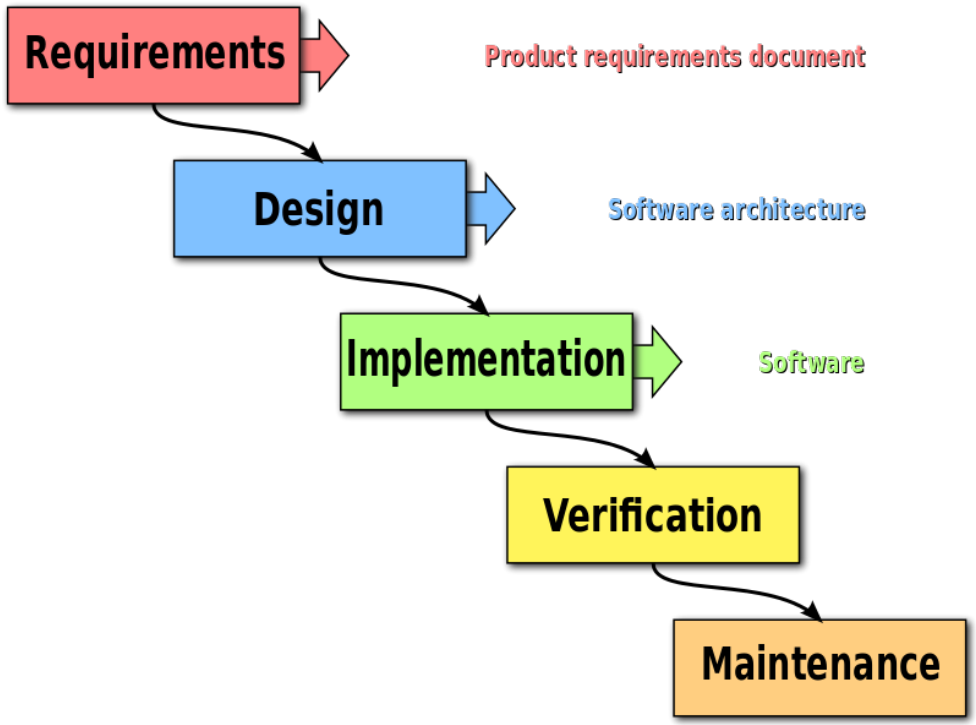
Agile Options

Too Many

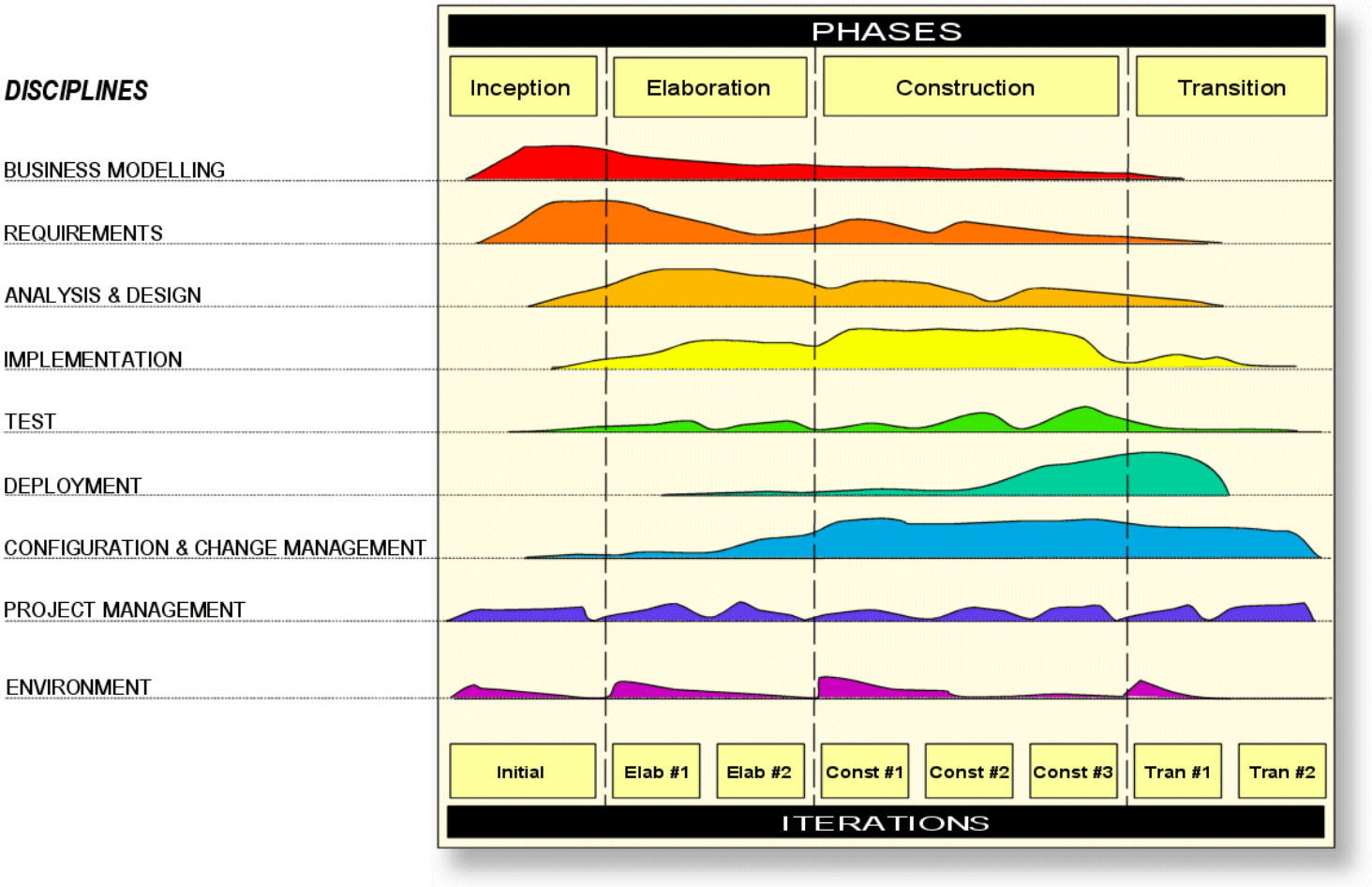
Methodologies

Waterfall



http://en.wikipedia.org/wiki/Waterfall_model

RUP



http://en.wikipedia.org/wiki/Rational_Unified_Process

Rational Unified Process

Although the Rational Unified Process (RUP) is independent of the UML, the two are often talked about together. So I think it's worth saying a few things about it here.

Although RUP is called a process, it actually is a process framework, providing a vocabulary and loose structure to talk about processes. When you use RUP, the first thing you need to do is choose a **development case**: the process you are going to use in the project. Development cases can vary widely, so don't assume that your development case will look that much like any other development case. Choosing a development case needs someone early on who is very familiar with RUP: someone who can tailor RUP for a particular project's needs. Alternatively, there is a growing body of packaged development cases to start from.

Whatever the development case, RUP is essentially an iterative process. A waterfall style isn't compatible with the philosophy of RUP, although sadly it's not uncommon to run into projects that use a waterfall-style process and dress it up in RUP's clothes.

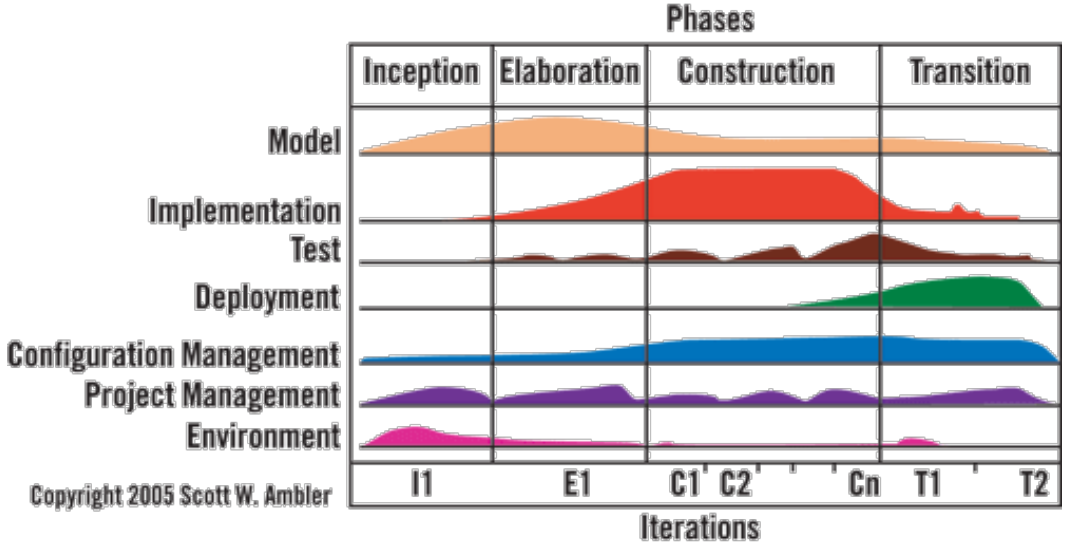
All RUP projects should follow four phases.

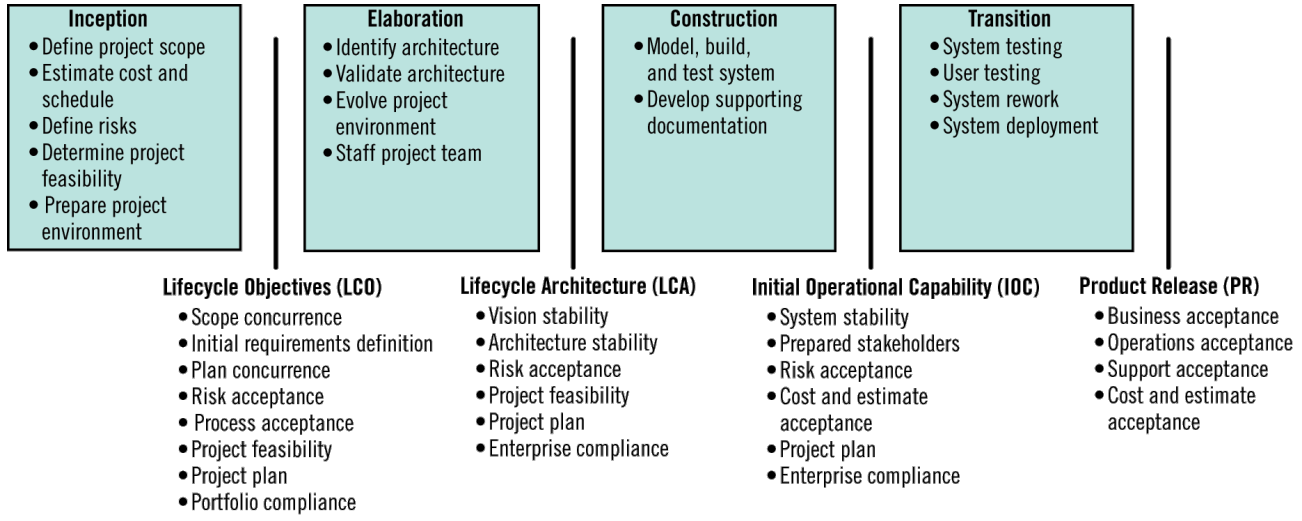
1. **Inception** makes an initial evaluation of a project. Typically in inception, you decide whether to commit enough funds to do an elaboration phase.
2. **Elaboration** identifies the primary use cases of the project and builds software in iterations in order to shake out the architecture of the system. At the end of elaboration, you should have a good sense of the requirements and a skeletal working system that acts as the seed of development. In particular, you should have found and resolved the major risks to the project.
3. **Construction** continues the building process, developing enough functionality to release.
4. **Transition** includes various late-stage activities that you don't do iteratively. These may include deployment into the data center, user training, and the like.

There's a fair amount of fuzziness between the phases, especially between elaboration and construction. For some, the shift to construction is the point at which you can move into a predictive planning mode. For others, it merely indicates the point at which you have a broad vision of requirements and an architecture that you think is going to last the rest of the project.

Sometimes, RUP is referred to as the Unified Process (UP). This is usually done by organizations that wish to use the terminology and overall style of RUP without using the licensed products of Rational Software. You can think of RUP as Rational's product offering based on the UP, or you can think of RUP and UP as the same thing. Either way, you'll find people who agree with you.

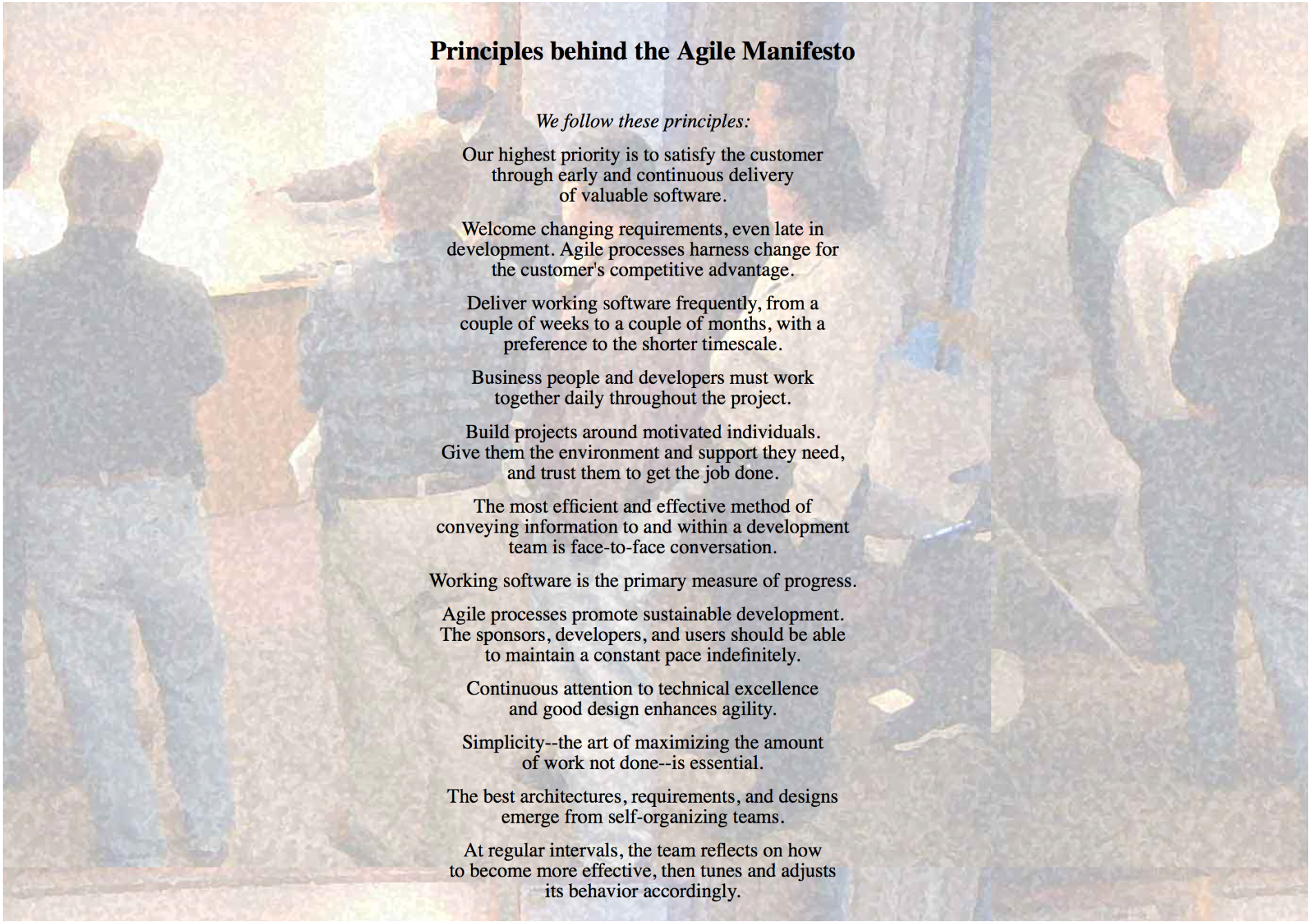
Agile UP





http://www.drdobbs.com/the-agile-edge-unified-and-agile/184415460

http://www.agilemanifesto.org/



Principles behind the Agile Manifesto

We follow these principles:

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.

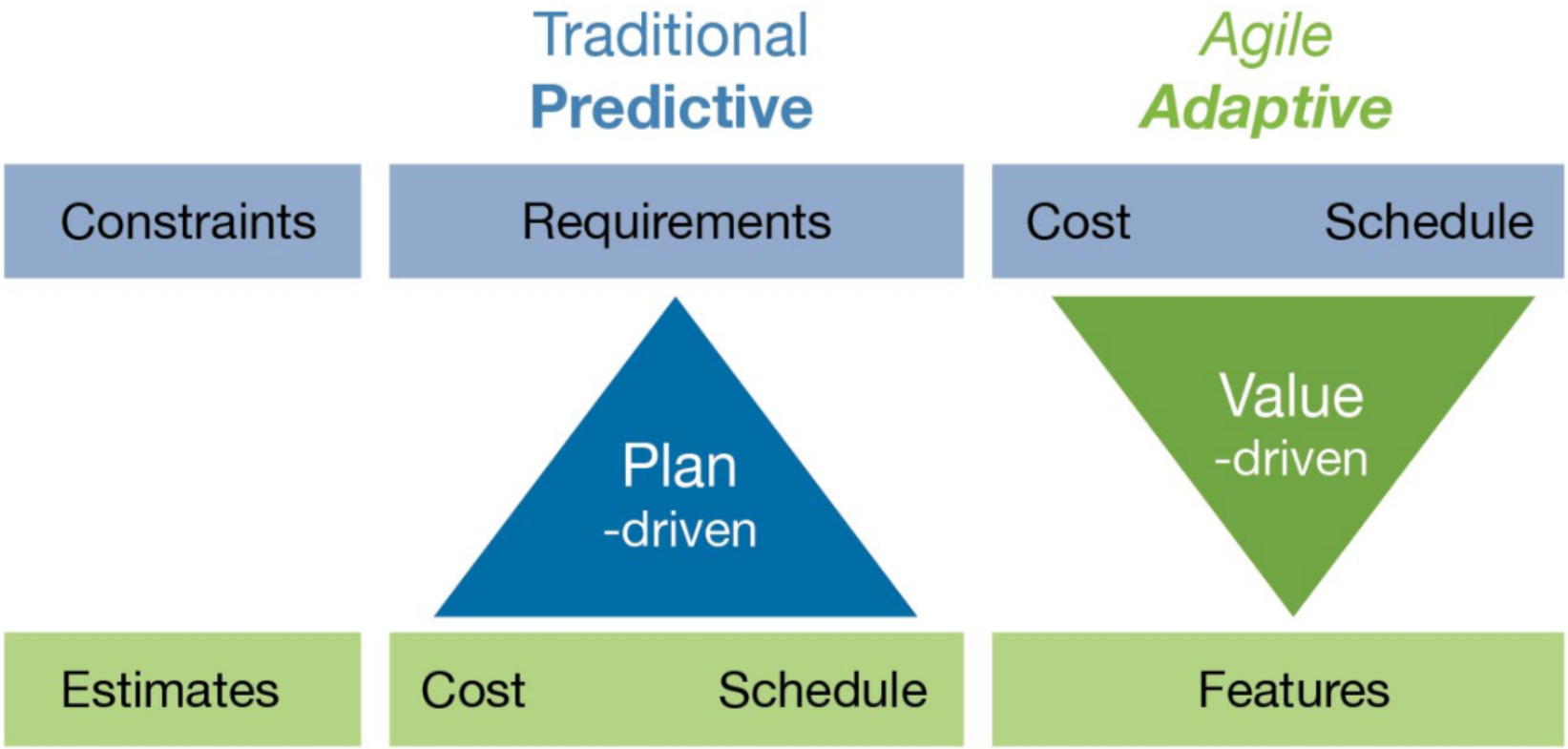
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

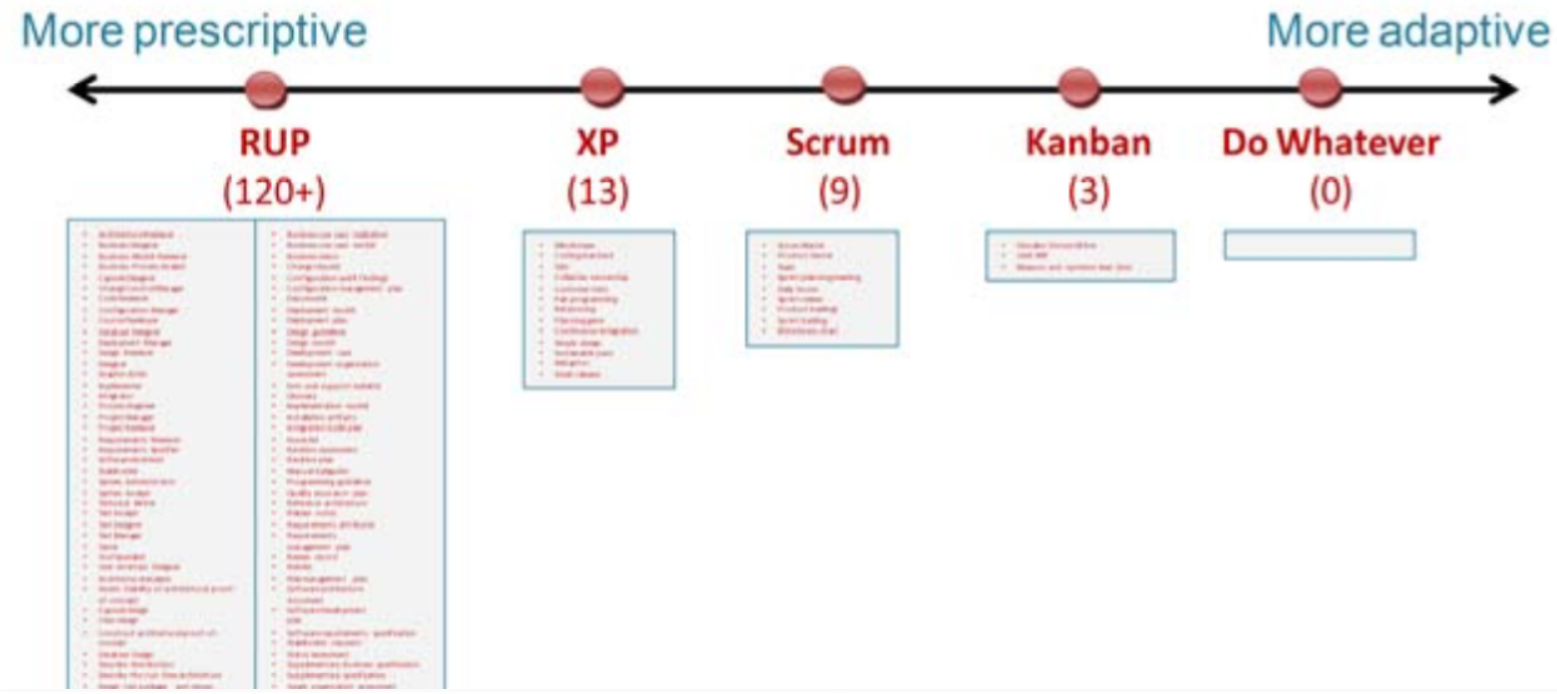
Simplicity--the art of maximizing the amount of work not done--is essential.

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.



In the traditional predictive model, the project manager attempts to constrain scope in order to provide reliable estimates for time and cost. In practice we experience a "triple constraint" or "iron triangle" situation and quality may become an undesired variable. In Agile we accept time and cost as real constraints. The product manager then works with the delivery team to maximize value by delivering iteratively and incrementally. The plan is regularly adapted to match reality. (How could we ever believe the other way around could work?)



RUP is pretty prescriptive – it has over 30 roles, over 20 activities, and over 70 artifacts; an overwhelming amount of stuff to learn. You aren't re

Scrum is less prescriptive than XP, since it doesn't prescribe any specific

Kanban leaves almost everything open. The only constraints are Visualize Your Workflow and Limit Your WIP. Just inches from Do Whatever, but still surprisingly powerful.

m

get too much, and you are supposed to remove the stuff you don't need. In Scrum you get too little, and you are supposed to add the stuff that is missing.

Don’t limit yourself to one tool!

Mix and match the tools as you need! I can hardly imagine a successful Scrum team that doesn’t include most elements of XP for example. Many Kanban teams use daily standup meetings (a Scrum practice). Some Scrum teams write some of their backlog items as Use Cases (a RUP practice) or limit their queue sizes (a Kanban practice). Whatever works for you.

Miyamoto Musashi a 17th century Samurai who was famous for his twin-sword fighting technique, said it nicely:

> Do not develop an attachment to any one weapon or any one school of fighting.
>
> - Miyamoto Musashi

http://www.ambysoft.com/essays/agileManifesto.html

1. **Individuals and interactions over processes and tools**. Teams of people build software systems, and to do that they need to work together effectively – including but not limited to programmers, testers, project managers, modelers, and your customers. Who do you think would develop a better system: five software developers and with their own tools working together in a single room or five low-skilled "hamburger flippers" with a well-defined process, the most sophisticated tools available, and the best offices money could buy? If the project was reasonably complex my money would be on the software developers, wouldn't yours? The point is that the most important factors that you need to consider are the people and how they work together because if you don't get that right the best tools and processes won't be of any use. Tools and processes are important, don't get me wrong, it's just that they're not as important as working together effectively. Remember the old adage, a fool with a tool is still a fool. As Fred Brooks points out in The Mythical Man Month, this can be difficult for management to accept because they often want to believe that people and time, or men and months, are interchangeable.

2. **Working software over comprehensive documentation**. When you ask a user whether they would want a fifty page document describing what you intend to build or the actual software itself, what do you think they'll pick? My guess is that 99 times out of 100 they'll choose working software. If that is the case, doesn't it make more sense to work in such a manner that you produce software quickly and often, giving your users what they prefer? Furthermore, I suspect that users will have a significantly easier time understanding any software that you produce than complex technical diagrams describing its internal workings or describing an abstraction of its usage, don't you? Documentation has its place, written properly it is a valuable guide for people's understanding of how and why a system is built and how to work with the system. However, never forget that the primary goal of software development is to create software, not documents – otherwise it would be called documentation development wouldn't it?

3. **Customer collaboration over contract negotiation**. Only your customer can tell you what they want. Yes, they likely do not have the skills to exactly specify the system. Yes, they likely won't get it right the first. Yes, they'll likely change their minds. Working together with your customers is hard, but that's the reality of the job. Having a contract with your customers is important, having an understanding of everyone's rights and responsibilities may form the foundation of that contract, but a contract isn't a substitute for communication. Successful developers work closely with their customers, they invest the effort to discover what their customers need, and they educate their customers along the way.

4. **Responding to change over following a plan**. People change their priorities for a variety of reasons. As work progresses on your system your project stakeholder's understanding of the problem domain and of what you are building changes. The business environment changes. Technology changes over time, although not always for the better. Change is a reality of software development, a reality that your software process must reflect. There is nothing wrong with having a project plan, in fact I would be worried about any project that didn't have one. However, a project plan must be malleable, there must be room to change it as your situation changes otherwise your plan quickly becomes irrelevant.

The Values of Extreme Programming

Extreme Programming (XP) is based on values. The rules we just examined are the natural extension and consequence of maximizing our values. XP isn't really a set of rules but rather a way to work in harmony with your personal and corporate values. Start with XP's values listed here then add your own by reflecting them in the changes you make to the rules.

Simplicity: We will do what is needed and asked for, but no more. This will maximize the value created for the investment made to date. We will take small simple steps to our goal and mitigate failures as they happen. We will create something we are proud of and maintain it long term for reasonable costs.

Communication: Everyone is part of the team and we communicate face to face daily. We will work together on everything from requirements to code. We will create the best solution to our problem that we can together.

Feedback: We will take every iteration commitment seriously by delivering working software. We demonstrate our software early and often then listen carefully and make any changes needed. We will talk about the project and adapt our process to it, not the other way around.

Respect: Everyone gives and feels the respect they deserve as a valued team member. Everyone contributes value even if it's simply enthusiasm. Developers respect the expertise of the customers and vice versa. Management respects our right to accept responsibility and receive authority over our own work.

Courage: We will tell the truth about progress and estimates. We don't document excuses for failure because we plan to succeed. We don't fear anything because no one ever works alone. We will adapt to changes when ever they happen.

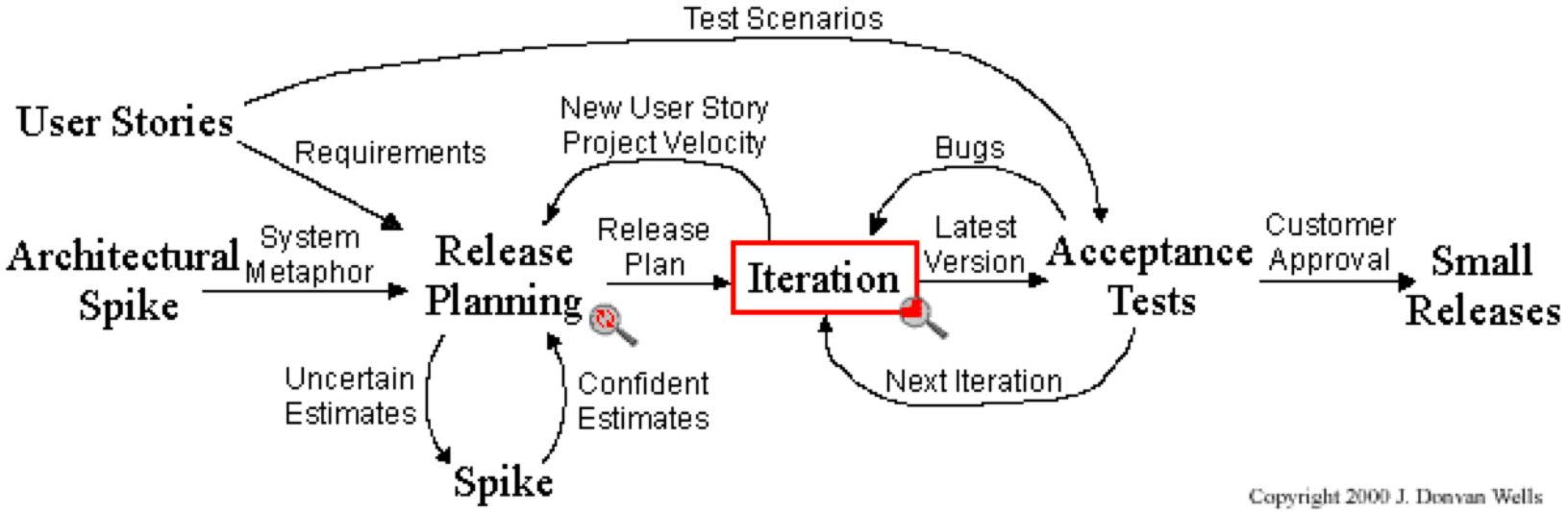
What lessons have we learned about implementing XP so far.

ExtremeProgramming.org home | XP Rules | XP Map | Lessons Learned | About the Author

http://www.extremeprogramming.org/rules.html

http://www.extremeprogramming.org/map/project.html

Extreme Programming Project



Designing

- Simplicity.
- Choose a system metaphor.
- Use CRC cards for design sessions.
- Create spike solutions to reduce risk.
- No functionality is added early.
- Refactor whenever and wherever possible.

Testing

- All code must have unit tests.
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance tests are run often and the score is published.

Let's review the values of Extreme Programming (XP) next.

ExtremeProgramming.org home | XP Map | XP Values | Test framework | About the Author

XP

When XP teams use **test-first programming**, they build unit tests first that express the behavior of the code that's about to be written, and then write code to make those tests pass.

Teams create a **10-minute build**, or an automated build that runs in under 10 minutes.

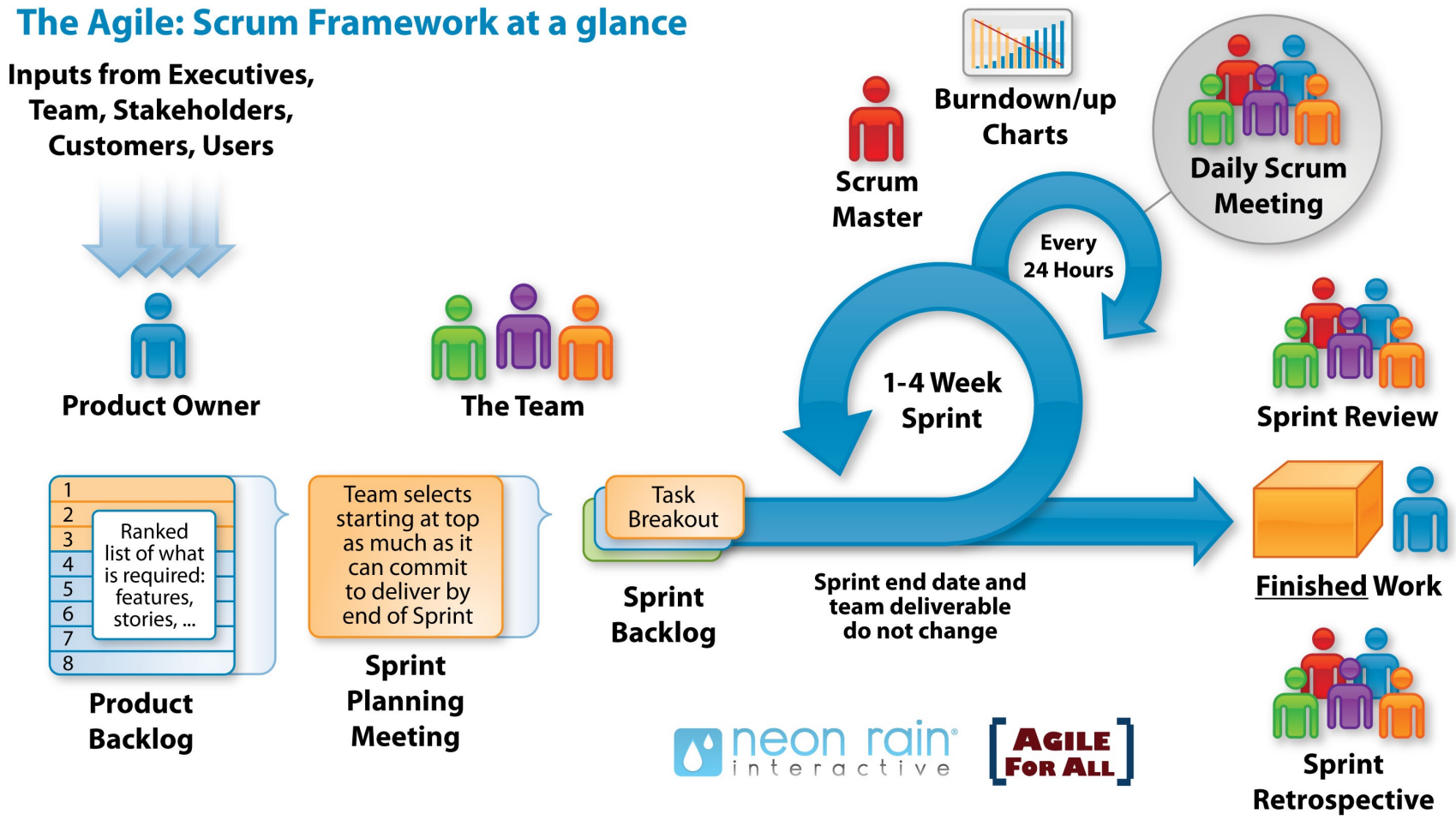
Individual developers use **continuous integration** to constantly integrate changes from their teammates so everyone's sandbox is up to date.

XP teams use **iteration** with their weekly cycle and quarterly cycle, and use stories in the same way that Scrum teams do.

When XP teams **sit together**, they periodically absorb project information through osmotic communication.

XP teams work in an **informative workspace** that using wall charts to automatically communicate information to people.

Scrum



https://www.scrum.org/

SCRUM

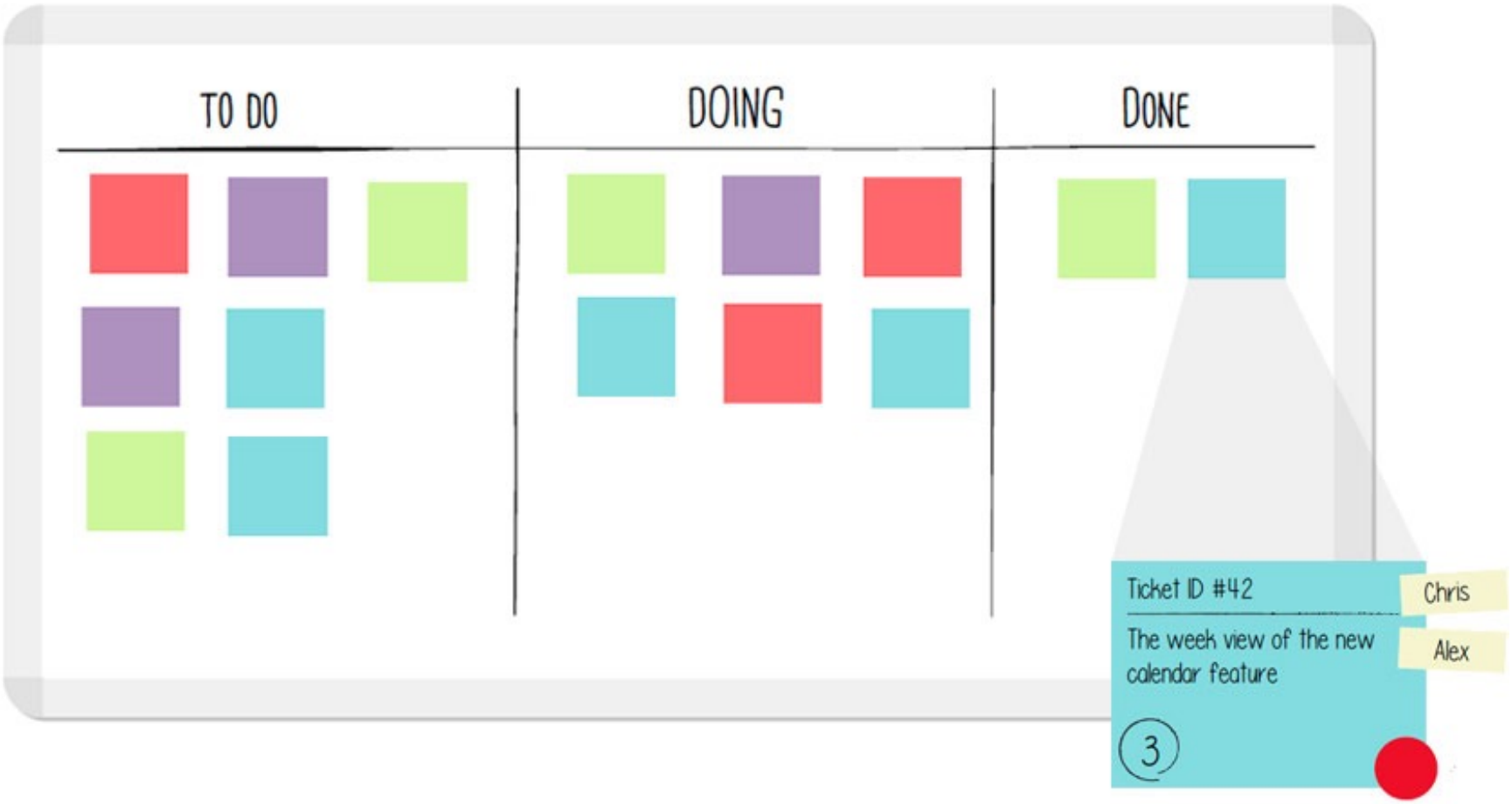
A **user story** helps the team understand their users by clearly explaining a specific need, and how the software will meet that need.

Each user story has **conditions of satisfaction** to help the team know when the story is done.

Teams use **story points and velocity** to estimate how many stories they can include in each sprint.

Posting a **burndown chart** clearly where everyone can see it can help everyone on the team understand what's completed, what's left to build, and whether or not they're on track.

Kanban



http://leankit.com/kanban/what-is-kanban/

LEAN

Lean is a mindset, not a methodology, which is why it **doesn't have practices** that help you run projects.

There are values shared between lean thinking and the larger world of agile: **decide as late as possible** (or making decisions at the last responsible moment) and **amplify learning** (using feedback and iterations).

The Lean value empower the team is very similar to the Scrum value of focus, and the XP value of **energized work**.

A **value stream** map is a visual tool to help Lean teams see the reality of how their project works by showing the entire lifespan of an Minimal Marketable Feature (**MMF)**, including the working and waiting time at every stage.

KANBAN

Kanban is a method for **process improvement**, or a way to help teams improve the way they build software and work together as a team, that's based on the **Lean mindset**.

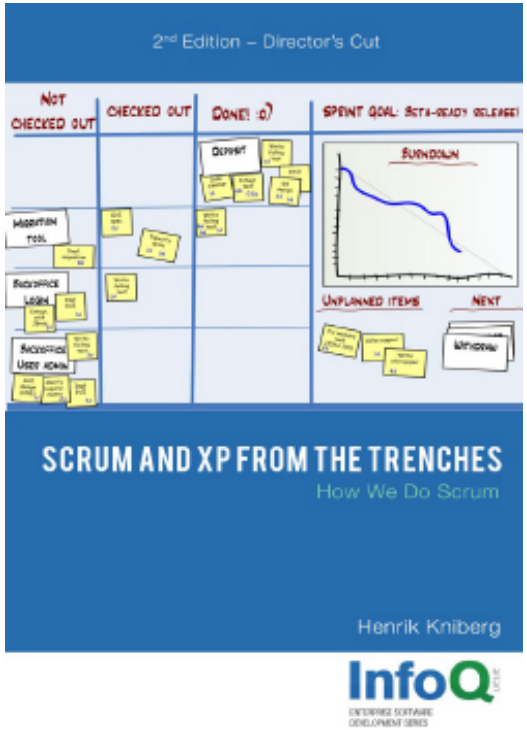
Kanban teams **start with what you do now** by seeing the whole as it exists today, and **pursue incremental, evolutionary change** to improve the system gradually over time.

The Kanban practice improve collaboratively, evolve experimentally means **taking measurements**, making **gradual improvements**, and then confirming that they worked using those measurements.

Kanban teams gradually improve the system by adding **WIP limits**, **managing flow**, and making **process policies explicit**, improved behavior often emerges in the rest of the company.

Recommended

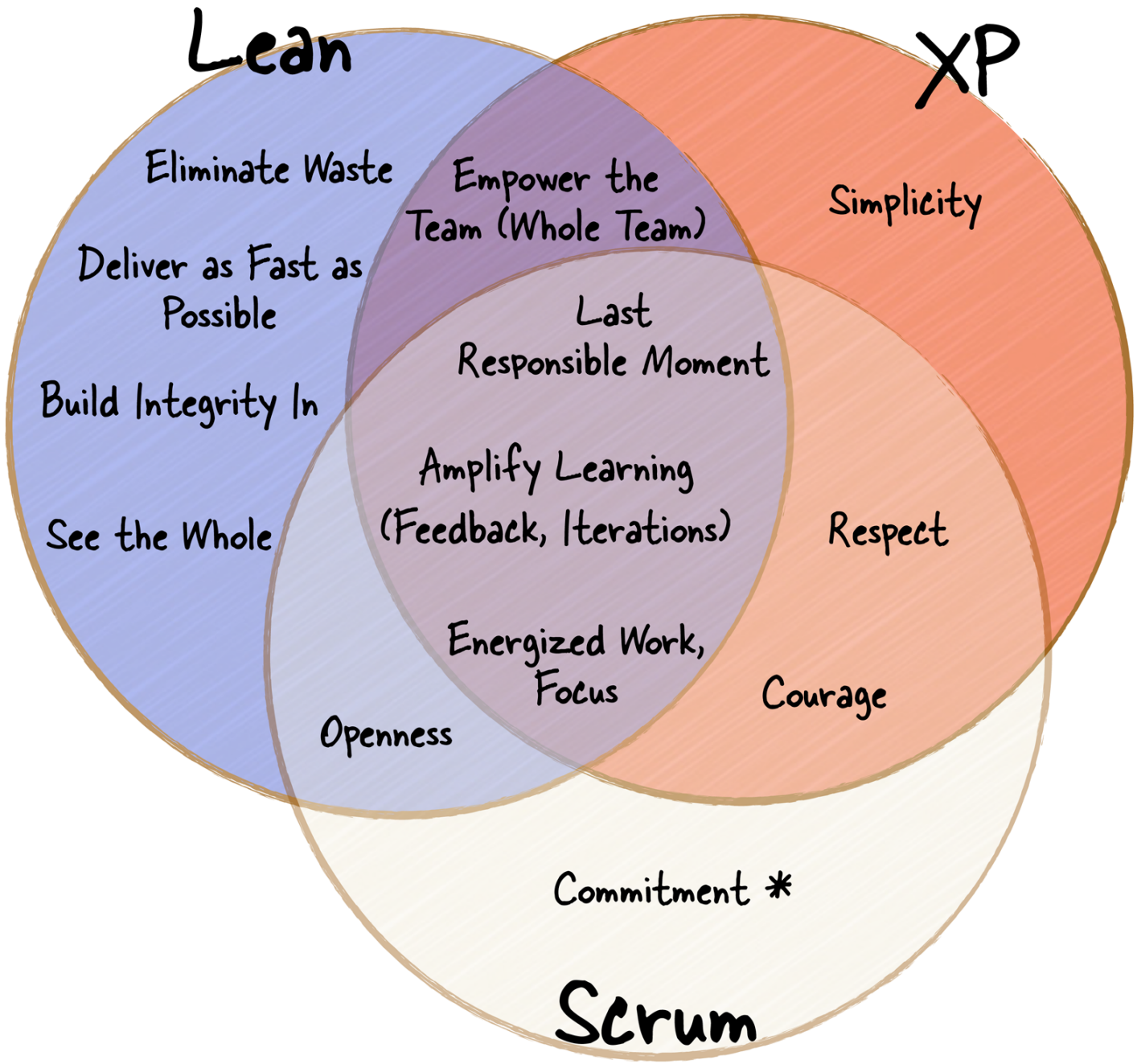
http://www.infoq.com/minibooks/scrum-xp-from-the-trenches-2



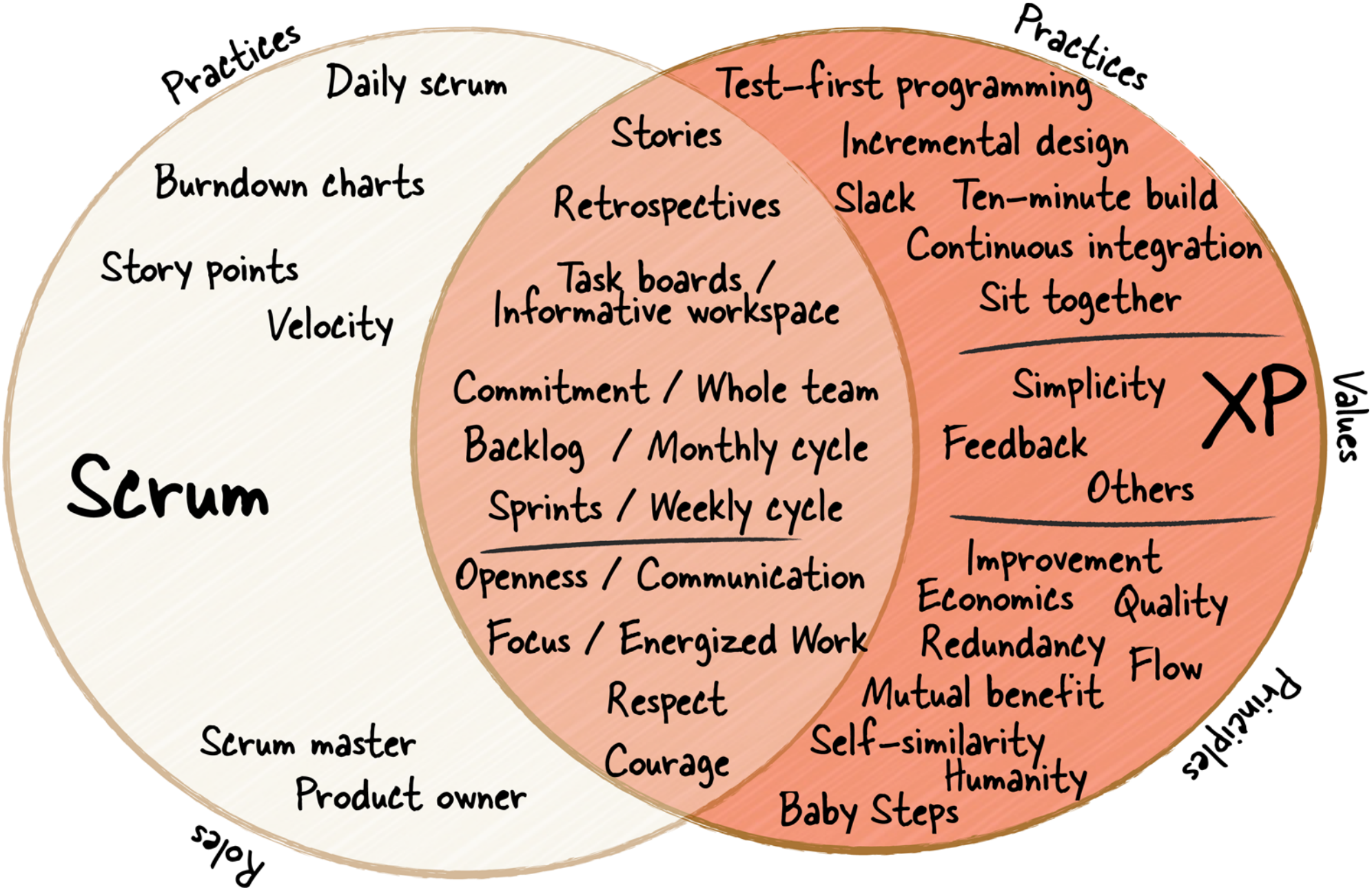
http://www.infoq.com/minibooks/kanban-scrum-minibook

Comparison

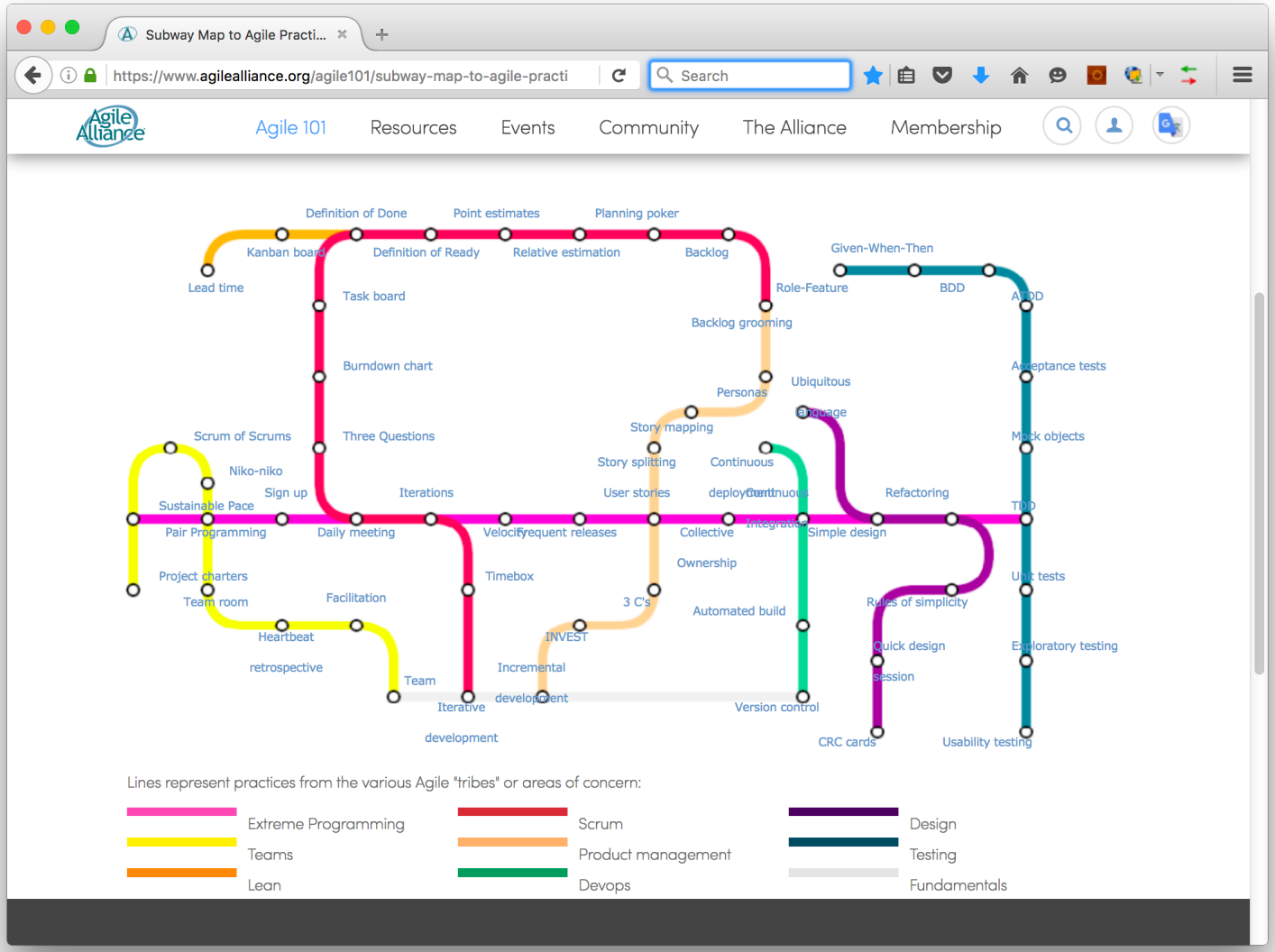
	Pre-Req	Strategy	Goal	Plan	Track	Key Metrics
Waterfall	Predictability Low Risk	Fixed Scope Fixed Time Variable Cost	Try to Stay on Budget	Planned Hours Milestones Schedule	Actual Hours Blockers/Risks	% Complete Gantt Charts
Scrum	Dedicated Team Independent Stories	Fixed Time Fixed Cost Variable Scope	Try to Predict Delivery Date & Team Capacity	Sprint Scope Define “Done”	Remaining Hours	Team Velocity Burn Down Chart
Kanban	N/A	Limit Work in Progress	Find Process Bottleneck	N/A	Status Queue	Avg. Cycle Time Cumulative Flow



* Commitment is an important part of XP and Lean, but Scrum explicitly calls it out as a value.

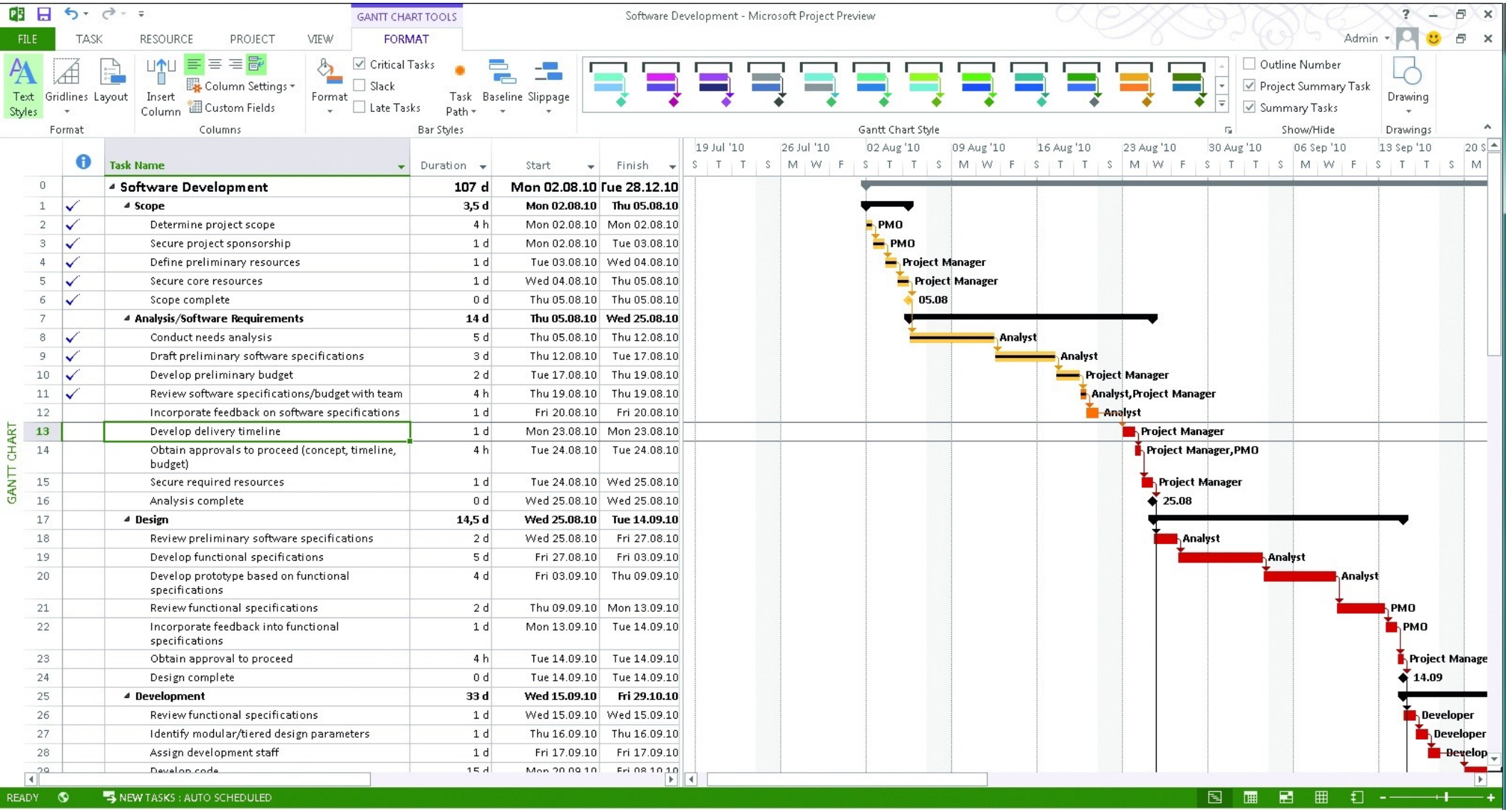

Practices
Daily scrum
Burndown charts
Story points
Velocity
Scrum
Scrum master
Product owner
Roles
Stories
Retrospectives
Task boards / Informative workspace
Commitment / Whole team
Backlog / Monthly cycle
Sprints / Weekly cycle
Openness / Communication
Focus / Energized Work
Respect
Courage
Practices
Test-first programming
Incremental design
Slack
Ten-minute build
Continuous integration
Sit together
Simplicity
XP
Feedback
Others
Improvement
Economics
Quality
Redundancy
Flow
Mutual benefit
Self-similarity
Humanity
Baby Steps
Principles
Values

Agile Practices



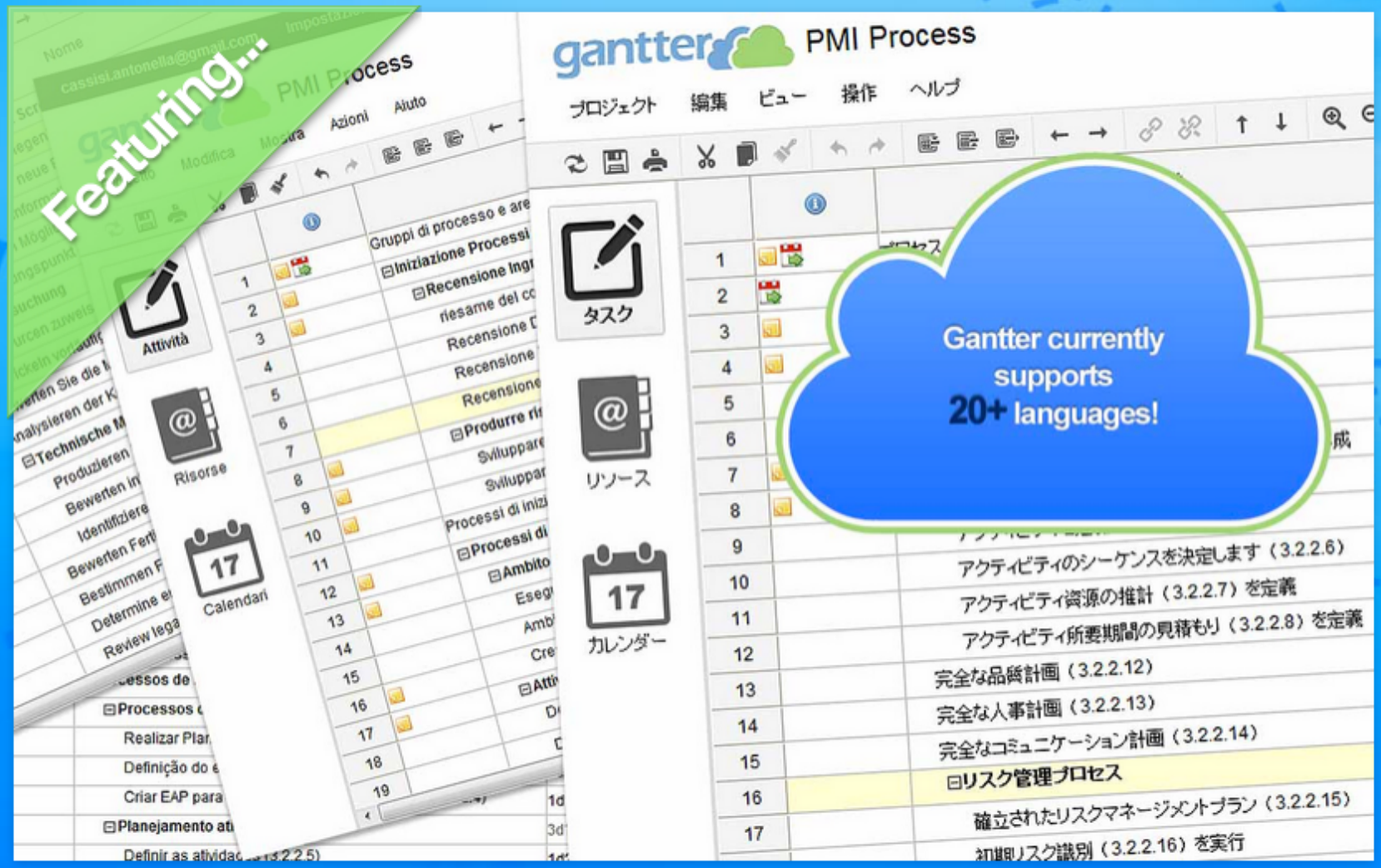
https://www.agilealliance.org/agile101/subway-map-to-agile-practices/

Tools


GANTT CHART TOOLS
Software Development - Microsoft Project Preview
FILE
TASK
RESOURCE
PROJECT
VIEW
FORMAT
Admin
Text Styles
Gridlines
Layout
Format
Insert Column
Column Settings
Custom Fields
Columns
Format
Critical Tasks
Slack
Late Tasks
Task Path
Baseline
Slippage
Bar Styles
Gantt Chart Style
Outline Number
Project Summary Task
Summary Tasks
Show/Hide
Drawing
Drawings
Task Name
Duration
Start
Finish
19 Jul '10
26 Jul '10
02 Aug '10
09 Aug '10
16 Aug '10
23 Aug '10
30 Aug '10
06 Sep '10
13 Sep '10
0 Software Development 107 d Mon 02.08.10 Tue 28.12.10
1 Scope 3,5 d Mon 02.08.10 Thu 05.08.10
2 Determine project scope 4 h Mon 02.08.10 Mon 02.08.10
3 Secure project sponsorship 1 d Mon 02.08.10 Tue 03.08.10
4 Define preliminary resources 1 d Tue 03.08.10 Wed 04.08.10
5 Secure core resources 1 d Wed 04.08.10 Thu 05.08.10
6 Scope complete 0 d Thu 05.08.10 Thu 05.08.10
7 Analysis/Software Requirements 14 d Thu 05.08.10 Wed 25.08.10
8 Conduct needs analysis 5 d Thu 05.08.10 Thu 12.08.10
9 Draft preliminary software specifications 3 d Thu 12.08.10 Tue 17.08.10
10 Develop preliminary budget 2 d Tue 17.08.10 Thu 19.08.10
11 Review software specifications/budget with team 4 h Thu 19.08.10 Thu 19.08.10
12 Incorporate feedback on software specifications 1 d Fri 20.08.10 Fri 20.08.10
13 Develop delivery timeline 1 d Mon 23.08.10 Mon 23.08.10
14 Obtain approvals to proceed (concept, timeline, budget) 4 h Tue 24.08.10 Tue 24.08.10
15 Secure required resources 1 d Tue 24.08.10 Wed 25.08.10
16 Analysis complete 0 d Wed 25.08.10 Wed 25.08.10
17 Design 14,5 d Wed 25.08.10 Tue 14.09.10
18 Review preliminary software specifications 2 d Wed 25.08.10 Fri 27.08.10
19 Develop functional specifications 5 d Fri 27.08.10 Fri 03.09.10
20 Develop prototype based on functional specifications 4 d Fri 03.09.10 Thu 09.09.10
21 Review functional specifications 2 d Thu 09.09.10 Mon 13.09.10
22 Incorporate feedback into functional specifications 1 d Mon 13.09.10 Tue 14.09.10
23 Obtain approval to proceed 4 h Tue 14.09.10 Tue 14.09.10
24 Design complete 0 d Tue 14.09.10 Tue 14.09.10
25 Development 33 d Wed 15.09.10 Fri 29.10.10
26 Review functional specifications 1 d Wed 15.09.10 Wed 15.09.10
27 Identify modular/tiered design parameters 1 d Thu 16.09.10 Thu 16.09.10
28 Assign development staff 1 d Fri 17.09.10 Fri 17.09.10
PMO
PMO
Project Manager
Project Manager
05.08
Analyst
Analyst
Project Manager
Analyst,Project Manager
Analyst
Project Manager
Project Manager,PMO
Project Manager
25.08
Analyst
Analyst
Analyst
PMO
PMO
Project Manage
14.09
Developer
Developer
Develop
GANTT CHART
READY
NEW TASKS : AUTO SCHEDULED

Collaborative cloud scheduling made easy

Start Now



Gantter is a FREE web-based project management tool. You can think of it as a web-based Microsoft Project.

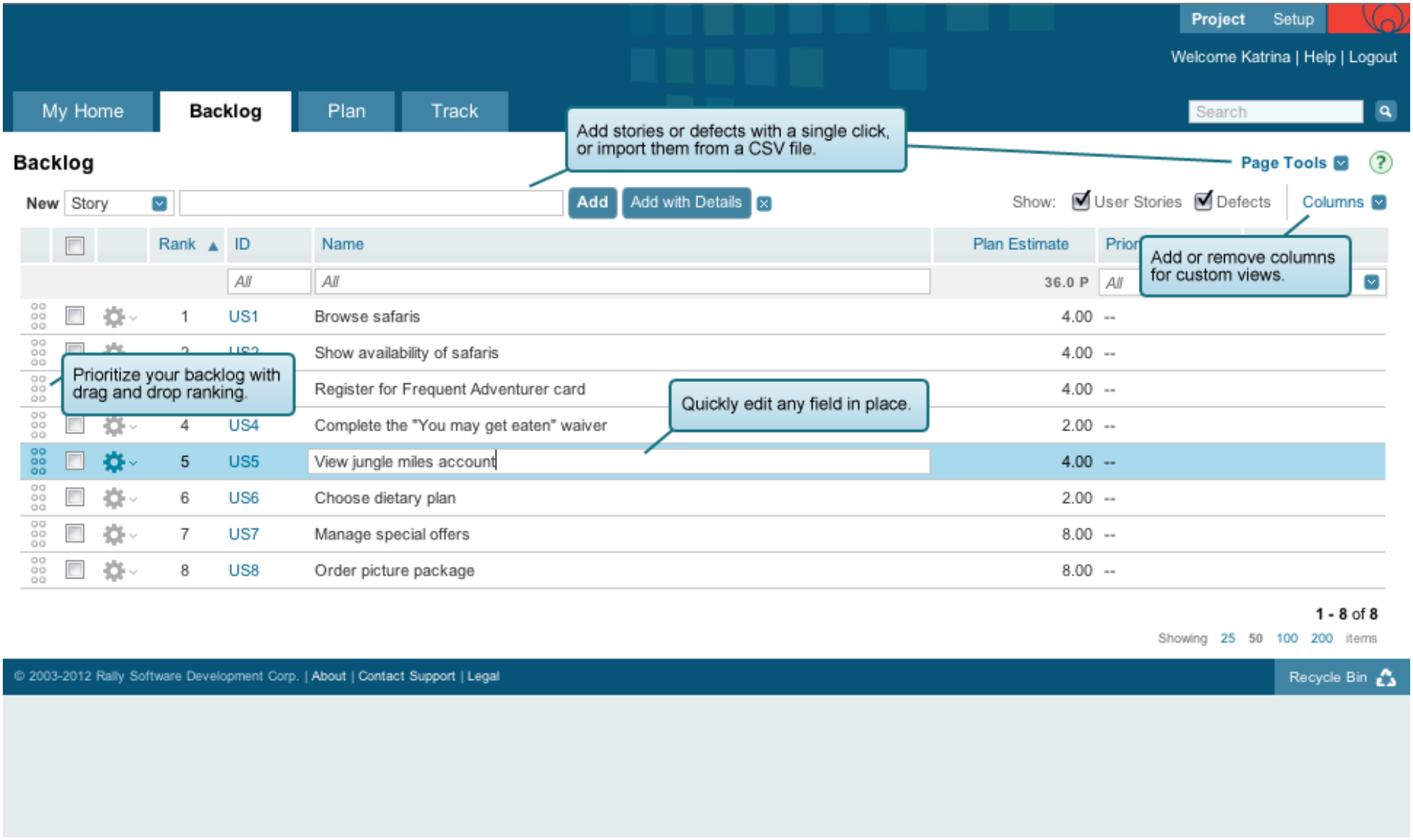
https://www.rallydev.com/platform-products/rally-editions

Team Collaboration | Agile Project Management | Agile at Scale | Product & Portfolio Management | Analytics & Metrics | Platform & Integrations

Team Collaboration

Get your teams in the flow.

	Community	Enterprise	Unlimited
Group chat: Discuss, notify, and share in realtime	✓	✓	✓
Team inbox: Email, feeds, and tool notifications in one place	✓	✓	✓
Chat threading: See your discussions organized by context	✓	✓	✓
ChatOps: Manage and update DevOps activities from your chat window	✓	✓	✓
Enterprise scale: Secure, single sign-on with service level agreements			✓

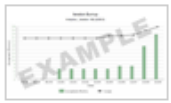

Project
Setup
Welcome Katrina | Help | Logout
My Home
Backlog
Plan
Track
Search
Backlog
Add stories or defects with a single click, or import them from a CSV file.
Page Tools
New
Story
Add
Add with Details
Show:
User Stories
Defects
Columns
Rank
ID
Name
Plan Estimate
Add or remove columns for custom views.
All
All
36.0 P
1 US1 Browse safaris 4.00 --
Show availability of safaris 4.00 --
Register for Frequent Adventurer card 4.00 --
4 US4 Complete the "You may get eaten" waiver 2.00 --
5 US5 View jungle miles account 4.00 --
6 US6 Choose dietary plan 2.00 --
7 US7 Manage special offers 8.00 --
8 US8 Order picture package 8.00 --
Prioritize your backlog with drag and drop ranking.
Quickly edit any field in place.
1 - 8 of 8
Showing 25 50 100 200 items
© 2003-2012 Rally Software Development Corp. | About | Contact Support | Legal
Recycle Bin

Sample Project | PLAN | TRACK

Reports

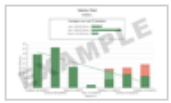
Iteration Burndown

Work remaining in the iteration to proactively anticipate whether the iteration



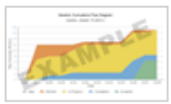
Iteration Burnup

Work delivered so far in the iteration to proactively anticipate wh



Velocity Chart

Trend in value delivery, including work accepted after the iteratio average velocity



Iteration Cumulative Flow Diagram

Work-in-progress status to visually analyze the trend in lead time

Iteration Defects by Priority

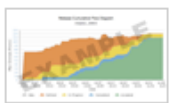
Non-closed defects in the iteration, categorized by priority to ass

Iteration Defects by State

All defects in the iteration, categorized by state to assess end of

Release Burnup

Work delivered so far in the release to proactively anticipate whe



Release Cumulative Flow Diagram

Work-in-progress status to visually analyze the trend in lead time

Story Burnup

High-level progress view for a feature/product/initiative (represer across multiple releases

Story Burndown

Work remaining to deliver a story, or a feature (epic story) suppo

Story Cumulative Flow Diagram

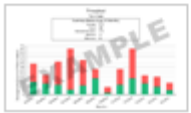
Work-in-progress status for a story, or a feature (epic story) supp

Tagged Story Burndown

Work remaining to deliver a set of stories grouped by a common tag

Cycle/Lead Time

The average number of days it takes work to flow through your process



Throughput

The count of work items accepted in a given interval

Release Defect Trend

Cumulative defects opened versus closed in a release

Defects by Priority

Defects categorized by priority over time

Defects by State

Defects categorized by state over time

Defect Trend

Cumulative defects opened versus closed over time, showing whether the trend is increasing or decreasing

Build Health

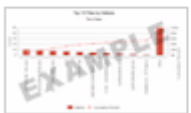
Proportions of build success to failure along with build duration for the past 30 days

Build Health by Iteration

Proportions of build success to failure along with build duration for the past 10 iterations

Top Files by Changes

Files that changed the most often during the past 30 days

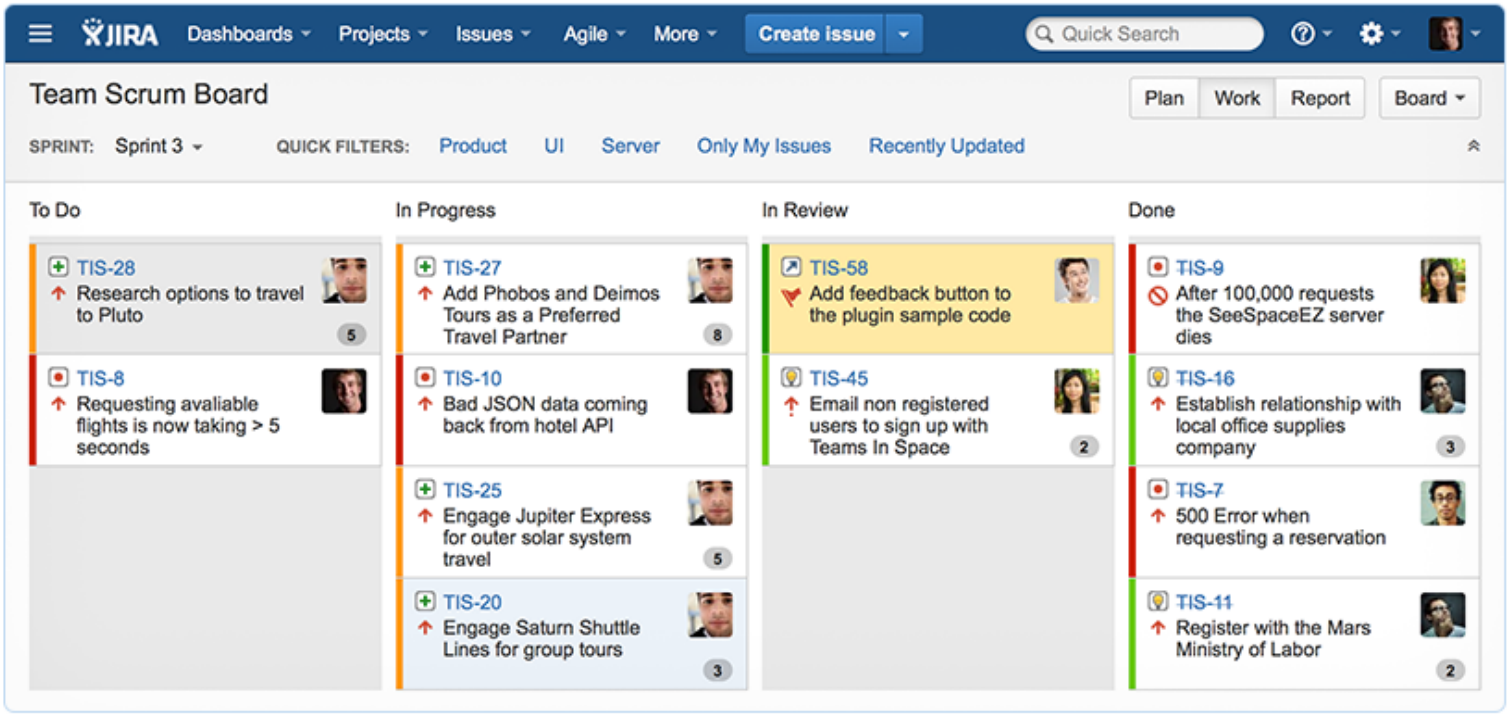


Top Files by Defects

Files that are related to the most defects during the past 30 days

Why JIRA Agile?

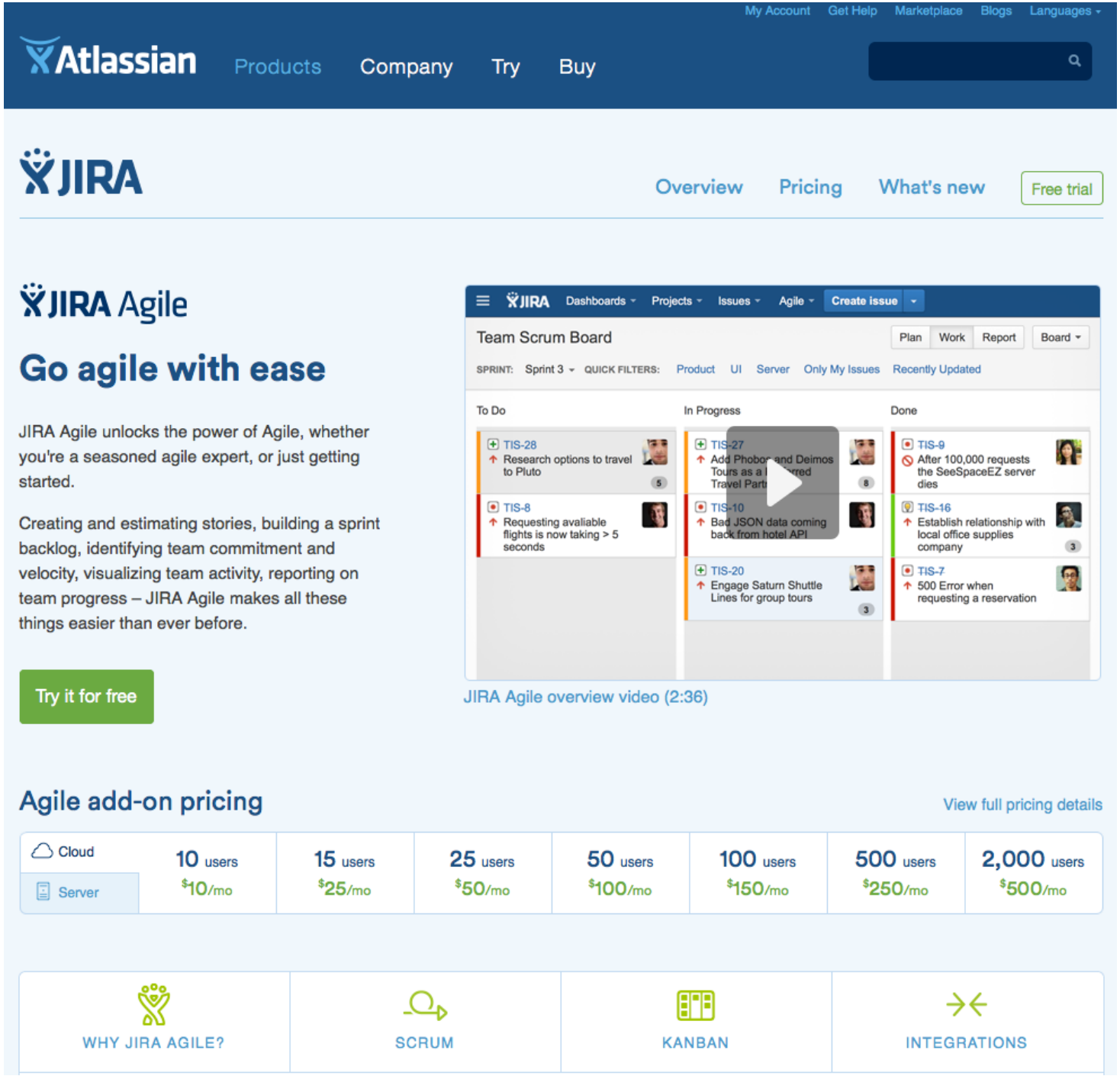
PLAN ESTIMATE PRIORITIZE EXECUTE EVOLVE



Flexible planning

Scrum? Check. Kanban? Check. JIRA Agile brings all the power of JIRA to agile. JIRA Agile's rich feature set enables you to flexibly plan and adopt the best agile practices for your organization.

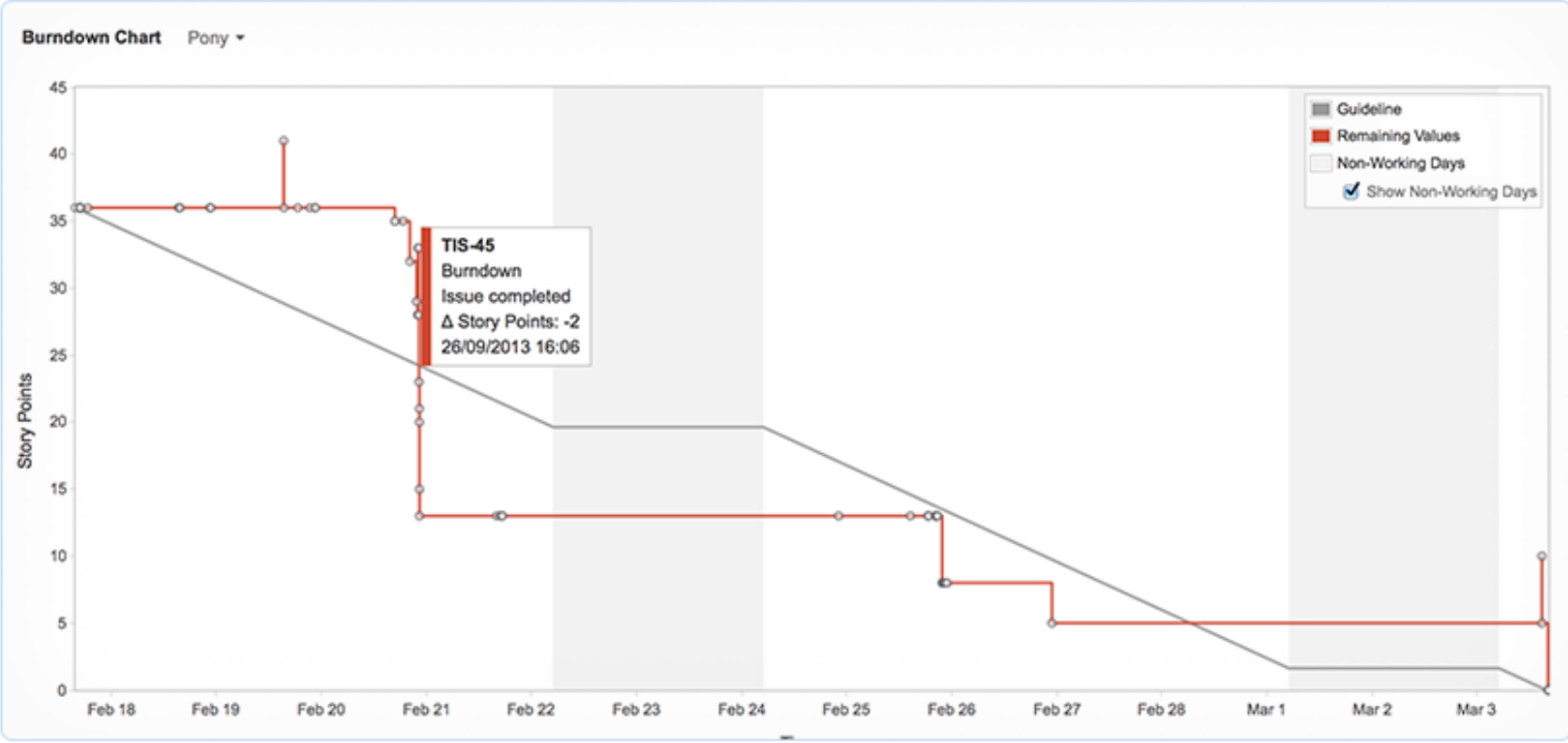
JIRA Agile integrates with JIRA, Confluence, and our Dev Tools like Stash for a seamless developer experience.



https://www.atlassian.com/software/jira/agile

Scrum

ADOPT SCHEDULE LAUNCH DELIVER GROW



Actionable results

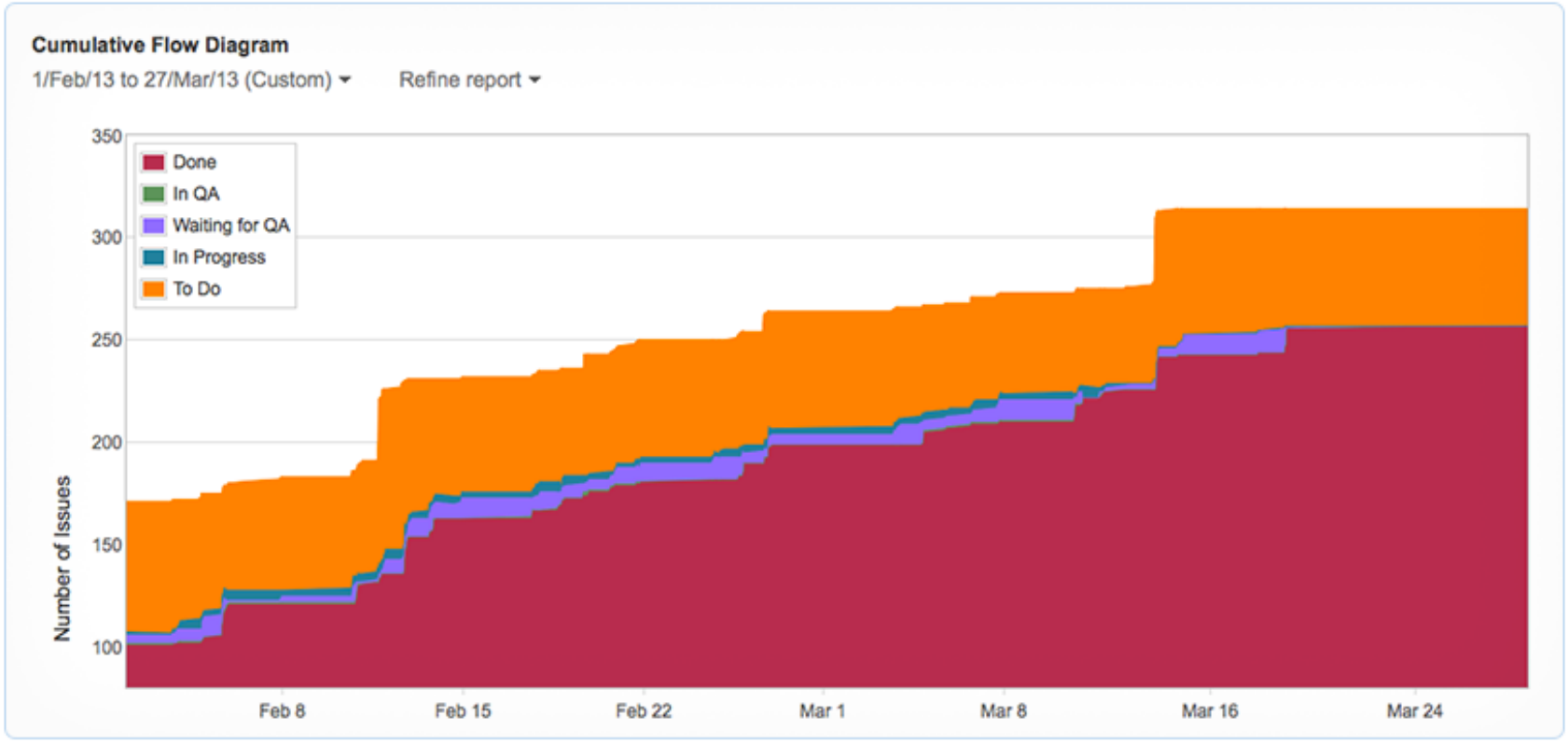
Retrospectives are sprint reviews which help the agile team evolve. JIRA Agile's extensive reports give the team critical insight into their agile process and results.

The team can follow their progress towards the sprint commitment using the burndown chart; If the scope of the sprint has changed, JIRA Agile highlights the additional work.

Want to track larger items? The epic and version reports go across sprints to show delivery of feature level work.

Kanban

CHAMPION VISUALIZE ORGANIZE OPTIMIZE ANALYZE



Constructive analysis

Monitor current trends and analyze past progress with the cumulative flow diagram. See where work is building up and bottlenecks occur.

Backlog

Control Chart

Switch report

Board

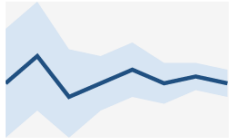
How to read this chart

Shows the cycle time for your product, version or sprint. This helps you identify whether data from the current process can be used to determine future performance.

Learn more | Hide this information

Visibility
See outliers and investigate their cause to reduce them in the future.

Efficiency
Decreasing rolling average indicates process improvements and increased throughput.

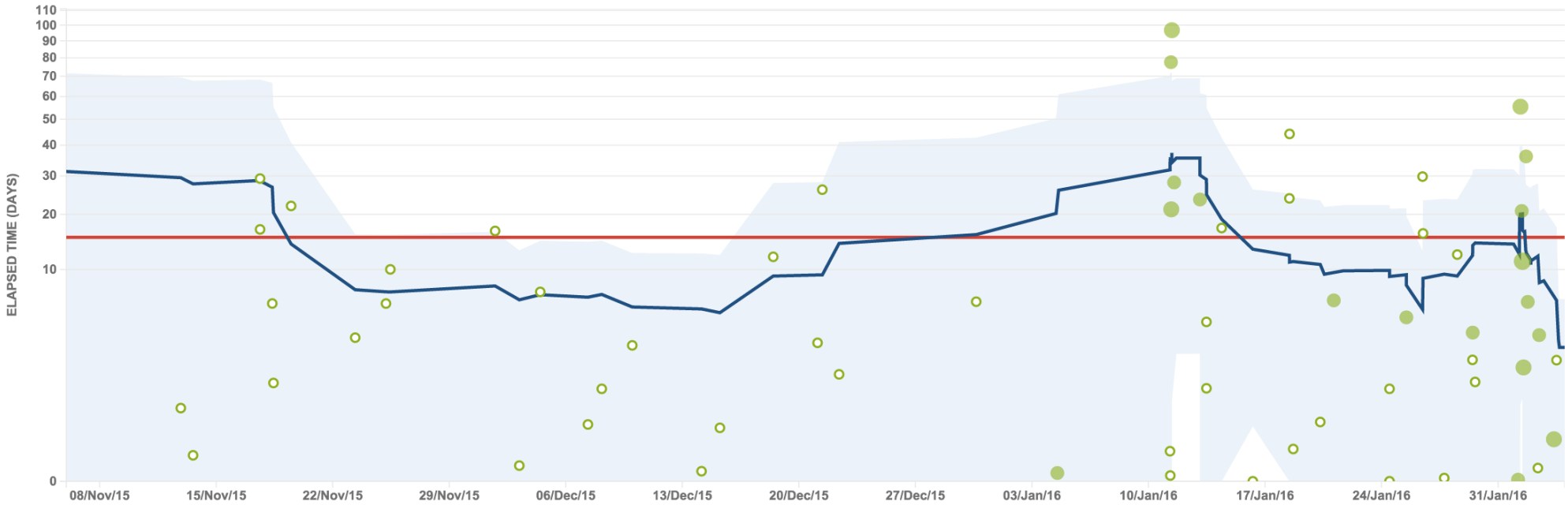


Predictability
Narrow standard deviation through process improvements to improve predictability of cycle time.

06/Nov/15 to 03/Feb/16 (Past 3 Months)

2w 1d 6h average **5d 16h** median **0** min time **14w 2d 1h** max time **86** issues

Issue
Cluster of issues
Average
Rolling average
17 issue window
Standard deviation



easyBacklog

Blog FAQ Support Sign up Log in

Turns the marathon into a sprint

Join now for free

easyBacklog

Makes Agencies Agile.

click to flip

- Manage backlog and sprints
- Faster than Excel
- Revision control
- Print and export
- Rapid quotations
- Scalable and secure

front

As a:

scrum master

I want to:

manage backlogs and sprints

So I can:

implement Agile effectively

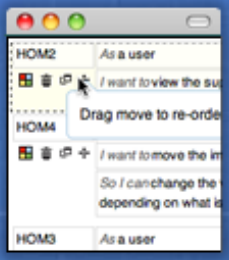
back

Acceptance criteria:

- Stories are grouped into themes
- Drag and drop prioritisation
- Automatic theme and backlog totals
- Burn up and burn down charts
- Sensible defaults and constraints
- Color coding categorisation

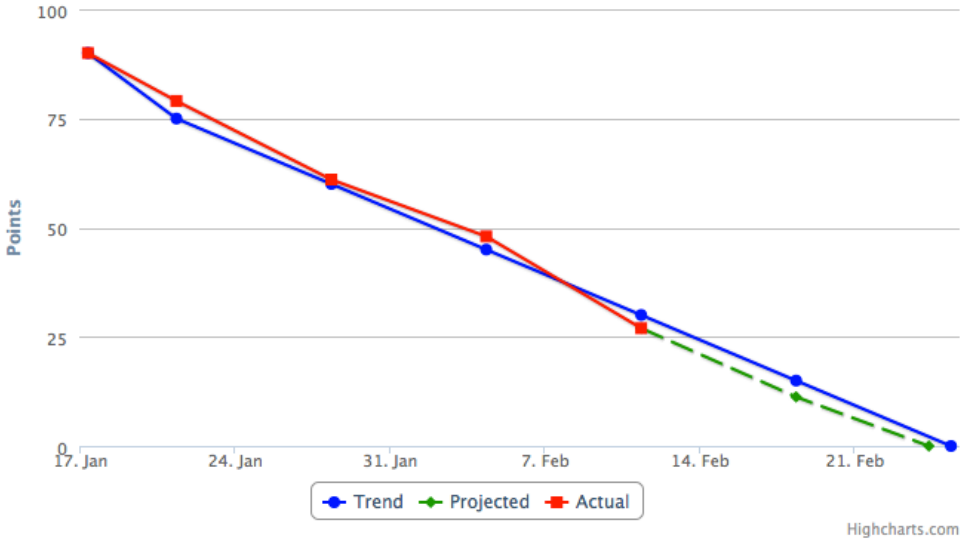
intro video

screenshots



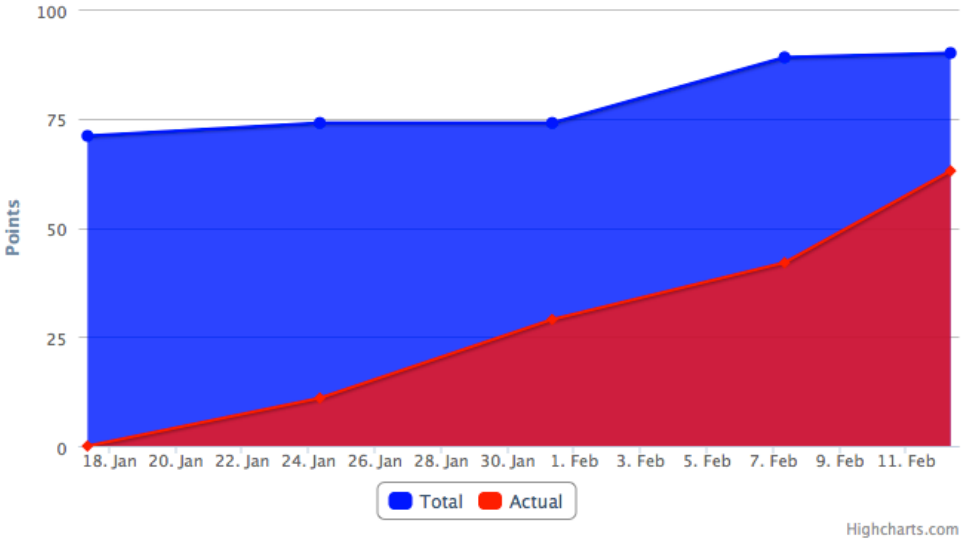
Statistics

Burn down

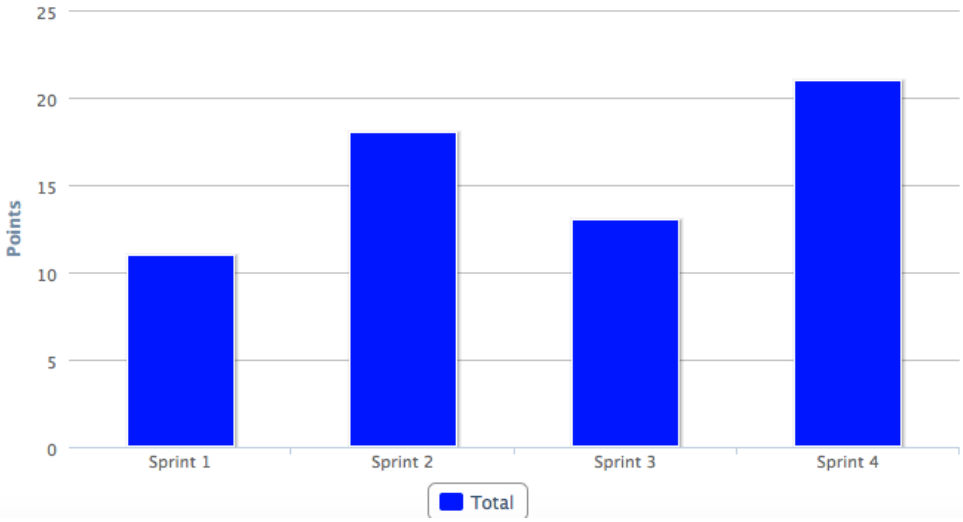


Show projected based on average velocity

Burn up



Velocity of completed sprints



Velocity

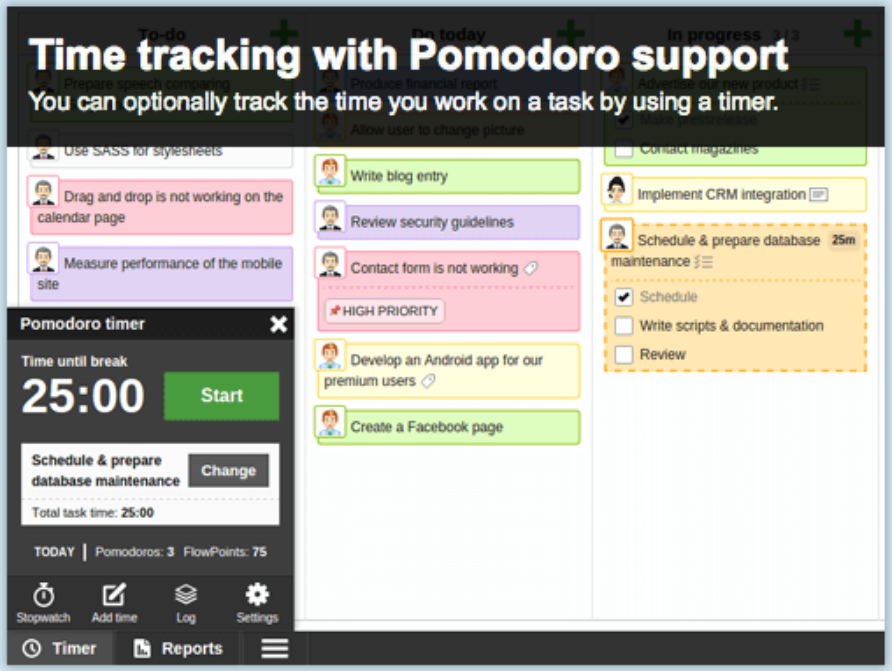
	Average	Expected
Velocity per day	3.2	3
Velocity per sprint	15.8	15

The expected backlog daily velocity is configured as 3 points per day per team member.
Change backlog settings »

KanbanFlow
LEAN PROJECT MANAGEMENT, SIMPLIFIED

HOME FEATURES PREMIUM PRESS SIGN UP LOGIN

Boost your personal or team productivity



Sign up for a free account

Full name

Email

Password

Sign up

Visualize your work

The Kanban board gives you an excellent overview of your current work situation. Visualizing work in a team environment simplifies communication and lead to improved effectiveness.

Limit your work in progress

By focusing your efforts on working on fewer items at the same time, you will get more done. Lots more actually. And you will feel less stressed.

Collaborate in real-time

Any changes you make on the board are immediately available on all the team members' displays. You will always know what your team is working on at any moment.

Time tracking with Pomodoro

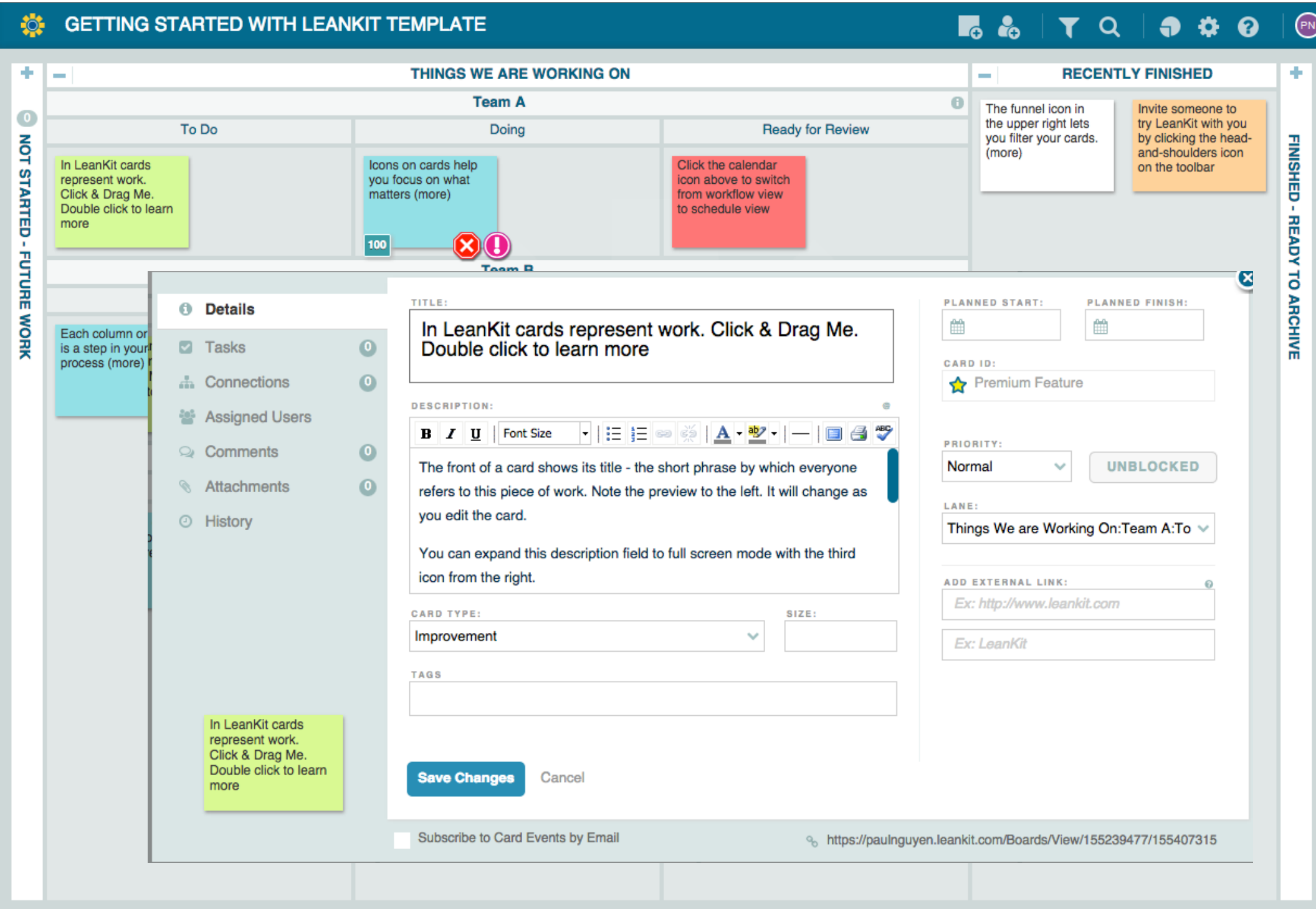
Optionally improve your focus even more by using the popular **Pomodoro technique** for time tracking of your tasks. Track and measure any interruptions to your focus.

Fast & intuitive UI

Get things done, quickly. It takes less than 2 minutes to learn how to use KanbanFlow. The user interface is built for speed, without frustrating loading times.

Mobile support

When you are away from your computer you can use our mobile site. It works well on **Android**, **iPhone**, **Windows Phone** and most other smartphones.


GETTING STARTED WITH LEANKIT TEMPLATE
NOT STARTED - FUTURE WORK
THINGS WE ARE WORKING ON
Team A
To Do
Doing
Ready for Review
In LeanKit cards represent work. Click & Drag Me. Double click to learn more
Icons on cards help you focus on what matters (more)
100
Click the calendar icon above to switch from workflow view to schedule view
Team B
RECENTLY FINISHED
The funnel icon in the upper right lets you filter your cards. (more)
Invite someone to try LeanKit with you by clicking the head-and-shoulders icon on the toolbar
FINISHED - READY TO ARCHIVE
Each column or is a step in your process (more)
Details
Tasks
Connections
Assigned Users
Comments
Attachments
History
TITLE:
In LeanKit cards represent work. Click & Drag Me. Double click to learn more
DESCRIPTION:
Font Size
The front of a card shows its title - the short phrase by which everyone refers to this piece of work. Note the preview to the left. It will change as you edit the card.
You can expand this description field to full screen mode with the third icon from the right.
CARD TYPE:
Improvement
SIZE:
TAGS
PLANNED START:
PLANNED FINISH:
CARD ID:
Premium Feature
PRIORITY:
Normal
UNBLOCKED
LANE:
Things We are Working On:Team A:To
ADD EXTERNAL LINK:
Ex: http://www.leankit.com
Ex: LeanKit
In LeanKit cards represent work. Click & Drag Me. Double click to learn more
Save Changes
Cancel
Subscribe to Card Events by Email
https://paulnguyen.leankit.com/Boards/View/155239477/155407315