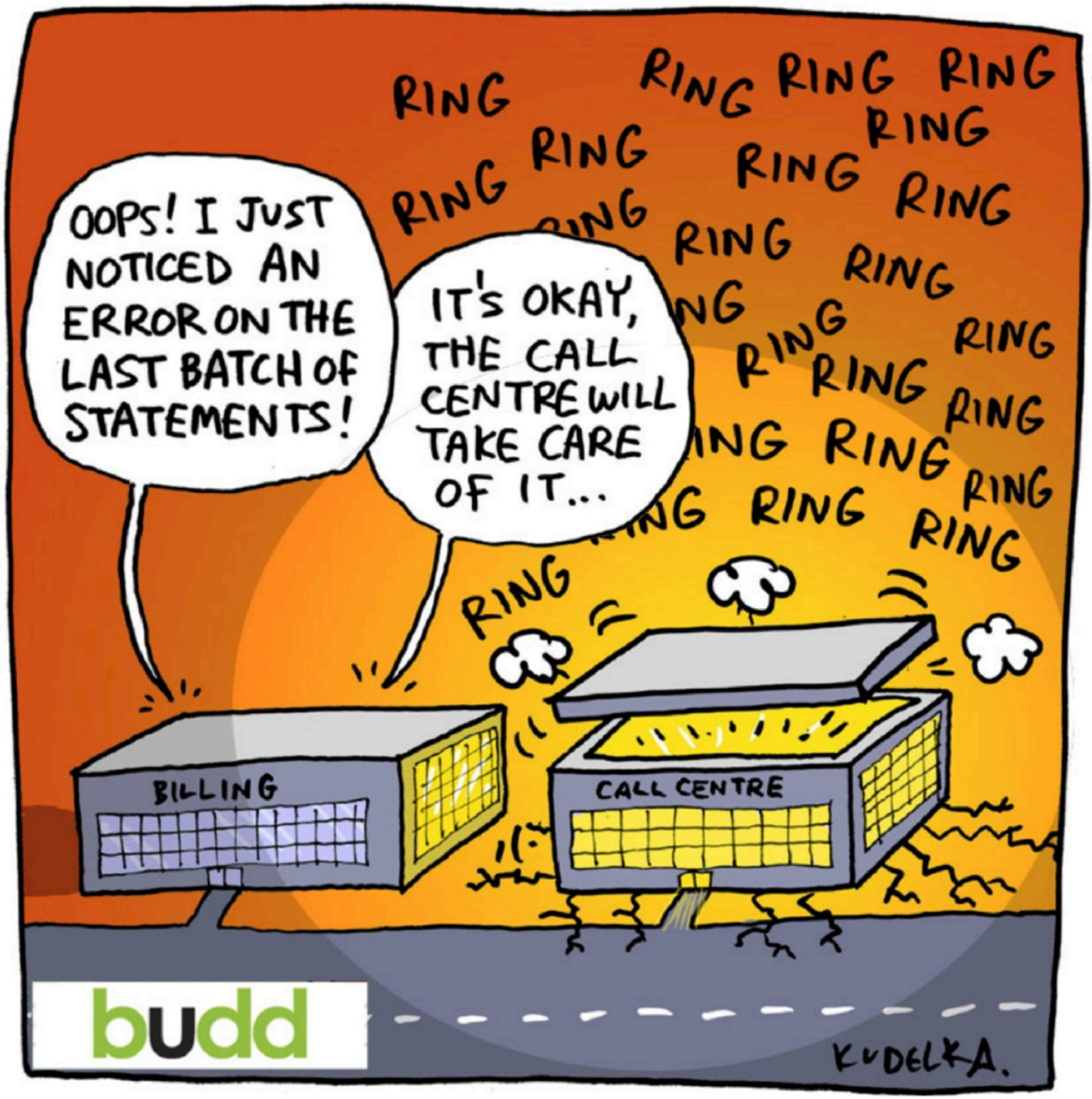
The IT Applications Development and Maintenance Expectations & Landscape

Deliver more with less

Achieve high uptimes with less

Complicating things is often faster in the short run

When do you celebrate/relax? Handoff or actual finish line?


OOPS! I JUST NOTICED AN ERROR ON THE LAST BATCH OF STATEMENTS!
IT'S OKAY, THE CALL CENTRE WILL TAKE CARE OF IT...
RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING
BILLING
CALL CENTRE
budd
KUDELKA.

http://design4services.com/concepts/failure-demand/

So don't be surprised if people optimize accordingly

A

Lucy & the Chocolate Factory

http://www.youtube.com/watch?v=FGfplQ1FUFs

So you spend most of the time FIGHTING fires

Familiar?

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

The Expectations

Deliver lots of value

Minimize disappointment/ surprises

Be Flexible Enough

Complicating things is often faster in the short run

When do you celebrate/relax? Handoff or actual finish line?

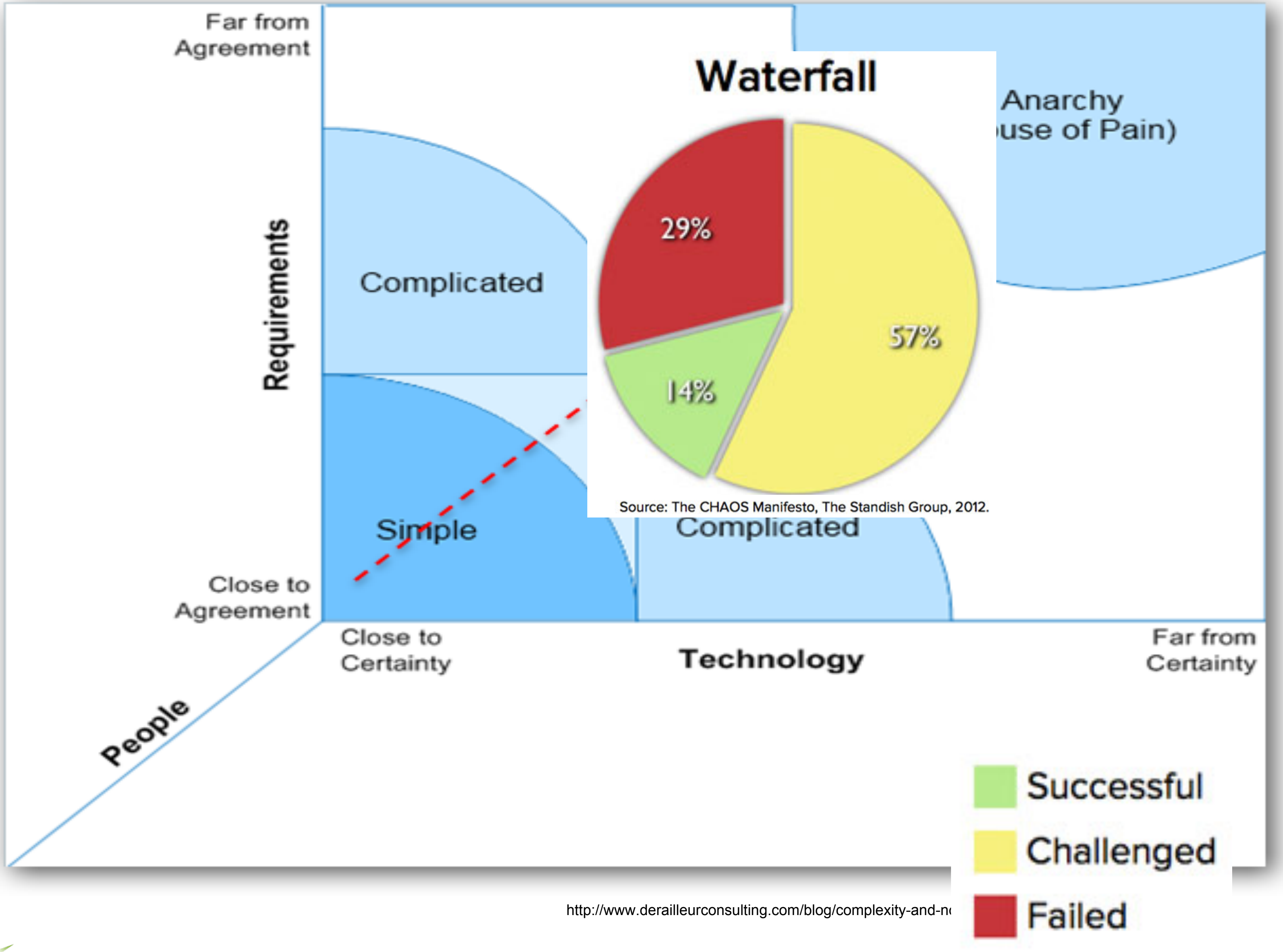
RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING RING
OOPS! I JUST NOTICED AN ERROR ON THE LAST BATCH OF STATEMENTS!
IT'S OKAY, THE CALL CENTRE WILL TAKE CARE OF IT...
BILLING
CALL CENTRE
budd

So don't be surprised if people optimize accordingly

So you spend most of the time FIGHTING fires

Harder and Harder to deliver

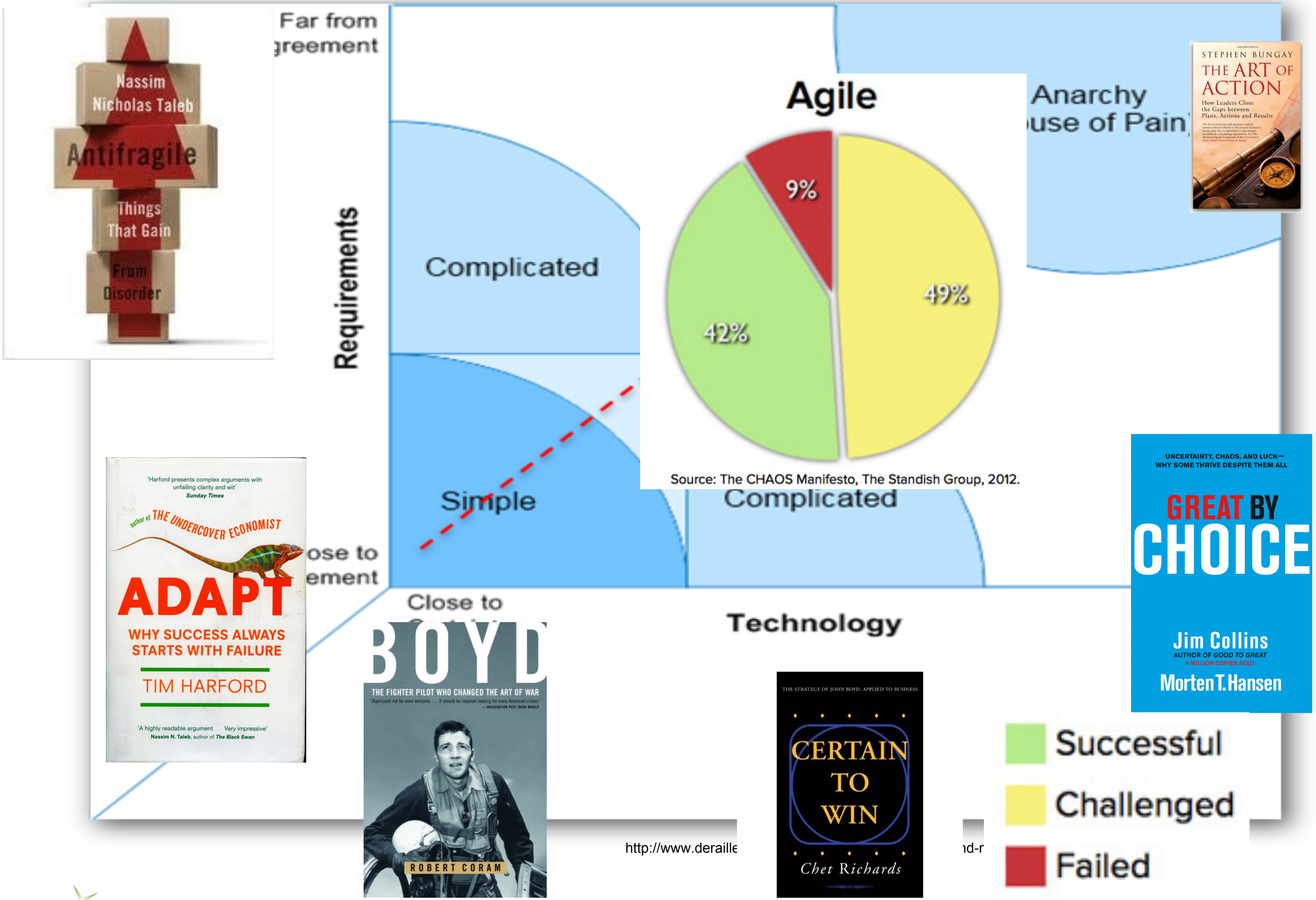
We live in uncertain times...



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

http://www.derailleurconsulting.com/blog/complexity-and-n

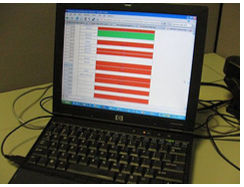
We need approaches that embrace uncertainty/complexity



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

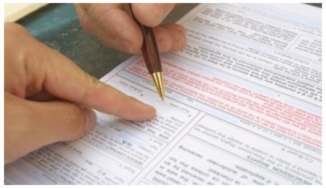
We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value :

Individuals and interactions over Process and tools

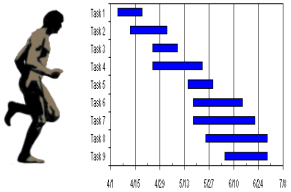


Working software over Comprehensive documentation

Customer collaboration over Contract negotiation



Responding to change over Following a plan

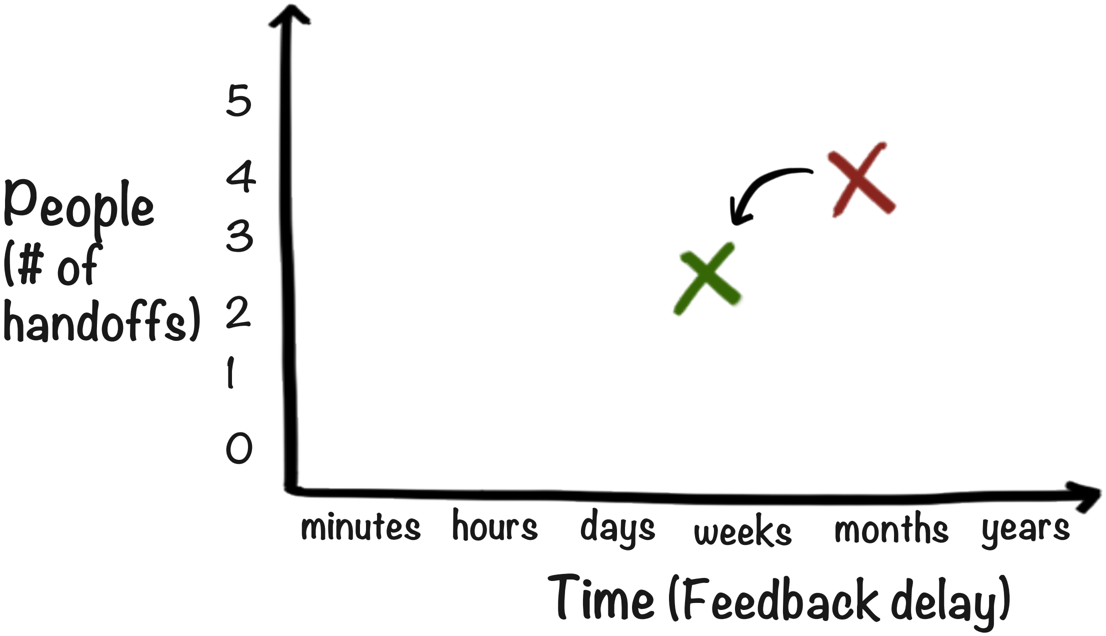


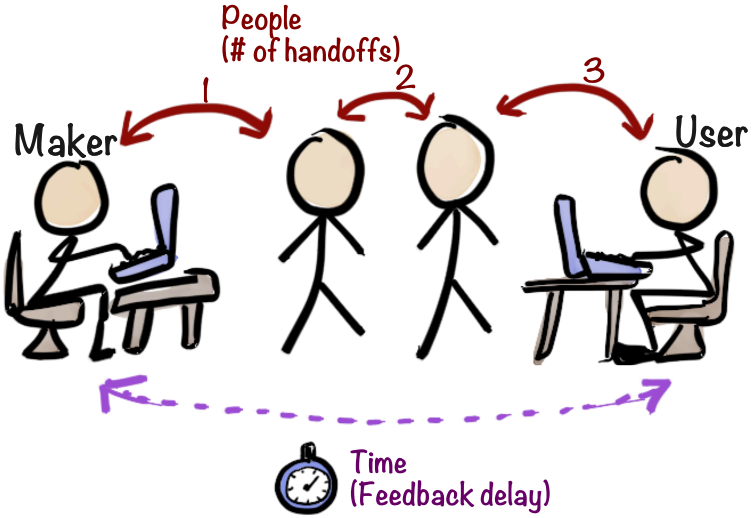
While there is value in the terms on the right
We value the items on the left more
)http://www.agilemanifesto.org)

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.

Minimize distance between Maker and User





Henrik Kniberg

AgileSparks

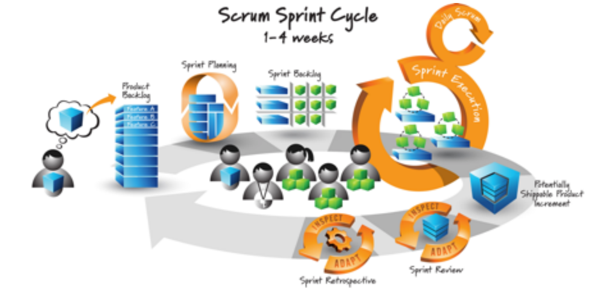
History of key Lean/Agile Frameworks

Kanban – 2010s

Evolve into Lean/ Agile

Focus on the flow

CONSUMER
CR TLG 249054 - Side Branch 13025 - Ban + Subscriber Cancellation Basic Flow



Copyright (c) 2010, Innolution, LLC & Kenneth S. Rubin. All Rights Reserved.

Scrum – 2000s

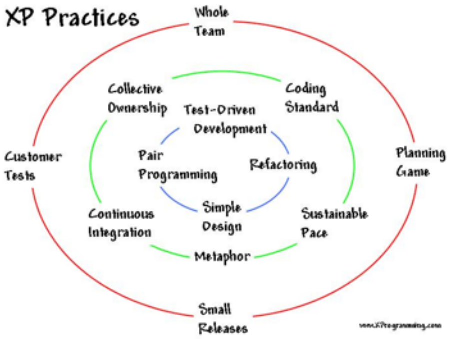
Jump into Lean/Agile

Still the leading classic agile approach

Extreme Programming – 90s

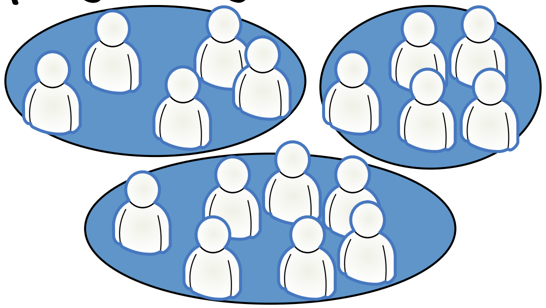
Lean/Agile Engineering practices

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

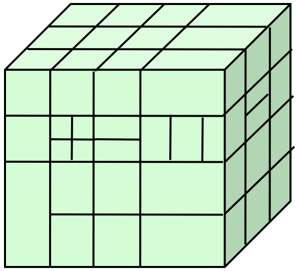


Scrum in a nutshell

Split your organization

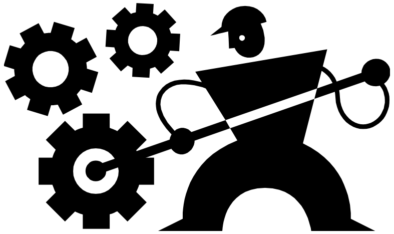


Split your product

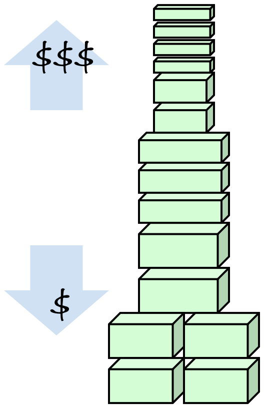


~~Large group spending a long time building a huge thing~~
Small team spending a little time building a small thing
… but integrating regularly to see the whole

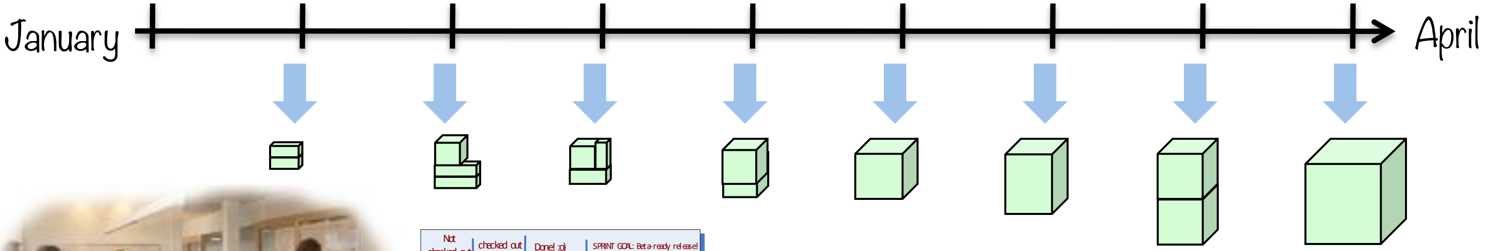
Optimize process

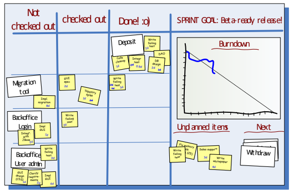


Optimize business value



Split time



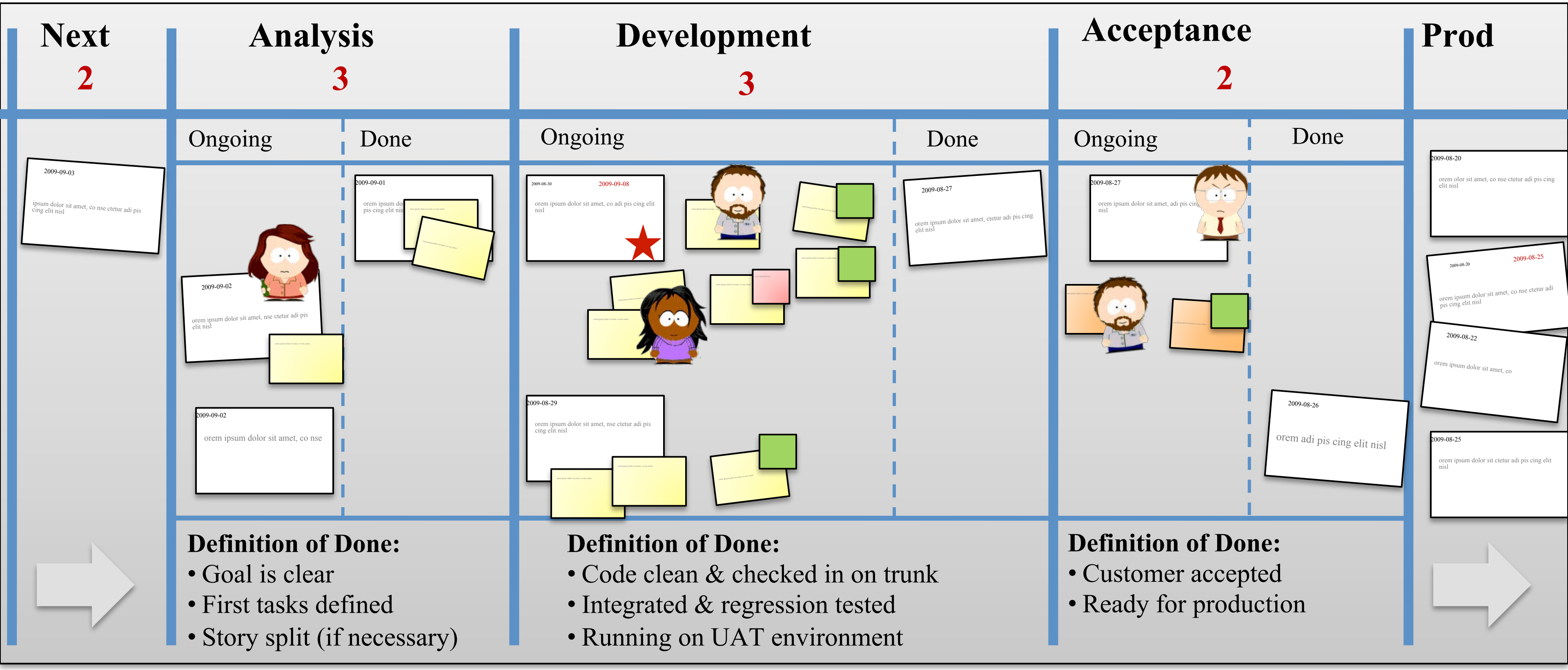


Henrik Kniberg

Kanban kick-start example

www.crisp.se/kanban/example

version 1.2
2009-11-16



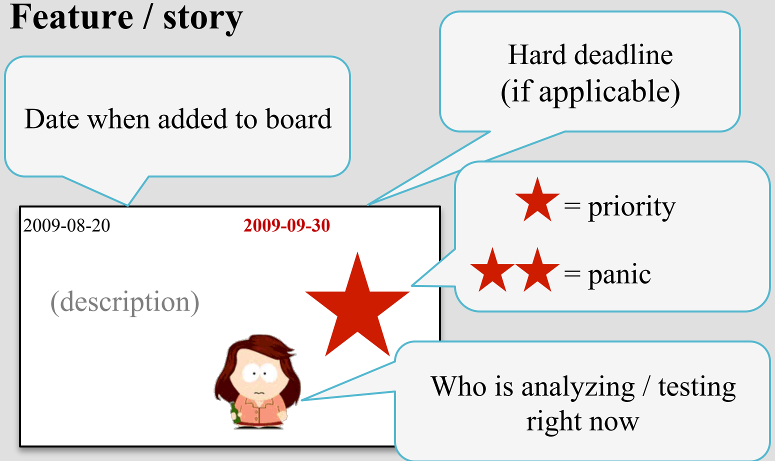
Next 2 | Analysis 3 (Ongoing, Done) | Development 3 (Ongoing, Done) | Acceptance 2 (Ongoing, Done) | Prod

Definition of Done:
- Goal is clear
- First tasks defined
- Story split (if necessary)

Definition of Done:
- Code clean & checked in on trunk
- Integrated & regression tested
- Running on UAT environment

Definition of Done:
- Customer accepted
- Ready for production

Feature / story



Date when added to board

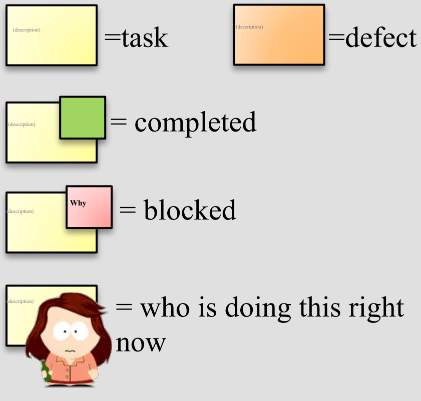
Hard deadline (if applicable)

★ = priority

★★ = panic

Who is analyzing / testing right now

Task / defect



=task

=defect

= completed

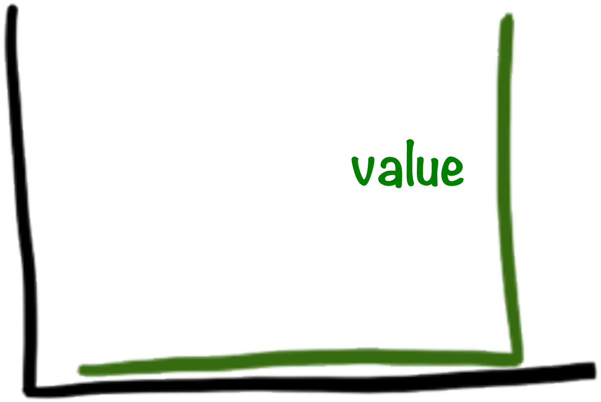
= blocked

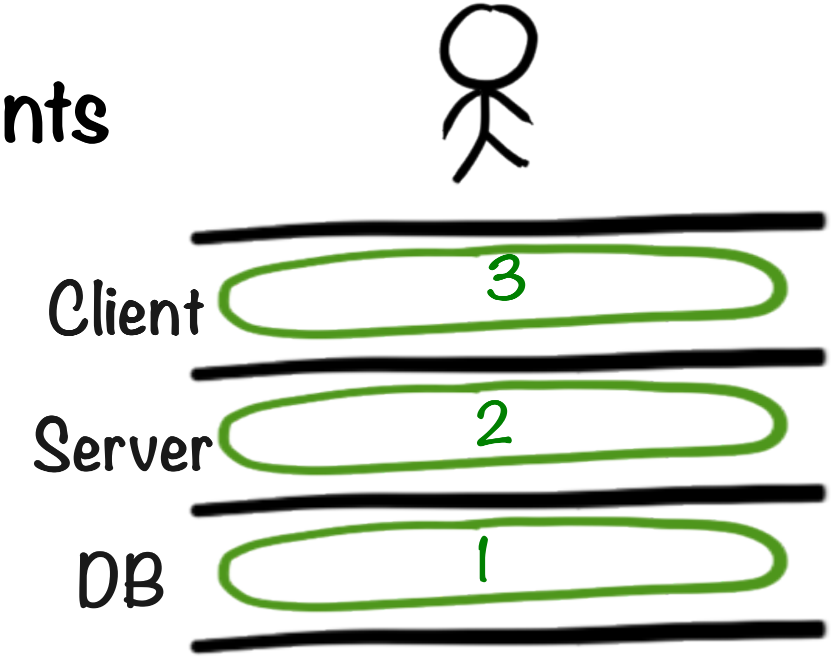
= who is doing this right now

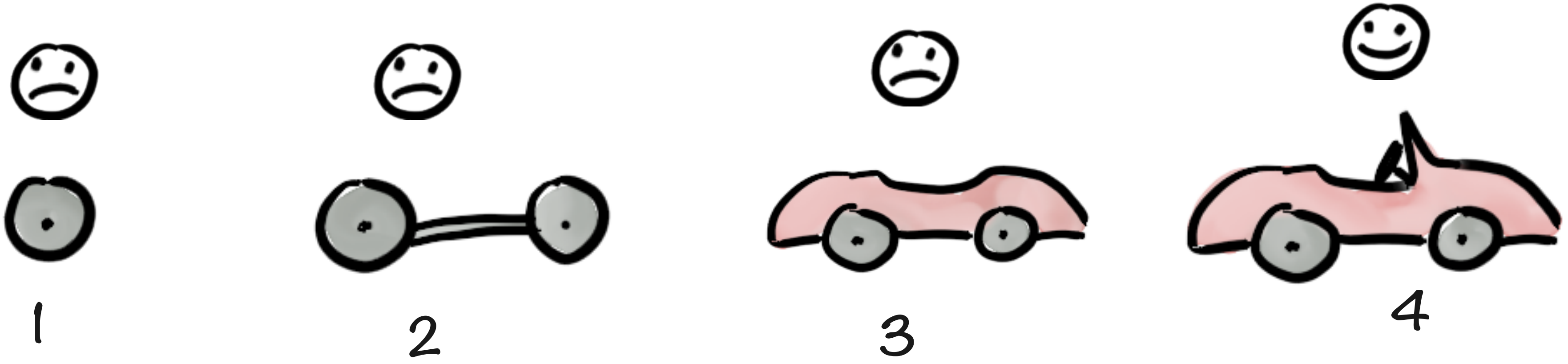
What to pull first

- **Panic** features ★★
(should be swarmed and kept moving. Interrupt other work and break WIP limits as necessary)
- **Priority** features
- Hard **deadline** features ★
(only if deadline is at risk)
- **Oldest** features

Not "horizontal" increments



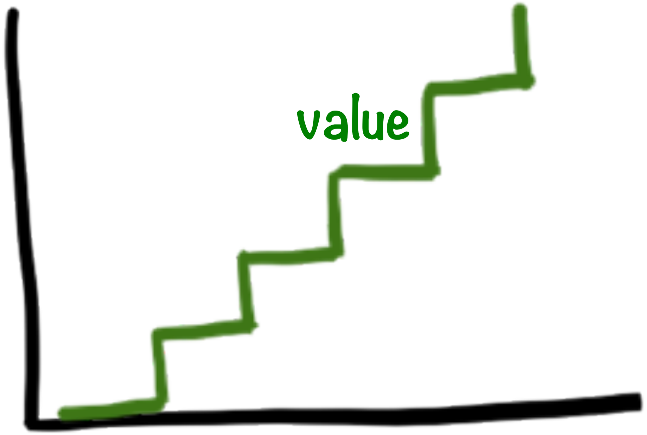


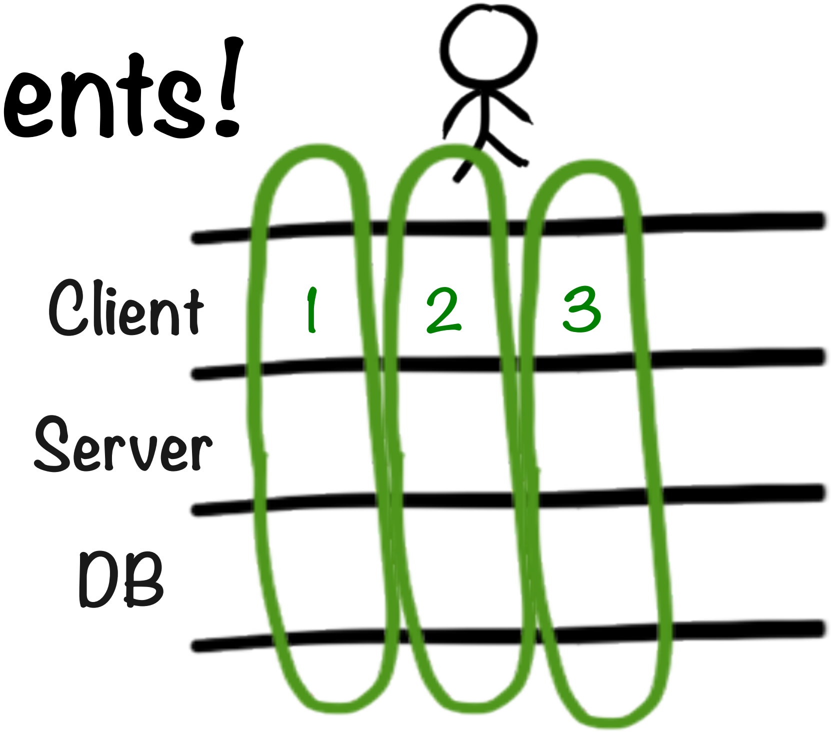


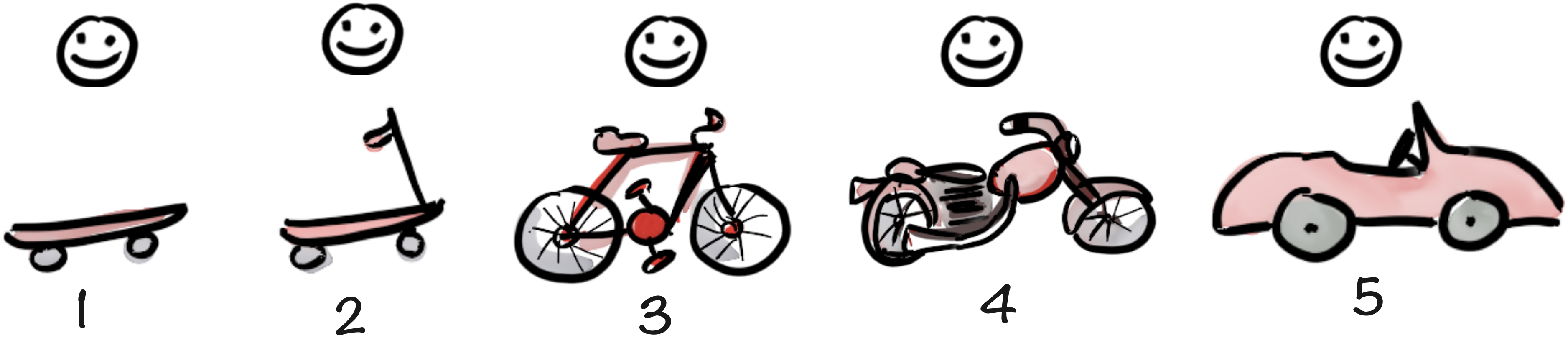
Henrik Kniberg

AgileSparks

”Vertical” increments!



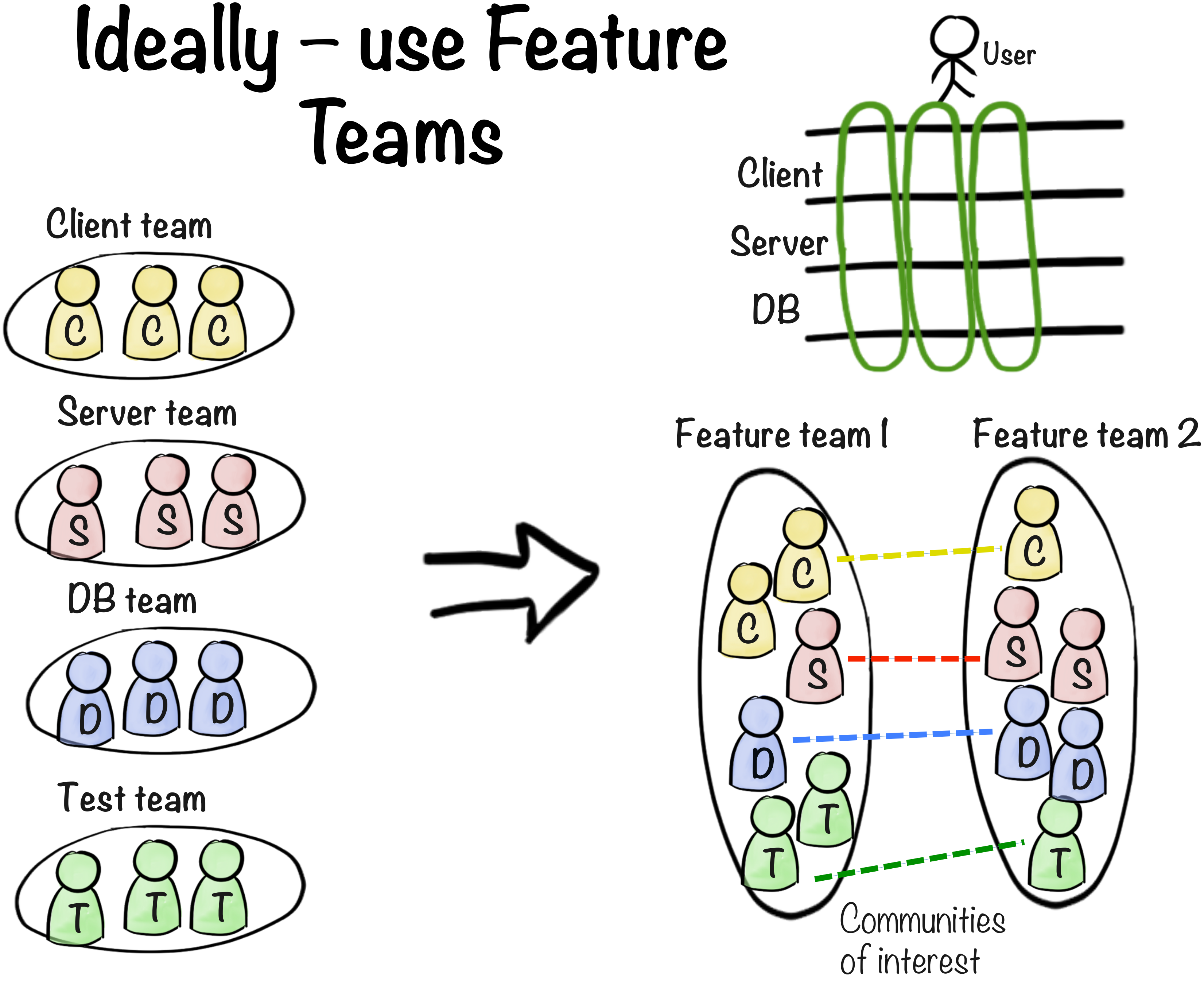




Henrik Kniberg

AgileSparks

Ideally – use Feature Teams



Henrik Kniberg

AgileSparks

The most important feedback loop:

THE RETROSPECTIVE

OWN YOUR PROCESS!!!

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”

Agile Manifesto principle

AgileSparks

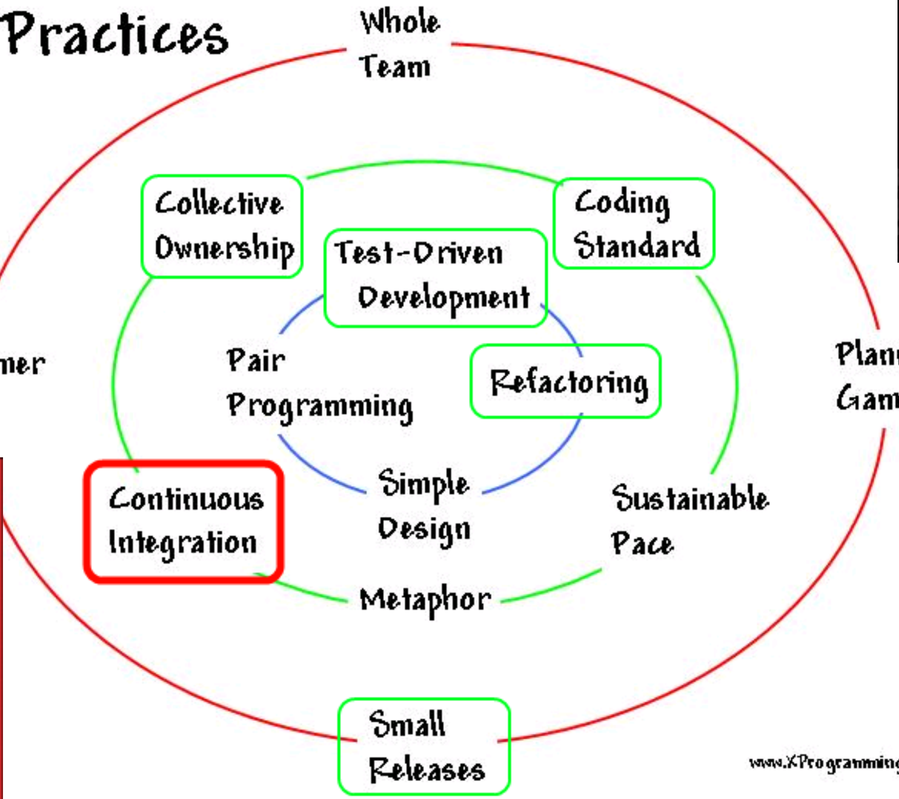
Changes that Agile will drive:

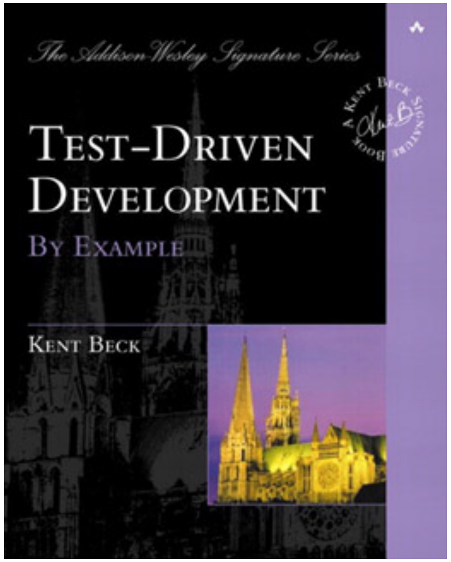
- Infrastructure Investments
  (release automation, test automation, etc)
- Evolve the organizational
  (new roles, cross-functional teams, etc)
- New skills
  (Vertical story-slicing, agile architecture, XP engineering practices, retrospectives, etc)
- New habits
  (Frequent customer interaction, frequent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

What will happen if we don't do this?

But no need to do it all at once

XP Practices

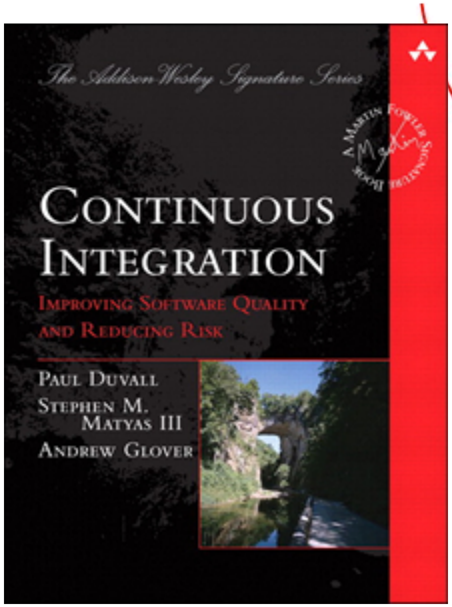



The Addison Wesley Signature Series

TEST-DRIVEN DEVELOPMENT

BY EXAMPLE

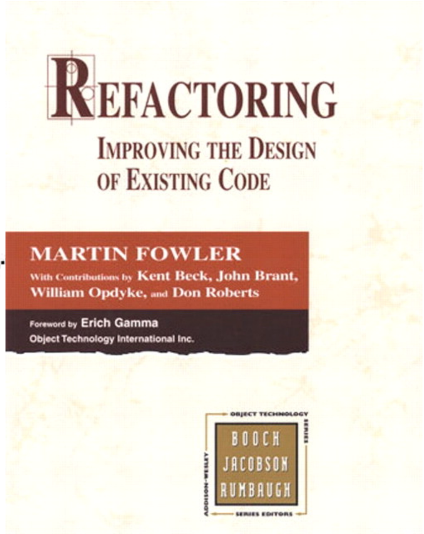
KENT BECK


The Addison Wesley Signature Series

CONTINUOUS INTEGRATION

IMPROVING SOFTWARE QUALITY AND REDUCING RISK

PAUL DUVALL
STEPHEN M. MATYAS III
ANDREW GLOVER


REFACTORING

IMPROVING THE DESIGN OF EXISTING CODE

MARTIN FOWLER
With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts

Foreword by Erich Gamma
Object Technology International Inc.

BOOCH JACOBSON RUMBAUGH

1. Visualize & Manage the Flow

4

6

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 not ready 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

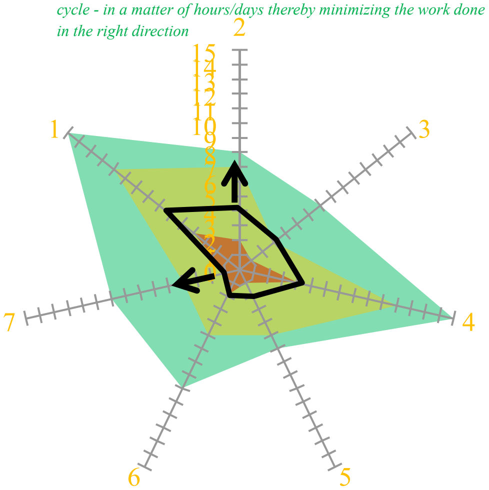
4

3

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

4

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. People are expected to refactor smelly code as part of "Coding Done" and estimations reflect it
7. Functional Design is specified Test-Driven (ATDD/BDD)
8. Sustained or improved code coverage is verified at build time using code coverage analysis tools (e.g. Sonar)
9. Team is pro-actively and methodically improving collective ownership
10. All code is reviewed in small batches, gaps are closed within hours
11. People have access to the tools they need to do effective SW engineering
12. *A prioritized backlog of Technical Debt (ugly code, missing tests, etc.) is available and capacity is allocated to reducing it*
13. *Team maintains a high level of Collective ownership - most tasks can be pulled by many members of the team without a major effect on efficiency*
14. *Technical Code Design is Test-Driven (TDD)*
15. *Regression cycle costs days at most (due to high level of automation)*

7. Improve

1

1. ✓ **Regular Lessons Learned events (frequency of no less than every 1-4 weeks) with actionable outcomes (e.g. Retrospectives/Kaizen)**
2. People at all levels are highly aware and involved in improvement activity
3. Actionable Improvement Work is visualized and managed using "Stop starting start 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*

6. Empowered Teams and Individuals

2

2

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 a
9. People are given opportunity to improve their mastery
10. People can shape their work environment – technologi

5. Build & Deployment

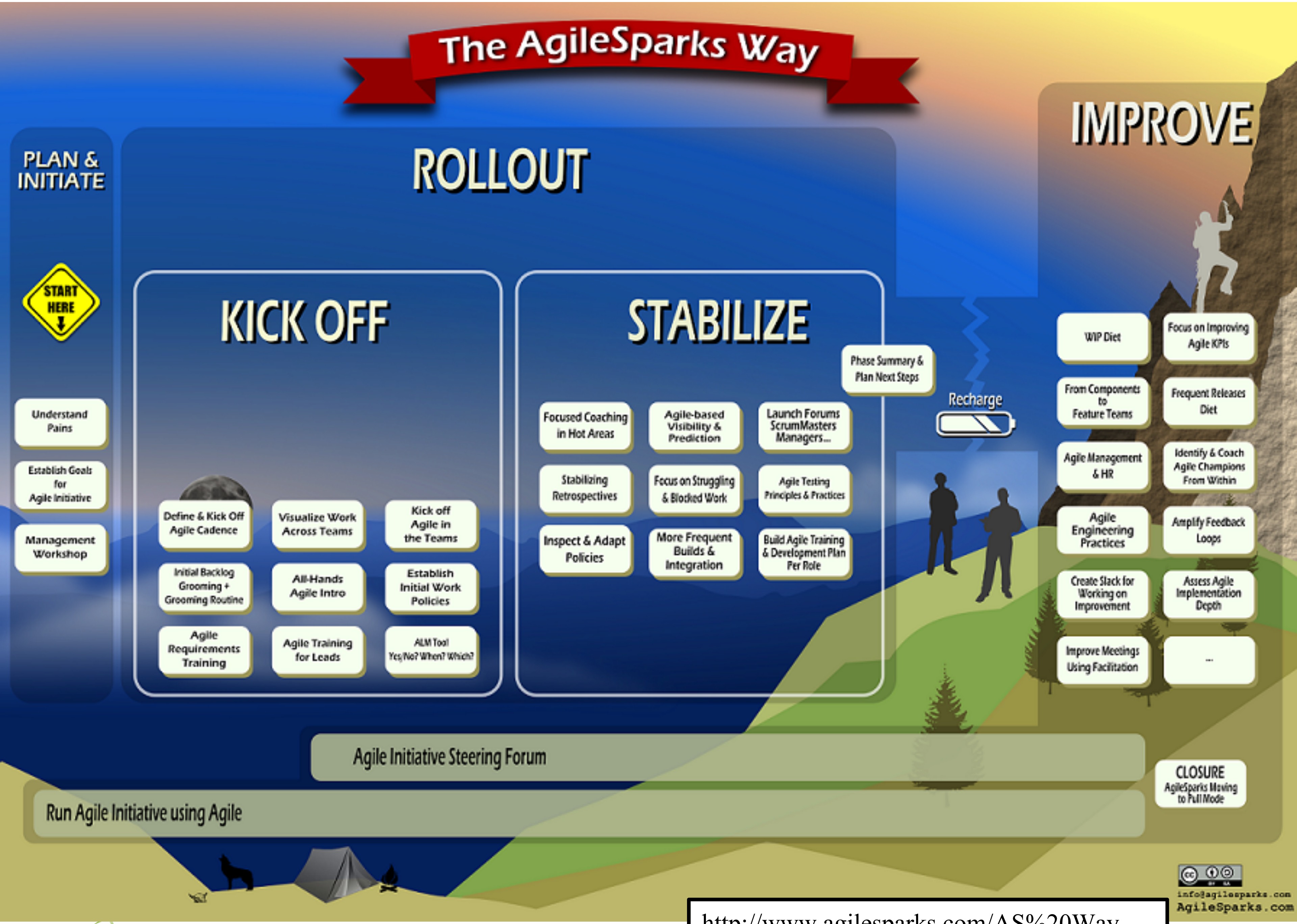
1. ✓ **Continuous Integration – automatic build running at least nightly**
2. ✓ **All code and artifacts are versioned using a single scheme**
3. Build is trigged automatically upon code checked in
4. Automated regression tests run as part of build and give a green/red binary answer (no need for analysis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

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

Agile Depth

Team: Sky1

Date: Sep 2013



The AgileSparks Way

PLAN & INITIATE

START HERE

- Understand Pains
- Establish Goals for Agile Initiative
- Management Workshop

ROLLOUT

KICK OFF

- Define & Kick Off Agile Cadence
- Visualize Work Across Teams
- Kick off Agile in the Teams
- Initial Backlog Grooming + Grooming Routine
- All-Hands Agile Intro
- Establish Initial Work Policies
- Agile Requirements Training
- Agile Training for Leads
- ALM Tool Yes/No? When? Which?

STABILIZE

- Phase Summary & Plan Next Steps
- Focused Coaching in Hot Areas
- Agile-based Visibility & Prediction
- Launch Forums ScrumMasters Managers...
- Stabilizing Retrospectives
- Focus on Struggling & Blocked Work
- Agile Testing Principles & Practices
- Inspect & Adapt Policies
- More Frequent Builds & Integration
- Build Agile Training & Development Plan Per Role

Recharge

IMPROVE

- WIP Diet
- Focus on Improving Agile KPIs
- From Components to Feature Teams
- Frequent Releases Diet
- Agile Management & HR
- Identify & Coach Agile Champions From Within
- Agile Engineering Practices
- Amplify Feedback Loops
- Create Slack for Working on Improvement
- Assess Agile Implementation Depth
- Improve Meetings Using Facilitation
- ...

Agile Initiative Steering Forum

Run Agile Initiative using Agile

CLOSURE
AgileSparks Moving to Pull Mode

info@agilesparks.com
AgileSparks.com

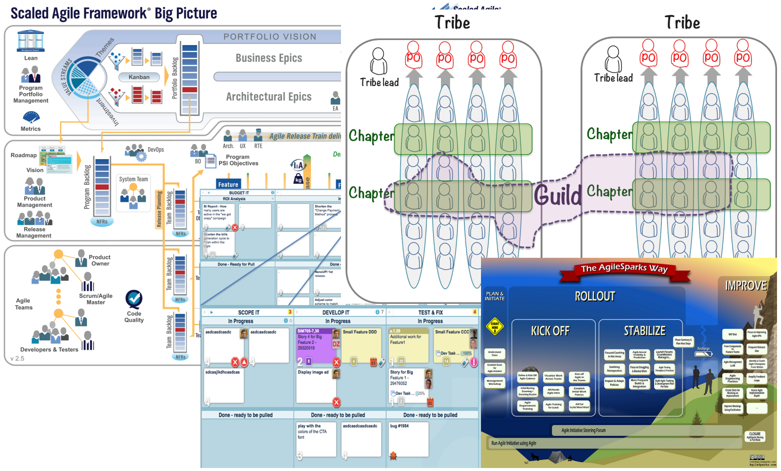
http://www.agilesparks.com/AS%20Way

AgileSparks

Current Trends – Spring 2014...

DevOps

Aiming to break the legacy conflict between Dev & Ops by leveraging lean/agile thinking/practices as well as strong automation-focused tooling

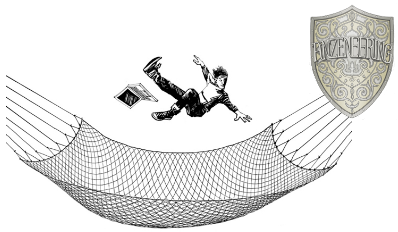


Patterns for Scaling Agile

Frameworks enabling agility from demand to delivery, across programs and portfolios – based on experiences scaling agile in recent years

Safety/Anzeneering

Protecting people as a core focus area to drive cultural change - **Protecting people underlies every Lean or Agile practice.** Reduce Technical Debt, Sustainable Pace, Empower people, etc.



AgileSparks

What is DevOps?

- Working together across Silos with the same goals and same measures of success/failure
- Providing modern working tools that enable smooth collaboration and frequent delivery of value all the way from ideation to production and operation
- If Agile is about better collaboration and flow between Business and the Dev side of IT, DevOps extends that to the Ops side as well.
- You will hear terms like Continuous Delivery/ Deployment, Infrastructure as Code, Environments as a service

Tech Safety

consider these common injuries to software makers:

- **Alteration Anxiety**: apprehensive uneasiness associated with making changes
- **Antique Agony**: mental anguish from working with old technology
- **Brain Hernia**: straining your brain to understand code with high conceptual weight
- **Browser Bruise**: pain caused by the blow of a browser bug
- **Bug Burn**: feeling burned by a defect, particularly one that injured users
- **Fractured Flow**: feeling interrupted causing an inability to focus
- **Fragility Frustration**: dissatisfaction with that which is easily and perpetually broken
- **Merge Misery**: suffering caused by difficult merges of code
- **Outage Ordeal**: severe stress caused by a major failure or interruption
- **Release Rage**: exhausting, manual release to production that robs one of family time, sleep, joy
- **Schedule Stress**: tension associated with a deadline

http://www.industriallogic.com/blog/techsafety/

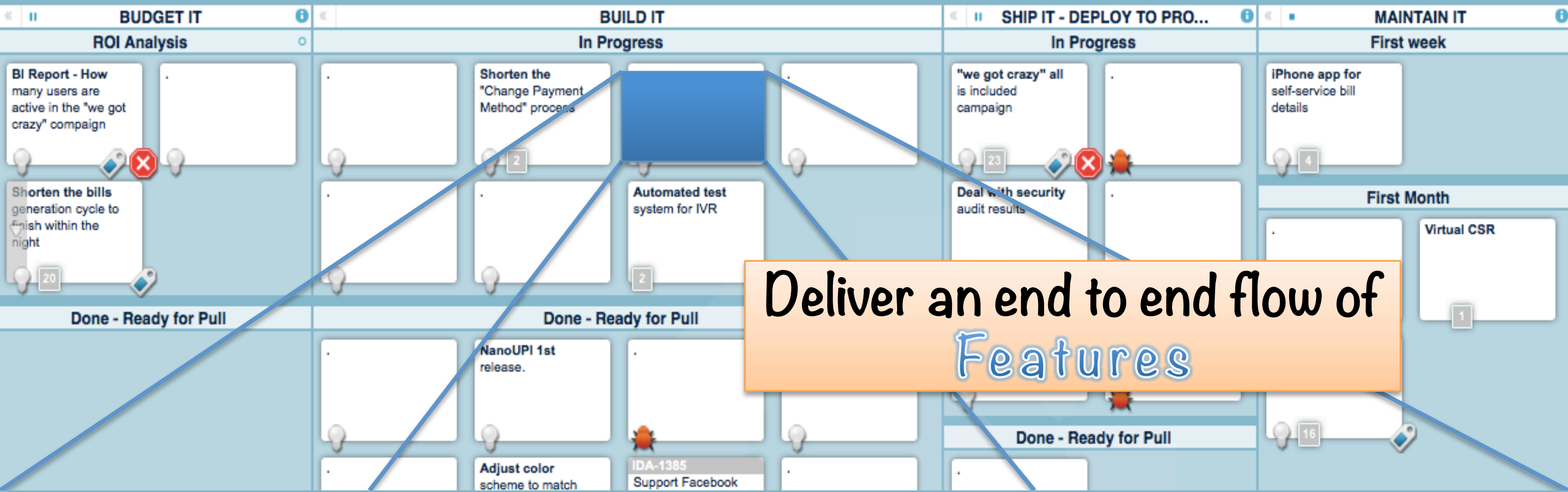
Anzeneering???

- Knowledge Work Lean/Agile
  - Kanban protects us from bottlenecks and decreased flow via visualized work, limited work-in-process and classes of service.
  - Scrum Teams w/ Self Organization - Effective Delegation / Decentralized Control protect us from micro-management and frustration of being stuck waiting for answers
  - Sustainable pace protects us from burnout, poor health and isolation.
  - Retrospectives protect us from repeating the same mistakes.
- Product Development / R&D Lean Agile Specifically
  - Continuous deployment protects us from stressful, error-prone releases while enabling safe, high-speed production improvements.
  - Extreme Programming's technical practices protect us from complexity, stress and defects via simple design, automated testing, continuous builds, test-driven development, refactoring and pair-programming.
  - Lean UX protects us from poor user experiences via interaction design and usability evaluations.
  - Lean Startups protect our time and money via minimum viable products/ features, validated learning and innovation accounting.

http://www.industriallogic.com/blog/anzeneering/

Some scaling examples

Inbar will talk about SAFe as an alternative



BUDGET IT

ROI Analysis

BI Report - How many users are active in the "we got crazy" compaign

.

Shorten the bills generation cycle to finish within the night

Done - Ready for Pull

BUILD IT

In Progress

.

Shorten the "Change Payment Method" process

.

.

.

Automated test system for IVR

Done - Ready for Pull

.

NanoUPI 1st release.

.

.

Adjust color scheme to match

IDA-1385 Support Facebook

.

SHIP IT - DEPLOY TO PRO...

In Progress

"we got crazy" all is included campaign

.

Deal with security audit results

.

Done - Ready for Pull

.

MAINTAIN IT

First week

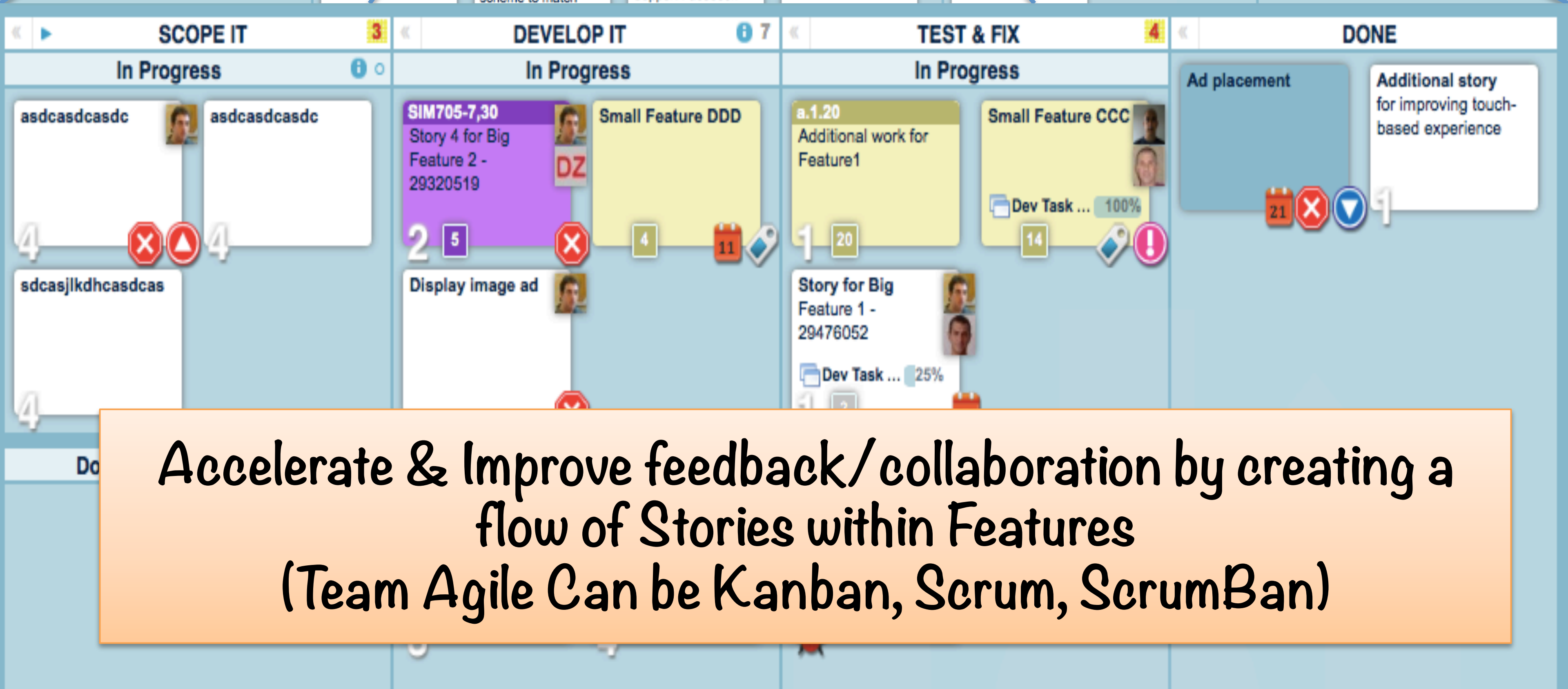
iPhone app for self-service bill details

First Month

.

Virtual CSR

Deliver an end to end flow of Features



SCOPE IT 3

In Progress

asdcasdcasdc

asdcasdcasdc

sdcasjlkdhcasdcas

DEVELOP IT 7

In Progress

SIM705-7,30 Story 4 for Big Feature 2 - 29320519

Small Feature DDD

Display image ad

TEST & FIX 4

In Progress

a.1.20 Additional work for Feature1

Small Feature CCC
Dev Task ... 100%

Story for Big Feature 1 - 29476052
Dev Task ... 25%

DONE

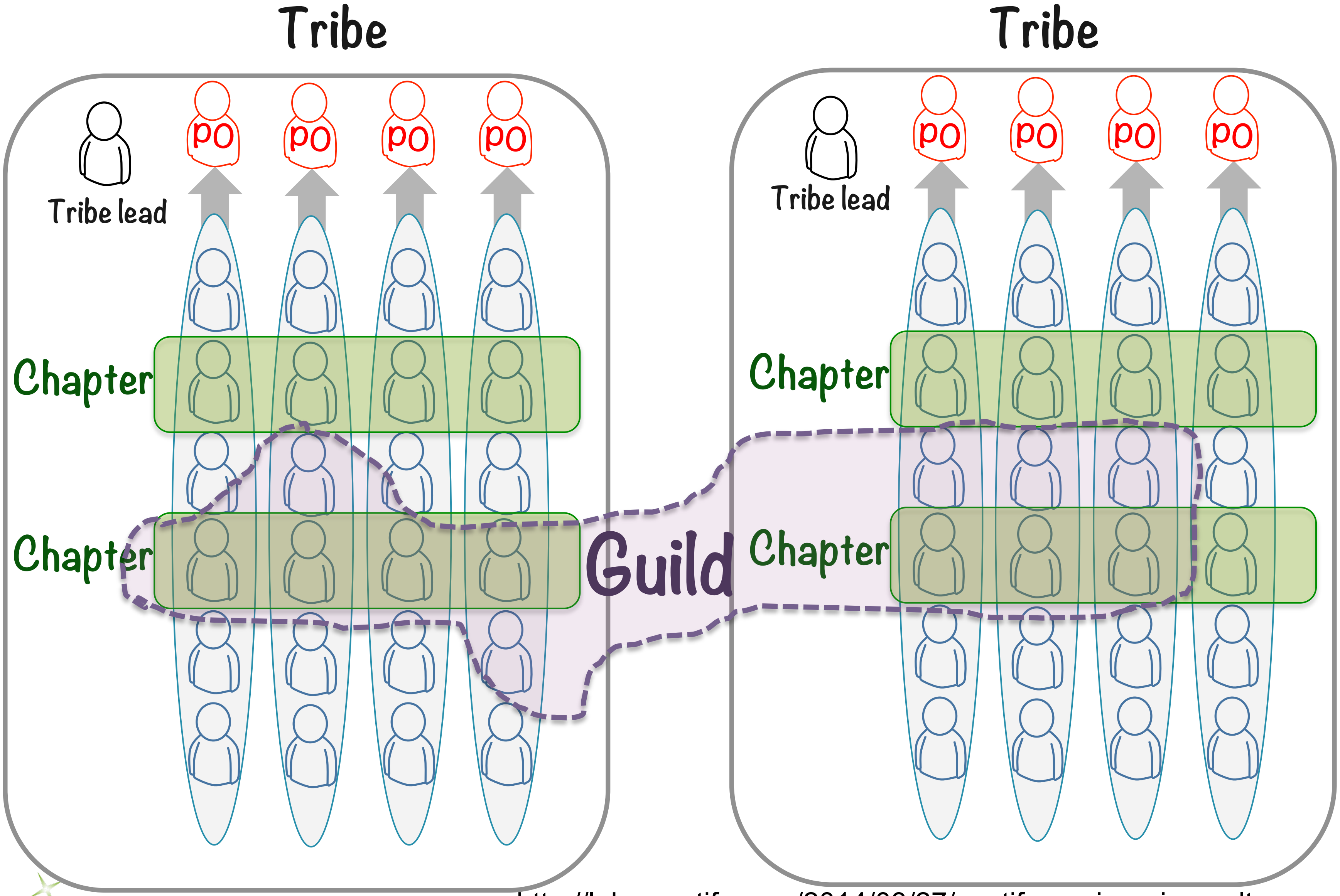
Ad placement

Additional story for improving touch-based experience

Do

Accelerate & Improve feedback/collaboration by creating a flow of Stories within Features
(Team Agile Can be Kanban, Scrum, ScrumBan)

Agile Teams at Scale – The Spotify Model



AgileSparks

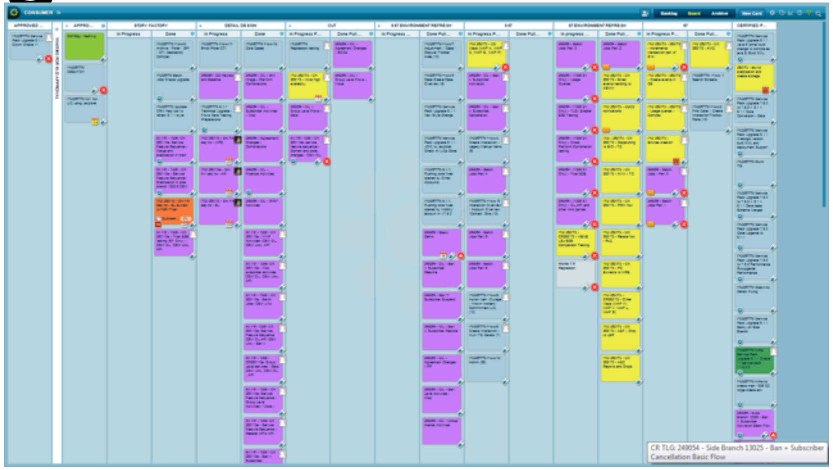
http://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/

key Lean/Agile Frameworks

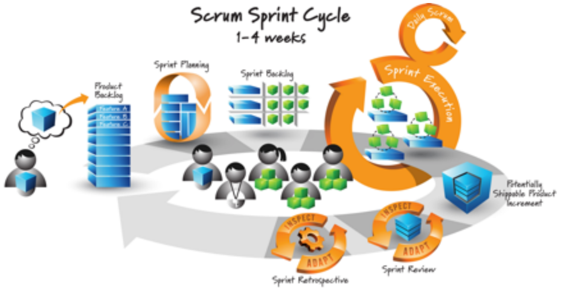
Kanban – 2010s

Evolve into Lean/Agile

Focus on the flow



CR TLG: 249054 - Side Branch 13025 - Ban + Subscriber Cancellation Basic Flow



Scrum – 2000s

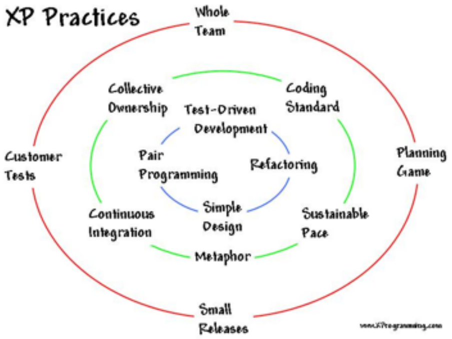
Jump into Lean/Agile

Still the leading classic agile approach

Extreme Programming – 90s

Lean/Agile Engineering practices

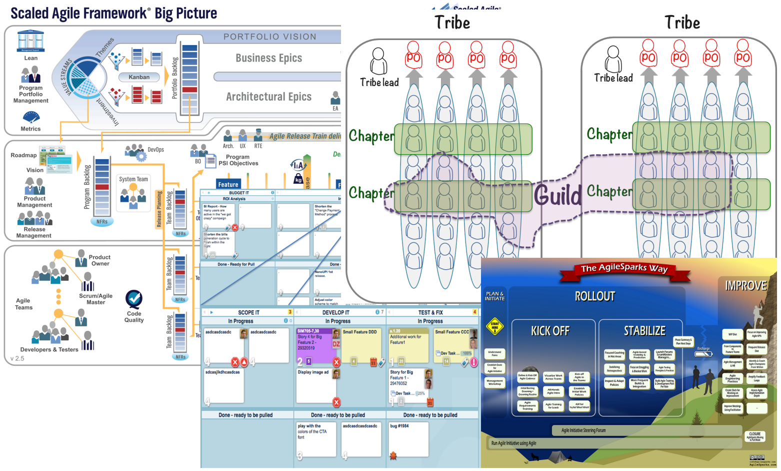
Bring in the right practices at the right time, but ignore at your own risk!



Current Trends – Spring 2014...

DevOps

Aiming to break the legacy conflict between Dev & Ops by leveraging lean/agile thinking/practices as well as strong automation-focused tooling

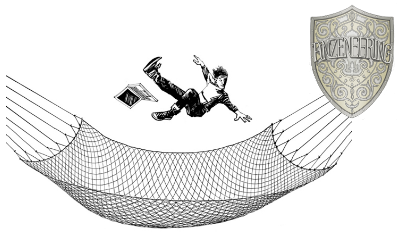


Patterns for Scaling Agile

Frameworks enabling agility from demand to delivery, across programs and portfolios – based on experiences scaling agile in recent years

Safety/Anzeneering

Protecting people as a core focus area to drive cultural change - **Protecting people underlies every Lean or Agile practice.** Reduce Technical Debt, Sustainable Pace, Empower people, etc.



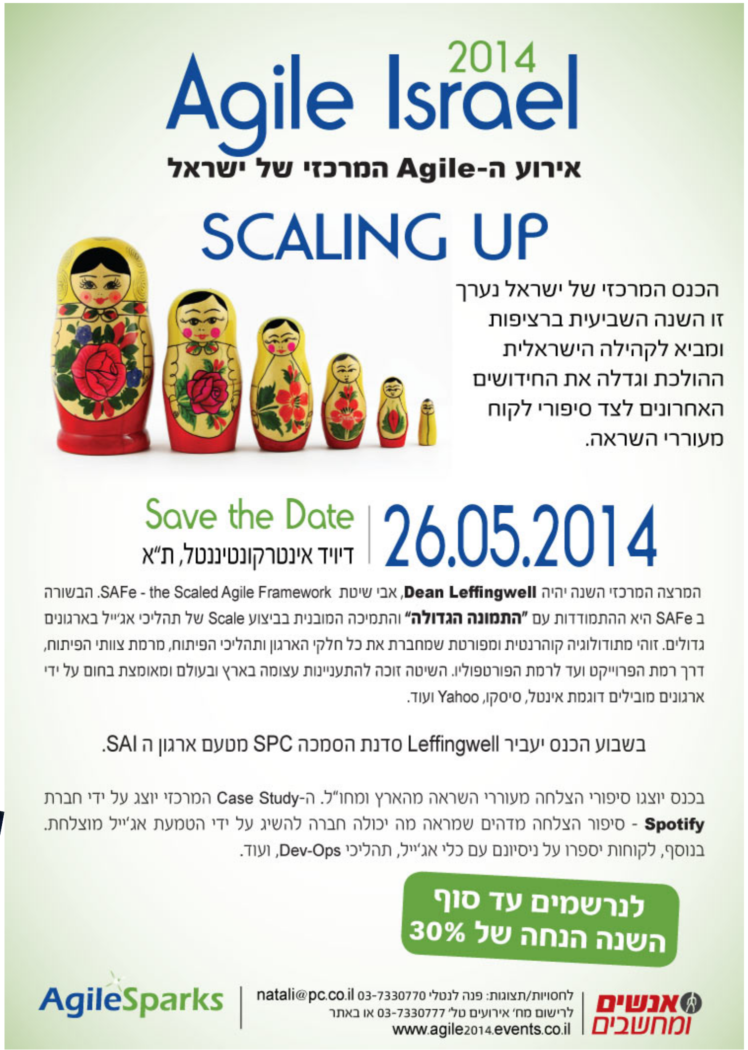
Summary

- Understand WHY you want to be more agile, rally your people around this reason and INVITE them to join you on the journey
  - Deal with Stress? Improve Safety? Business Agility? Improve productivity? Quality?
- Understand WHAT are your options
  - Scrum? Kanban? SAFe? Revolution? Evolution? Start with engineering practices? Add them on the way?
- Plan to learn and adapt along the way – use agile to become agile
- Relatively simple to understand, Non-trivial to practice.

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



2014 Agile Israel

אירוע ה-Agile המרכזי של ישראל

SCALING UP

הכנס המרכזי של ישראל נערך
זו השנה השביעית ברציפות
ומביא לקהילה הישראלית
ההולכת וגדלה את החידושים
האחרונים לצד סיפורי לקוח
מעוררי השראה.

Save the Date | 26.05.2014
דיויד אינטרקונטיננטל, ת"א

המרצה המרכזי השנה יהיה **Dean Leffingwell**, אבי שיטת SAFe - the Scaled Agile Framework. הבשורה ב SAFe היא ההתמודדות עם **"התמונה הגדולה"** והתמיכה המובנית בביצוע Scale של תהליכי אג'ייל בארגונים גדולים. זוהי מתודולוגיה קוהרנטית ומפורטת שמחברת את כל חלקי הארגון ותהליכי הפיתוח, מרמת צוותי הפיתוח, דרך רמת הפרוייקט ועד לרמת הפורטפוליו. השיטה זוכה להתעניינות עצומה בארץ ובעולם ומאומצת בחום על ידי ארגונים מובילים דוגמת אינטל, סיסקו, Yahoo ועוד.

בשבוע הכנס יעביר Leffingwell סדנת הסמכה SPC מטעם ארגון ה SAI.

בכנס יוצגו סיפורי הצלחה מעוררי השראה מהארץ ומחו"ל. ה-Case Study המרכזי יוצג על ידי חברת **Spotify** - סיפור הצלחה מדהים שמראה מה יכולה חברה להשיג על ידי הטמעת אג'ייל מוצלחת. בנוסף, לקוחות יספרו על ניסיונם עם כלי אג'ייל, תהליכי Dev-Ops, ועוד.

לנרשמים עד סוף
השנה הנחה של 30%

AgileSparks | לחסויות/תצוגות: פנה לנטלי 03-7330770 natali@pc.co.il
לרישום מח' אירועים טל' 03-7330777 או באתר
www.agile2014.events.co.il | אנשים ומחשבים

AgileSparks