Introduction to Agile

Why Agile?

Today it seems like a deprecated question.

Applied correctly, in software development Agile nowadays seems to have no alternative. In part, because it simply gets the job done - by **building the right product**.

Through the iterative and incremental way of work, it allows to:

- assess the direction of a project throughout the development lifecycle and making necessary changes,
- continue gathering and refining requirements while building the product,
- greatly reduce both development costs and time to market.

In waterfall model, development teams only have *one chance* to get each aspect of a project right.

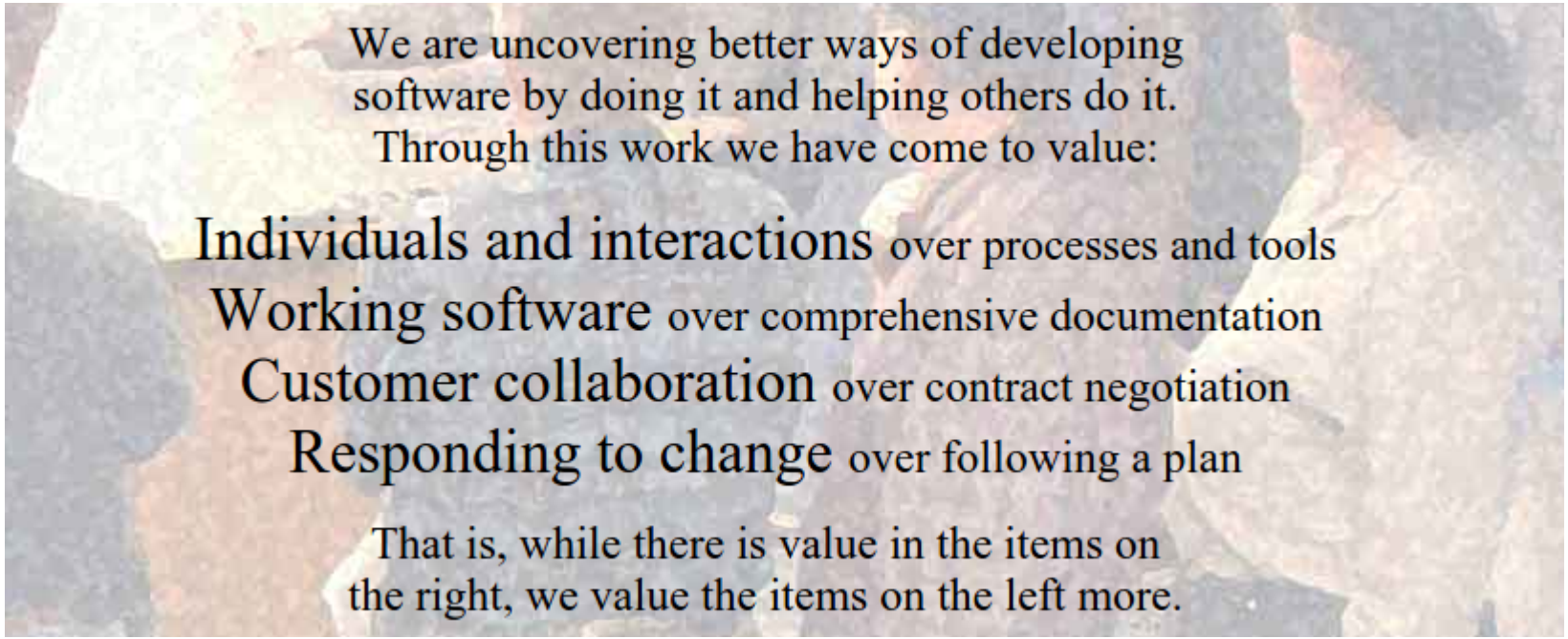
While in Agile every aspect of development — requirements, design, etc. — is c*ontinually revisited* throughout the lifecycle. When a team stops and re-evaluates the direction of a project every two weeks, there's always time to steer it in better direction.

Agile story in brief

- Ideas about iterative and incremental software development (as opposed to “waterfall” models) can be traced back to the **1950th**.
- Big boost in **1990th** with appearance of (what would later become) SCRUM and XP as well as their less popular counterparts like Rational Unified Process (RUP), Feature-driven development (FDD), Dynamic systems development method (DSDM) or Crystal Clear.
- Agile Manifesto in **Feb, 2001**.
- Also in **2001**, Ken Schwaber’s and Mike Beedle's book "Agile Software Development with Scrum".
- Project Management “Declaration of Independance” in **2005**.
- Application of Kanban in software development in **mid-2000th**.
- In **2011** the Agile Alliance created the Guide to Agile Practices.
- The "The Lean Startup" book by Eric Ries, **2011**.

Agile Manifesto

Created in 2001. Can always be read at http://agilemanifesto.org/.



Agile Principles

The Manifesto was based on **12 Agile principles**, which are available at http://agilemanifesto.org/principles.html

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

PM Declaration of Interdependence

Created in **2005** with intention **to help Project Managers to be most effective in Agile context**:

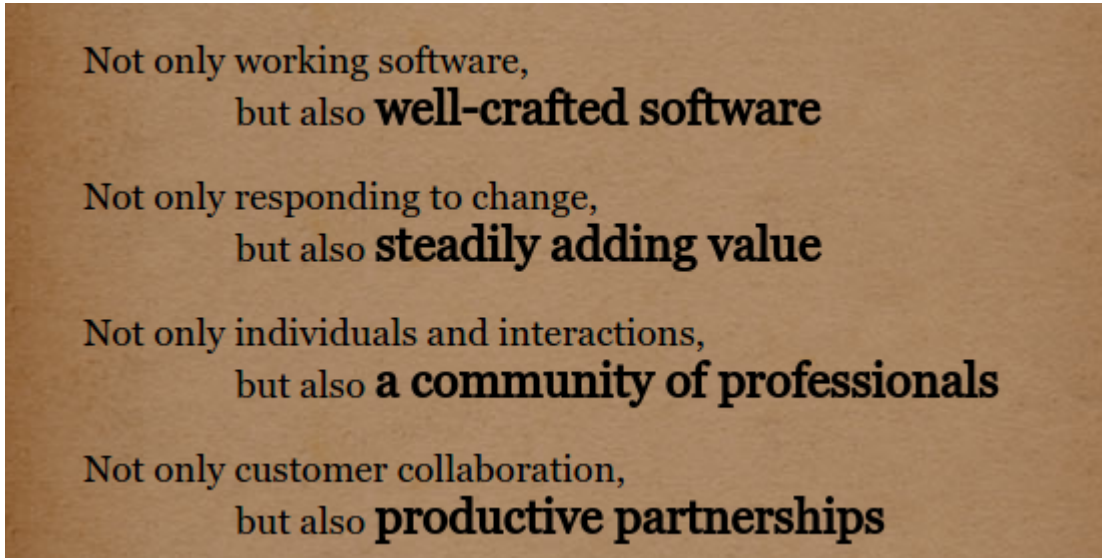
"We ...

- **increase return on investment** by *making continuous flow of value* our focus.
- **deliver reliable results** by *engaging customers* in frequent interactions and shared ownership.
- **expect uncertainty** and manage for it through *iterations, anticipation and adaptation*.
- **unleash creativity and innovation** by recognizing that *individuals are the ultimate source of value* and creating an environment where they can make a difference.
- **boost performance** through *group accountability* for results and *shared responsibility for team effectiveness.*
- **improve effectiveness and reliability** through *situationally specific strategies*, processes and practices."

See more: http://pmdoi.org/

Software Craftsmanship Manifesto

Published in 2009, available on http://manifesto.softwarecraftsmanship.org/:



Agile Requirements

- **Software requirements in Agile change their form and the "philosophy".** As opposed to comprehensive specification documents used in heavy weight "waterfall" methodology, Agile requirements are light and organized as "small portions".

- The most popular form of Agile requirements is User Story (US): **short, simple feature description** which define specific value that said feature brings to specific stakeholders.

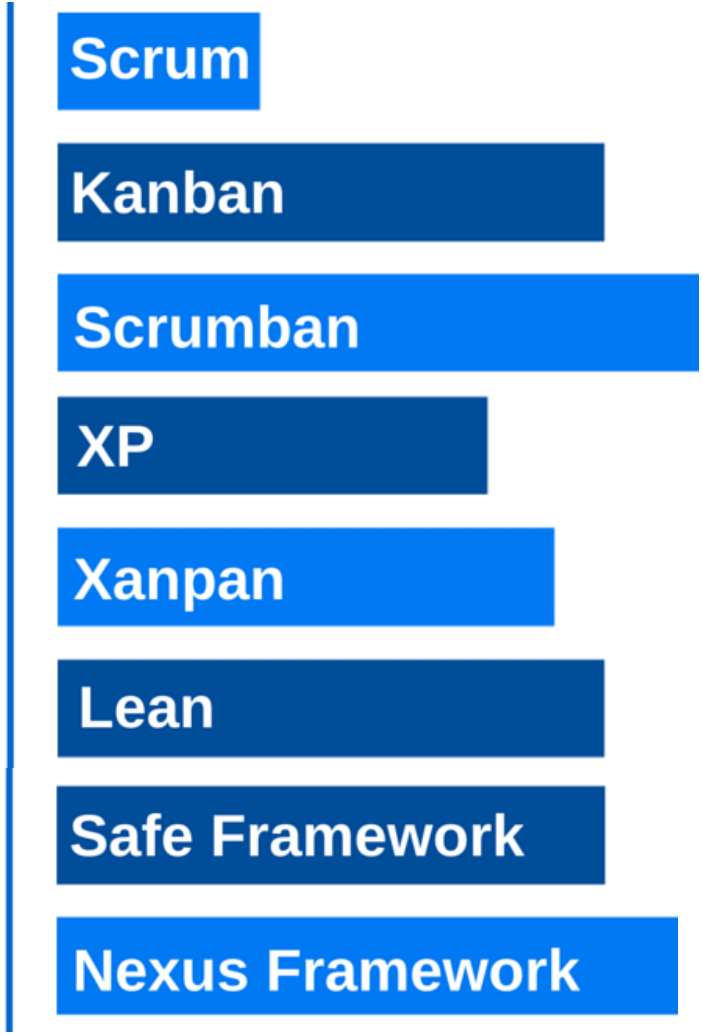
- One of the most popular formats of US is the following (taken from the book "User Stories Applied" by Mike Cohn):

  "As a <role>, I want <goal/desire> so that <benefit>" (however, other formats exist and are used in successful projects)

- User Stories are collected together into the Agile artifact usually called Product Backlog.

Read more about it: in Mike Cohn's "User Stories Applied"

Agile methods



SCRUM

SCRUM

SCRUM is one of the most popular Agile frameworks, especially for new software product development (as opposed to maintenance and support).

SCRUM was created by **Jeff Sutherland** and **Ken Schwaber** who presented a paper describing it at OOPSLA ’95 *(methodology which later was reformulated as SCRUM)* in Austin, Texas.

Jeff Sutherland is the CEO at Scrum, Inc. Ken Schwaber is a founder of Scrum.org.

In 2001 Ken Schwaber and Mike Beedle wrote a book **"Agile Software Development with Scrum"**.

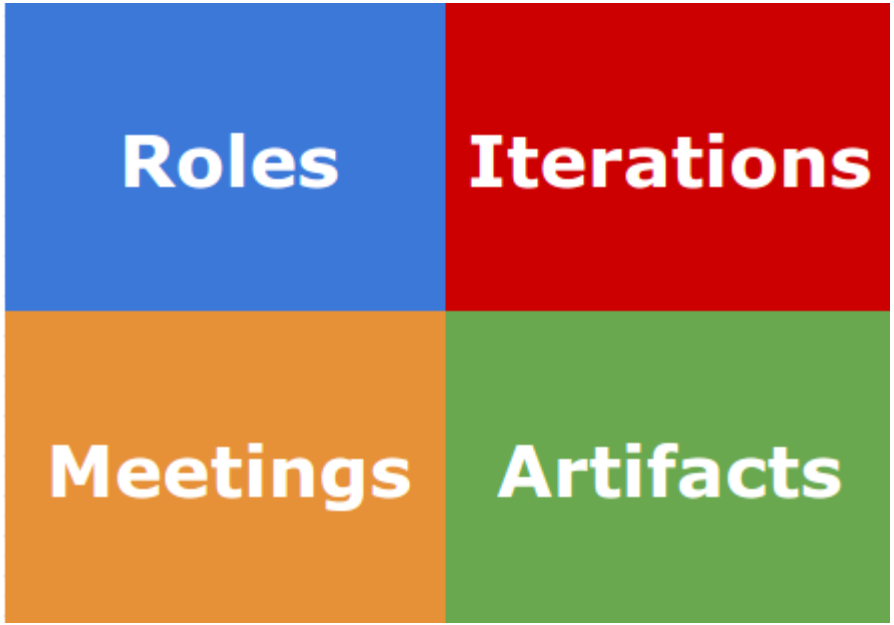
About SCRUM online:

- https://www.scrumalliance.org/
- https://www.scrum.org/

SCRUM Process

SCRUM process consists of **four major categories**:

- Roles
- Iterations
- Meetings
- Artifacts



SCRUM Roles

- **Product Owner (PO)**

  For Development Team PO represents all project's Stakeholders and is responsible for the Product Requirements (aka. Product Backlog).

- **Scrum Master (SM)**

  SM is a "Scrum leader" in the Team: he/she makes sure that accepted Scrum process is fully followed and inside the Iteration protects the Team from external disturbances/interruptions.

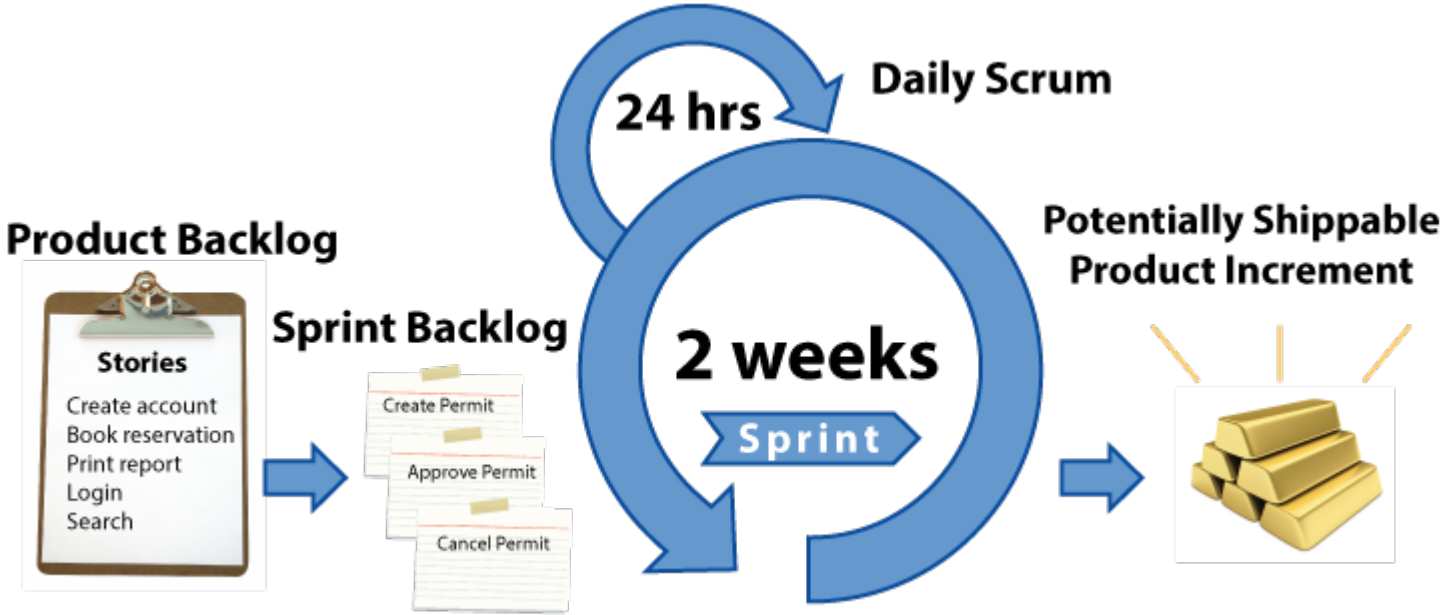
- **Development Team**

  Group of software development professionals responsible for delivering the Product. Scrum does not distinguish between professional specialization of the Team members, so depending on the Product Team might consist of developers, testers, analysts, technical writers and so on.

SCRUM Iterations

- Iterations in Scrum are called **Sprints.**
- The duration of each Sprint is **restricted**, it is usually between 1 week to couple of month, 1-3 weeks being the most popular time-box.
- During each Sprint Team works on selected number of User Stories and **commits to delivering them in the end of the Sprint**:
    - Sprint starts with **Planning meeting** where Team commits to specific number of User Stories to be completed;
    - Every day Team meets for **short Stand-up meetings** to track progress and voice existing impediments;
    - In the end of Sprint during **Demo** all completed User Stories are showed to Product Owner and are accepted or rejected;
    - After Demo, on **Retrospective meeting** Team reflects on good and bad things which happened during Sprint, and chooses the ways which will help to fix the most harmful among bad ones.

SCRUM process



SCRUM Meetings

Planning

One of the longer meetings which occur at the beginning of each Sprint and consists of two parts. During the first part Team discusses with PO the highest priority User Stories (US) from Product Backlog and commit to deliver several of them in the end of the Sprint. Second part mainly consists of breaking down these stories to the specific tasks level, writing them down, estimating and putting them on Scrum Board.

Daily Stand-up

Time-boxed (usually 15 mins) meeting which occurs daily at the same time in the same place (usually - in front of Scrum Board). Everyone on the Team should be present, prepared and be standing up (not sitting). Every Team member has to basically answer 3 standard questions: *What I was doing since previous Stand-up? What I plan to do until the next Stand-up? Are there any impediments to my work?* (If any significant impediments are voiced - SM helps deal with them after the meeting).

Demo

In the end of Sprint Team shows completed User Stories to PO and gets PO’s approval or rejection on every one of them.

Retrospective

Internal Teams meeting dedicated to discussing problems in the process and ways to fix them. If used correctly, this meeting is a powerful tool of making Scrum really work for a Team and bring results.

Backlog Grooming

Optional for some Teams, this meeting gives the Team additional time and PO’s undivided attention to discuss Product Backlog items.

SCRUM Artifacts

Product Backlog

Product Backlog is a prioritized list of desired features (requirements) which is usually maintained by Product Owner.

Spring Backlog

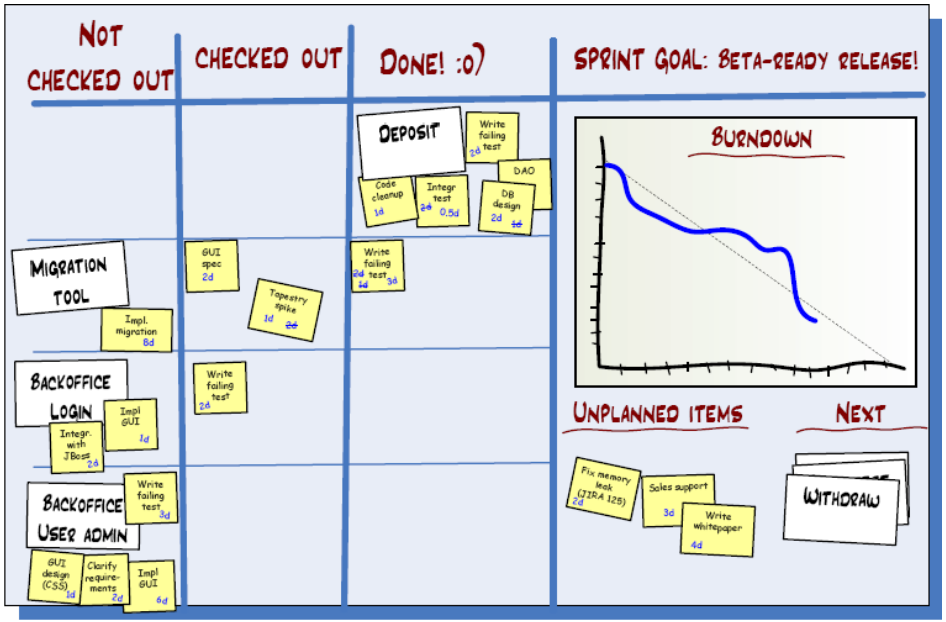
Spring Backlog is a list of features from Product Backlog which are taken into development during specific Scrum Iteration.

Scrum Board

Visible representation of Sprint on the whiteboard owned by the Team.

Sprint Burndown chart

The Sprint Burndown chart is a publicly displayed chart showing showing the evolution of remaining effort against time.



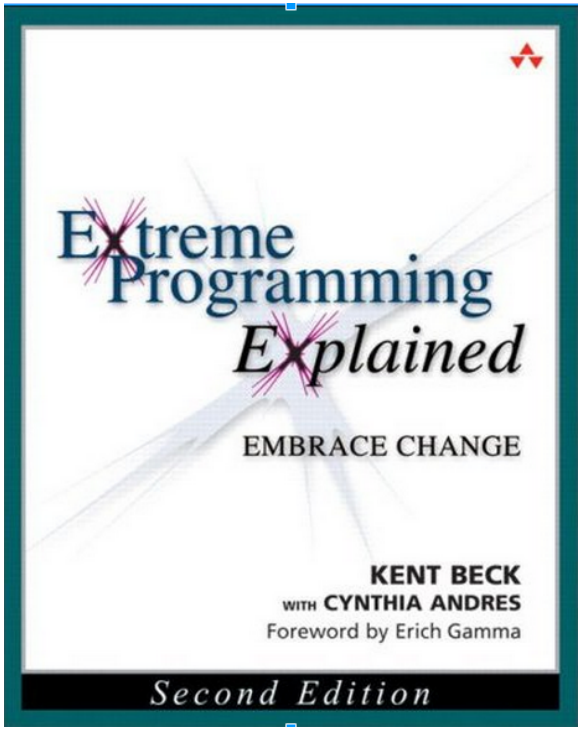
eXtreme Programming (XP)

XP

- Extreme Programming was created by **Kent Beck** and first described in his book **"Extreme Programming Explained" (1999)**. Overall, XP is **complete software development methodology**, quite similar to Scrum and **centered around programmers**.
- However, as a methodology it is used much less often than its **12 software development practices** are used on their own. These practices are almost at a level of industry standard and are accepted by all Agile methodologies as “optional, but extremely useful”.
- XP is one of those methodologies that doesn’t see analysts or architects as legitimate roles in development team. Also the role of tester is very much reduced to sort of “part-time” and may be assigned to one of the programmers.

12 software development practices of XP:

1. Pair programming
2. Planning game
3. Test-driven development
4. Whole team
5. Continuous integration
6. Refactoring or design improvement
7. Small releases
8. Coding standards
9. Collective code ownership
10. Simple design
11. System metaphor
12. Sustainable pace



XP vs. Scrum

- From “methodology” point of view SCRUM and XP are very similar. Many differences are mainly semantics (SCRUM’s product owner is close to XP’s customers; SCRUM’s sprints are XP’s iterations; and so on).
- However, from more strict point of view, XP is a **software development methodology** that encompasses the entire lifecycle of a project. SCRUM is a **project management methodology** that explicitly says nothing about how a software project gets done.
- Scrum does not allow changes to the scope inside of the Spirit while XP is more flexible about it.
- Scrum allows testers, analysts and other IT specialists (who are not programmers per ce) to be part of a Team if it makes sense for the project.
- Perhaps the biggest difference, though, is the very thing that makes these two frameworks so compatible. **XP mandates a set of engineering practices**; Scrum does not. At the same time, **SCRUM mandates planning ceremonies and artifacts**, where XP does not.

Kanban

Kanban

Kanban is a technique for managing a software development process in an efficient way inspired by the **Toyota Production System** and by **Lean manufacturing**.

The application of Kanban is much broader than just in software development and it is used in many other business areas.

Application of Kanban in software development gained first popularity in mid-2000th.

More about Kanban:

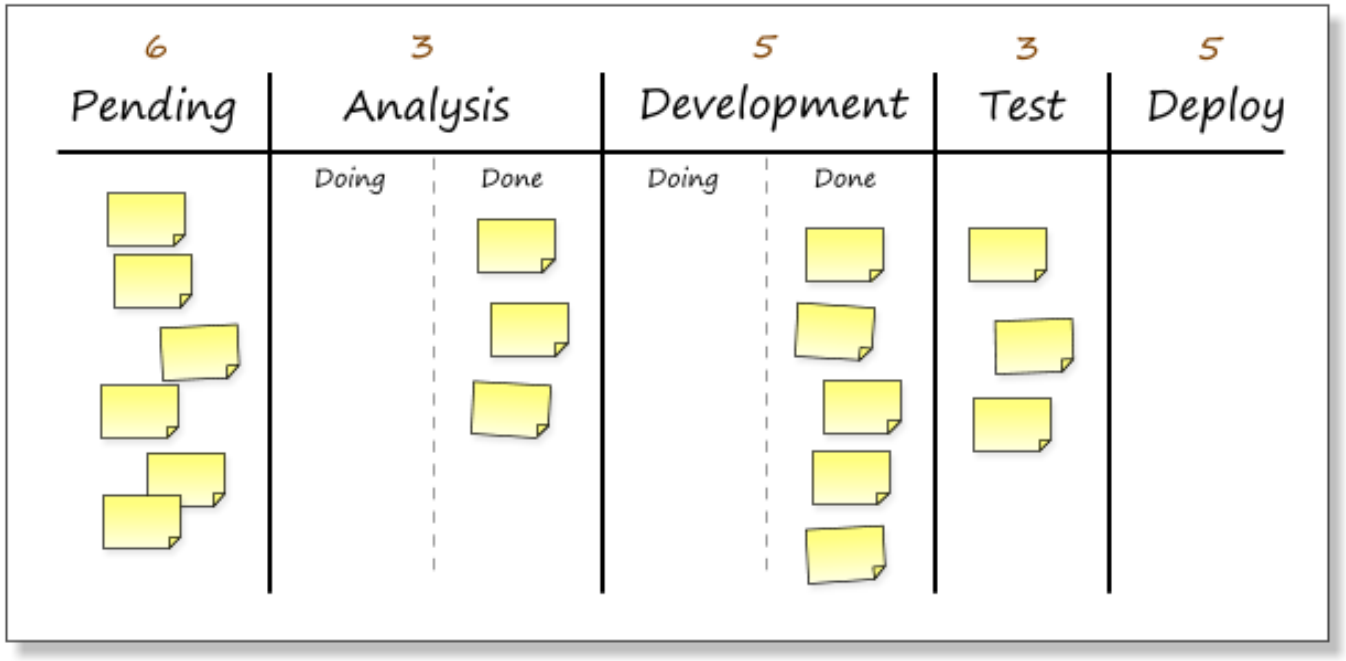
- http://kanbanblog.com/
- http://leankit.com/learn/kanban/
- http://www.everydaykanban.com/

The key elements of Kanban:

- **Visualizing the workflow** using Kanban board by breaking down the workload to smaller tasks, putting them on Kanban Board and moving them from one Board's section to another as they progress.
- **Limiting WIP (work in progress)** – defining specific limits of how many work items are allowed be in progress at each workflow state.
- **Measuring the lead time** (average time to complete one item) and improving it by bringing changes to existing process.

Kanban Board

Kanban Board is a central element of entire methodology:



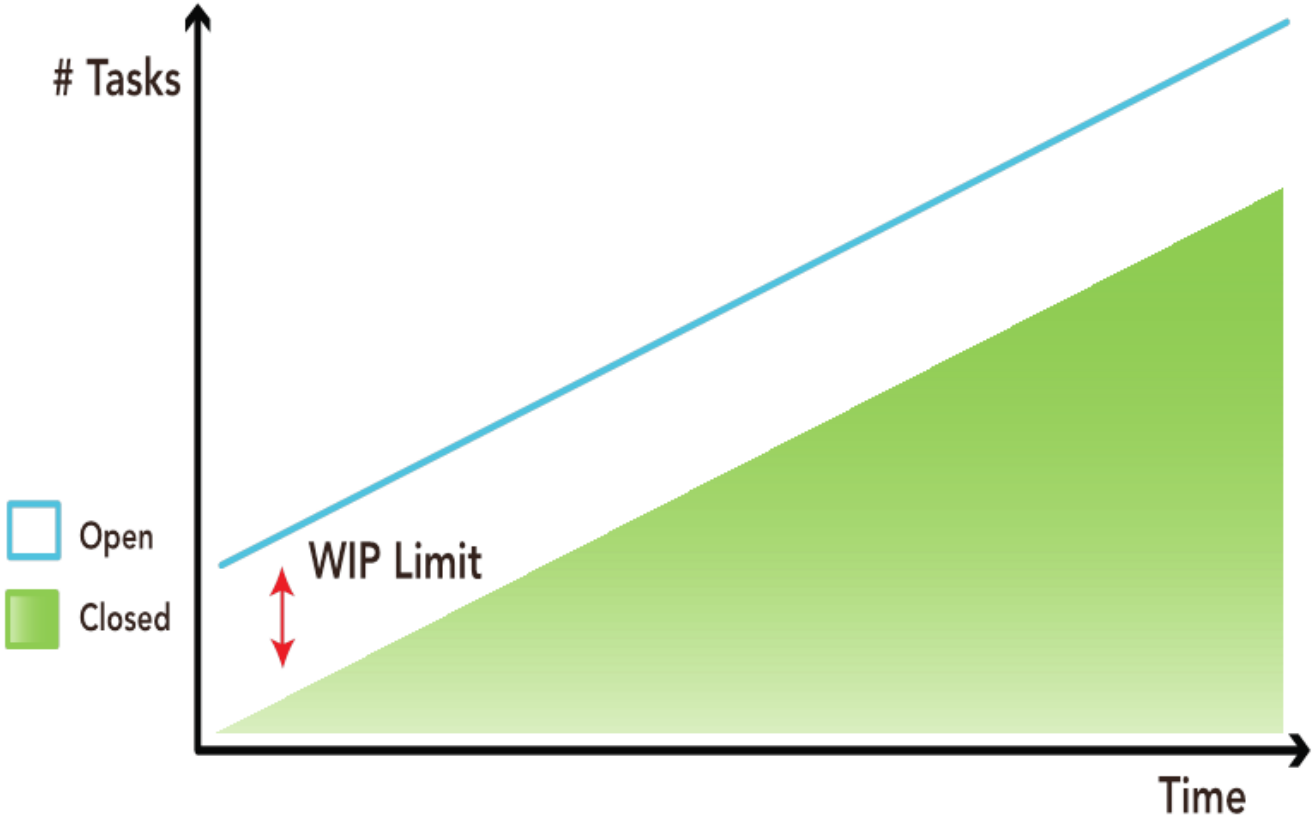
How Kanban works?

- Value flow is divided into smaller **Work Items**. These **Work Items** are placed onto **Kanban Board** in TODO section.
- Then **Work Items** are being worked on and subsequently they "travel" through the sections of **Kanban Board** until they are *done*.
- At the same time **WIP limits** are indicated for every section of the **Kanban Board**. Of if the **WIP limit** is reached on any section - no additional items can be placed into this section. When it becomes a problem for a team - blocked Team members help finish those **Work Items** which are causing the bottleneck.
- This way **value stream** will not get stuck on some stage - and **value** will relatively quickly flow to the consumer.

After the Kanban process is established, more improvements can be done:

- Team can play with WIP limits
- For planning and predictability team can borrow some practices from Scrum or from Project Management domain;
- Also some other practices can be adopted, for example engineering practices from XP.

Idealized Kanban Process



Kanban vs. SCRUM

- SCRUM is **more prescriptive** than Kanban, which means it has more rules to follow.
- SCRUM **requires a set of roles and mandatory meetings**, which Kanban is free from. However, many Kanban teams choose to keep some of the roles and/or meeting if they add value to their process.
- SCRUM mandates **timeboxed iteration**, which Kanban says nothing about.
- Kanban **limits WIP per workflow state**, SCRUM **limits WIP per Iteration**.
- Kanban does not require from **Team to be cross-functional**.
- SCRUM **does not accept change within an Iteration**, while Kanban not having prescribed Iterations has a choice how to handle the change.
- SCRUM **Board is reset between each Iteration**, which is not necessary in Kanban.
- SCRUM **Backlog Items must fit in an Iteration.**
- SCRUM prescribes **estimation** and **velocity**. In Kanban, estimation is not prescribed
- SCRUM requires a prioritized **Product Backlog**
- In SCRUM, **Burndown Charts** are mandatory

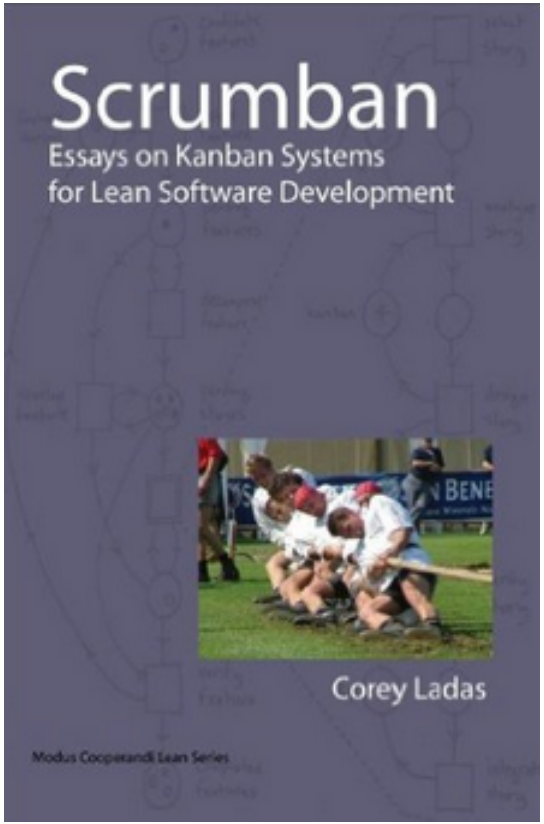
Scrumban

Scrumban

Scrumban is a methodology that provides a middle ground between SCRUM and Kanban. It mixes the **structure of SCRUM** and **loose planning of Kanban** which makes it suitable for for projects in **fast changing environment**.

Many experts even don't consider Scrumban to be a development process in and of itself. But rather a **process of taking existing SCRUM process and making it more Lean**.

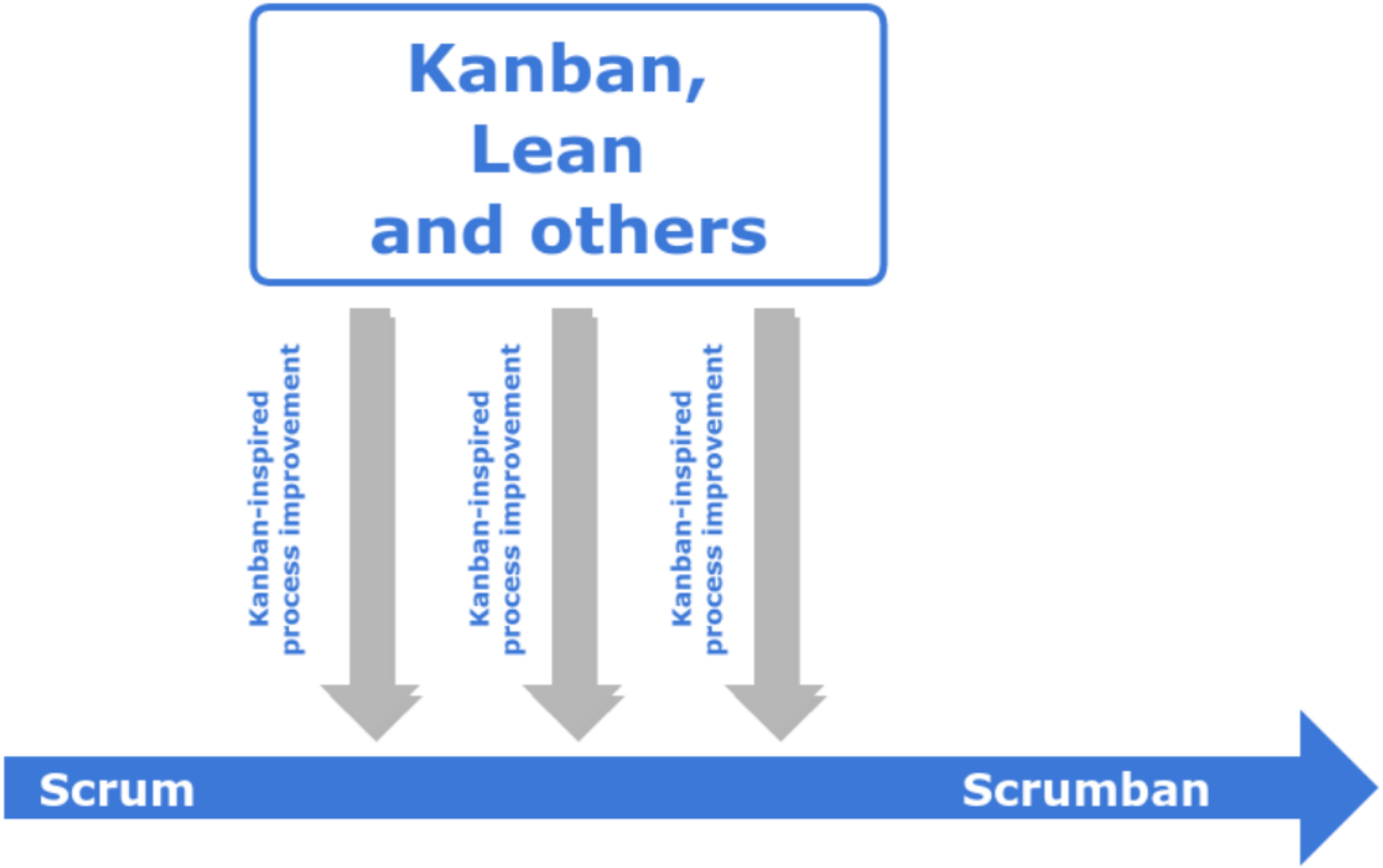
Much more about Scrumban can be found in Corey Ladas' book "**Scrumban - Essays on Kanban Systems for Lean Software Development**" available on Amazon.



How to get to Scrumban

Basic algorithm:

- Team starts with a **SCRUM or SCRUM-like process** in place.
- If not done previously, the **value flow is visualized** using the board.
- Very often though not mandatory, **limiting WIP** is applied and tuned throughout the project length.
    - One option for this would be size **limit for the Iteration Backlog**. This means that Backlog is always the same size for each Iteration. E.g. 20 software defects. Or 32 IT support requests.
- Continue **managing the process** flow by:
    - Evaluating Scrum required rituals/roles/artifacts and changing them if this brings more value. Even if after these changes the process stops being SCRUM per se.
    - Applying other useful techniques/rituals from Kanban, Lean, XP or other frameworks, or even from areas that have more to do with Project Management and Program Management domains.


Kanban,
Lean
and others
Kanban-inspired process improvement
Kanban-inspired process improvement
Kanban-inspired process improvement
Scrum
Scrumban

XanPan

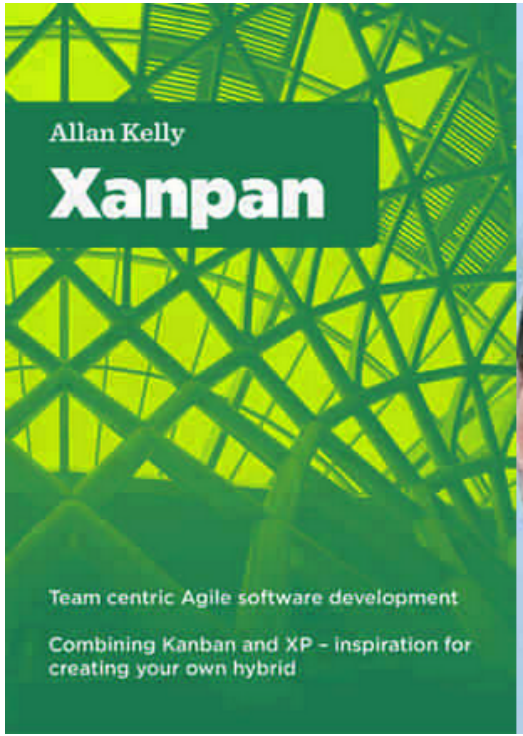
XanPan

Xanpan is a fairly new Agile software development methodology created and promoted by Allan Kelly.

His Book available at https://leanpub.com/xanpan

Essentially it is a **mix of Kanban and XP** methodologies with Product Management and some other practices.

Alan's video presentation about Xanpan: http://www.infoq.com/presentations/xanpan



XanPan elements

From Extreme Programming: The technical practices

- TDD
- Code review and pair programming
- Continuous integration
- Rough up front design
- Refactoring
- Shared code ownership
- Minimal documentation
- Velocity measurement

From Kanban: Process flow

- Kanban board
- WIP limits
- Continuous improvement
- Cumulative flow charts

Scrum: Process rhythm

- Iterations
- Set of meeting in the beginning/end of the Iteration
- Burn-down charts
- User Stories

From Lean: Culture of improvement, Kaizen and Learning

- Leaders' commitment to improvement
- Team Coach
- Multi Skilled team members together with specialists

XanPan elements

Xanpan doesn’t take:

- Scrum Master role
- Anti-manager ethos
- Kanban’s stop the line quality control

Something Xanpan doesn’t like but doesn’t outlaw (because it can’t):

- Matrix management
- Distributed teams, particularly teams in different timezones

Underlying Xanpan philosophy:

- **Quality is free:** automate as much testing as you can
- **Prioritisation is vital** and omnipresent: there is no such thing as too little time, just mis-understood priorities
- **People are key** to good software development :we need to grow more good people
- You can **always improve**
- Customer/End **user involvement** is key
- **Xanpan is a pick-n-mix of bits of various development methods** and adopters are encouraged to continue the approach
- **No process or Methodology can cope with all situations**, so if you are doing Xanpan the same in six months as you do today you aren’t doing Xanpan

Lean Startup

Lean Startup

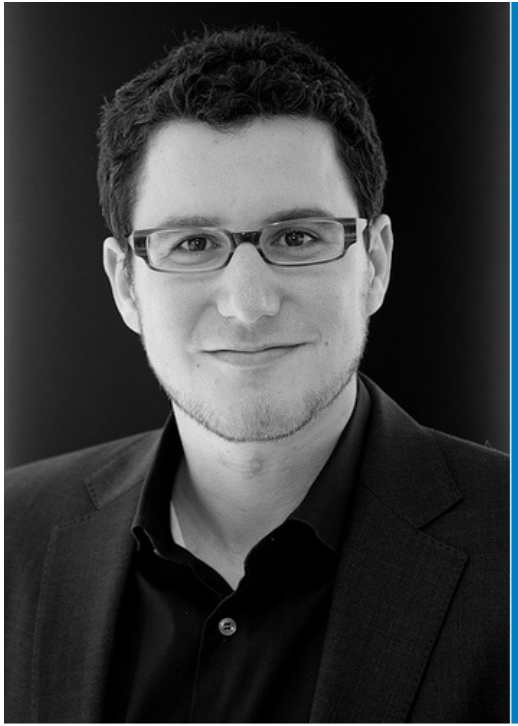
Lean Startup book

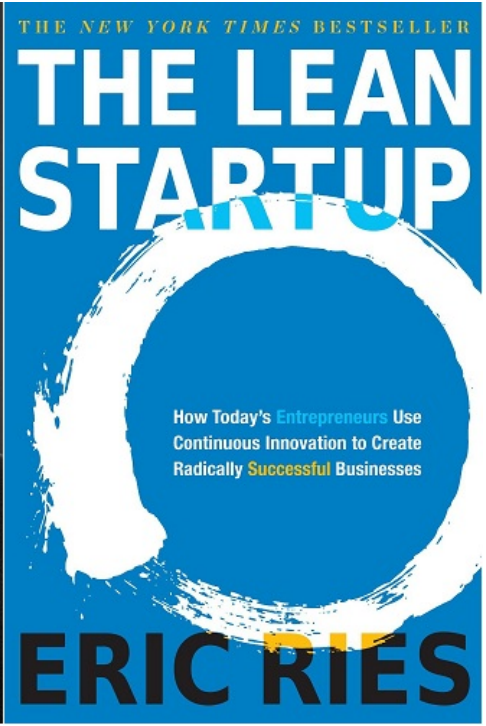
The concept of Lean Startup was presented by **tech entrepreneur Eric Reis** in his book, *“The Lean Startup: How Today’s Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.”*

While originally geared toward technology companies, the **idea has expanded to encompass nearly all industries**.

You can find the book by the following link: http://theleanstartup.com/book,

and visit Eric’s blog here: http://www.startuplessonslearned.com/.





Lean Startup

Idea

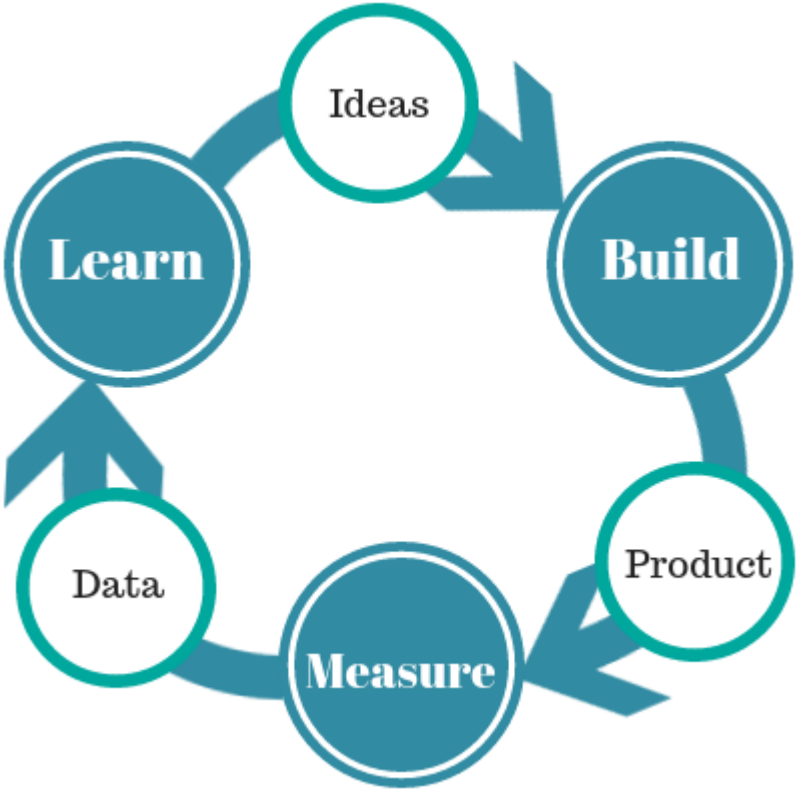
The main idea is that nowadays the problem in software development is not about building the Products **right way**, but rather in building the **right Products**.

Eric Ries claims that startups can shorten their product development cycles by adopting a combination of business-hypothesis-driven experimentation, iterative product releases, and what he calls **"validated learning"**.

Proces

The Lean Startup process is known as the **"build-measure-learn" feedback loop.**

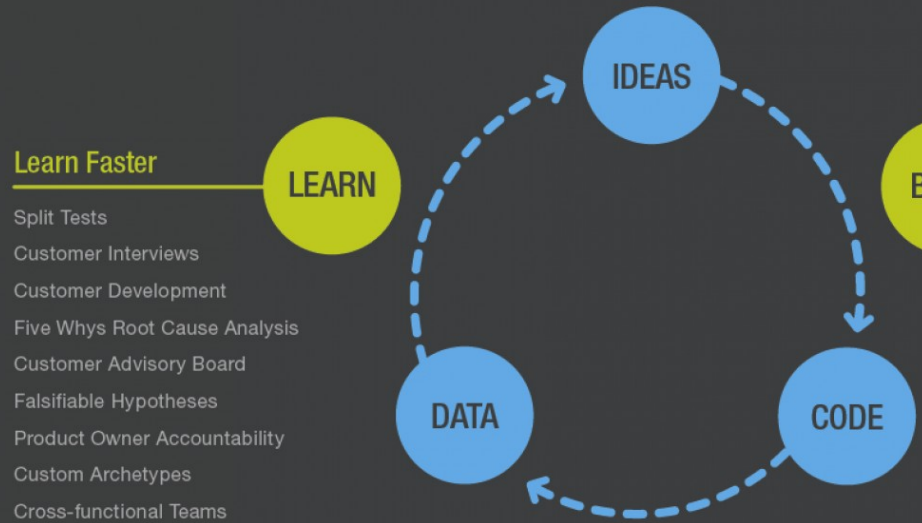
It encourages the entrepreneur to **get a minimum viable product (MVP) into customers' hands** as soon as possible, then test it, learn from it, and **refine it into a finished product**. Doing this allows the business to learn what customers want and to focus its development efforts accordingly.



THE LEAN STARTUP

Created by Eric Ries - startuplessonslearned.blogspot.com

Designed by KISSmetrics



Learn Faster

- Split Tests
- Customer Interviews
- Customer Development
- Five Whys Root Cause Analysis
- Customer Advisory Board
- Falsifiable Hypotheses
- Product Owner Accountability
- Custom Archetypes
- Cross-functional Teams
- Smoke Tests

Code Faster

- Unit Tests
- Usability Tests
- Continuous Integration
- Incremental Deployment
- Free & Open-Source Components
- Cloud Computing
- Cluster Immune System
- Just-in-time Scalability
- Refactoring
- Developer Sandbox

Measure Faster

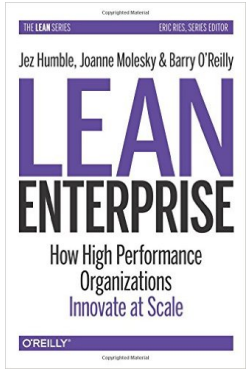
- Split Tests
- Clear Product Owner
- Continuous Deployment
- Usability Tests
- Real-time Monitoring
- Custom Liaison
- Funnel Analysis
- Cohort Analysis
- Net Promoter Score
- Search Engine Marketing
- Real-Time Alerting
- Predictive Monitoring

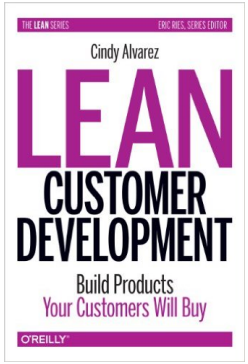
Lean Startup

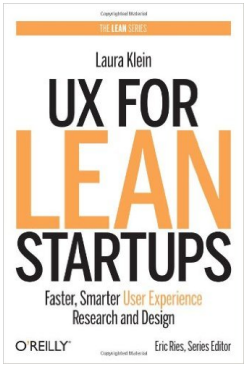
Other Lean Startup books

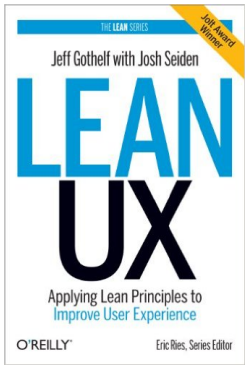
Lean Startup movement inspired a series of books explaining how to apply Lean principles in other areas of business:

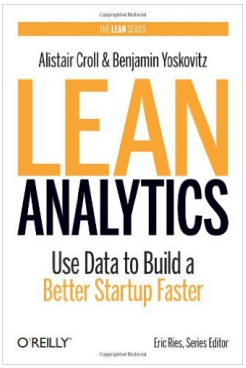
- Lean UX
- UX for Lean Startups
- Running Lean
- Lean Analytics
- Lean Enterprise
- Lean Customer Development
- Lean Branding

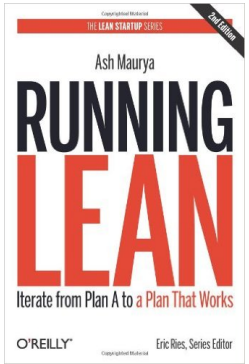


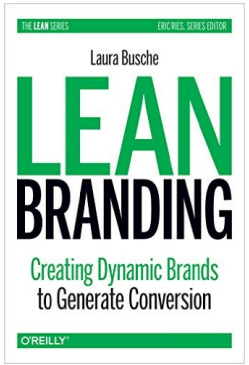












Agile for multiple teams and organizations

NEXUS Framework

NEXUS Framework

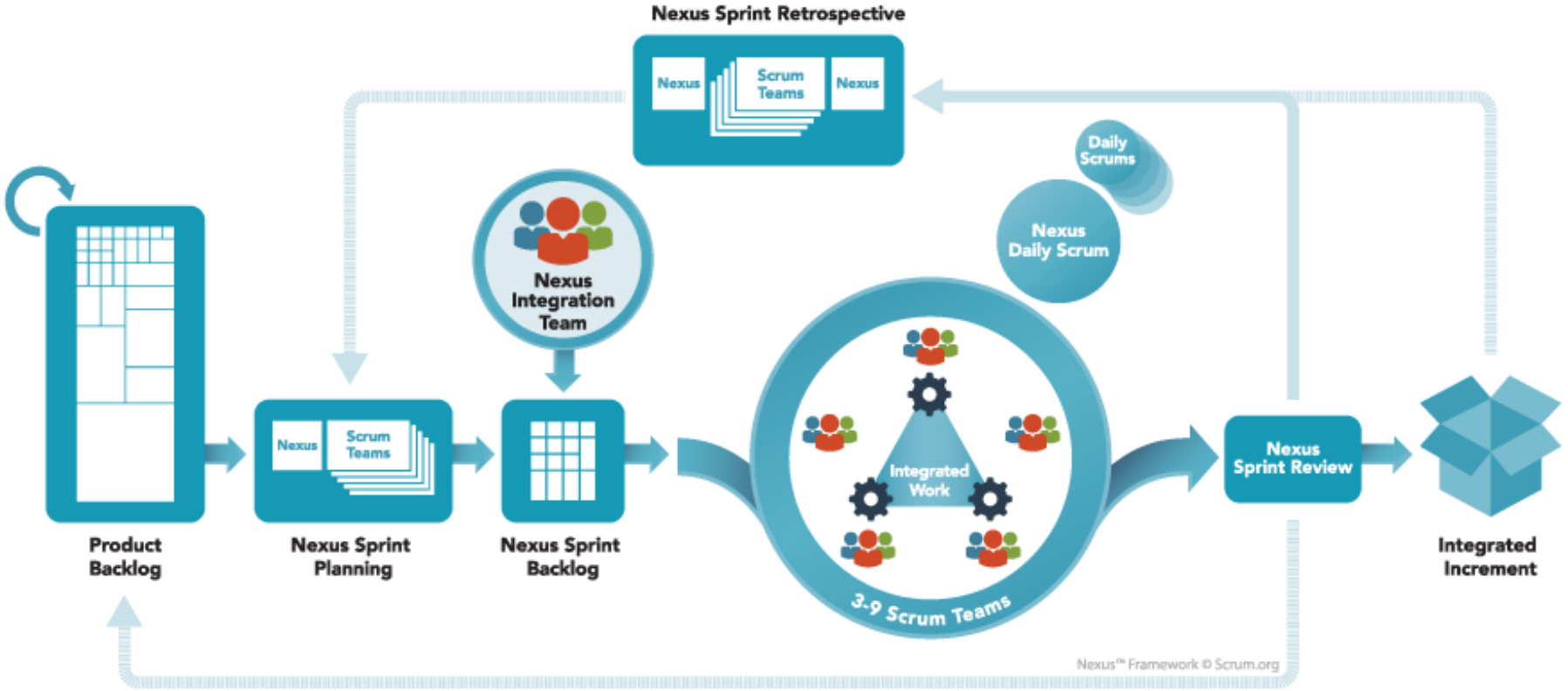
The framework for scaling and managing large projects from www.scrum.org.

TeamScaled Professional Scrum consists of the Nexus™ Framework and approximately 40 practices which cause the Nexus and your scaling initiative to operate predictably. It builds on the existing Scrum framework and values. The result is an effective development group of up to 100 people using best industry practices.

Nexus Benefits

- Organizes teams to maximize their productivity
- Organizes people into right teams so efforts are optimized
- Shows managers how to organize and manage large number of teams to rapidly build software
- Helps managers detect anomalies in productivity
- Provides practices for addressing them
- Presents patterns that enable self-organization of larger number of developers

NEXUS FRAMEWORK: FOR SCALING AND MANAGING LARGE AGILE PROJECTS



Scaled Agile Framework®

SAFe

The Scaled Agile Framework® (called SAFe®) is a roadmap for adopting Agile at organizational level.

There are three levels in SAFe:

- Team
- Program
- Portfolio

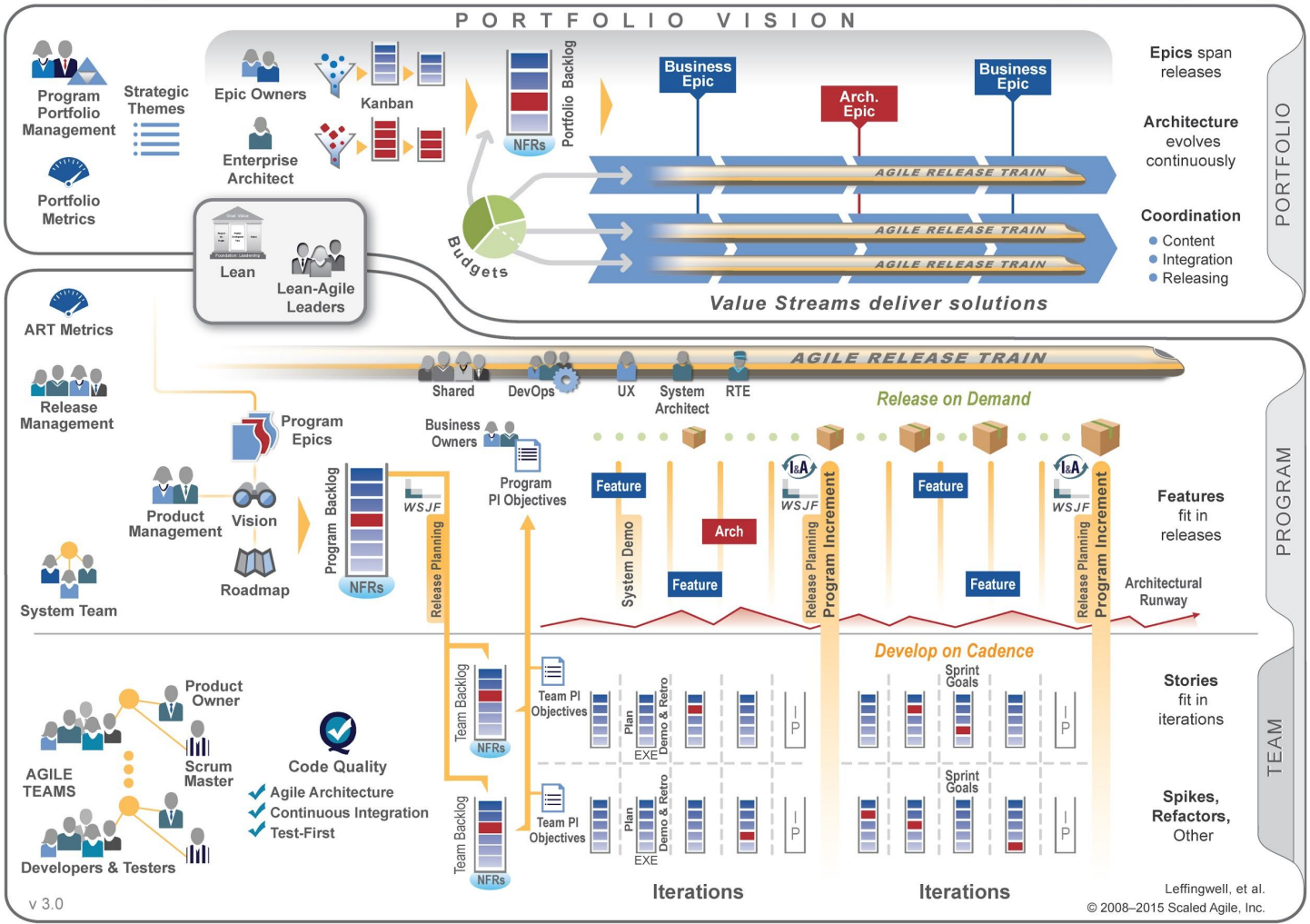
SAFe works with tough issues like the following at organizational scale:

- architecture,
- integration,
- funding,
- governance and
- roles.

More about SAFe: http://www.scaledagileframework.com/

Scaled Agile Framework®

SAFe®



SAFe

Team Level

- Scrum with XP engineering practices are used. Kanban or Scrumban also possible.
- Ideally there are five to nine members of each team.
- Teams deliver working, fully tested software every two weeks.

- **7+/- 2 Members**, collocated
- **Self-managing:** No managers on the team
- **Scrum Master:** A full or part-time role for a team member, or a scrum master may be shared across 2 – 3 teams
- **Product Owner:** A team has only one product owner, who may be dedicated to 1 or 2 teams

SAFe

Program Level

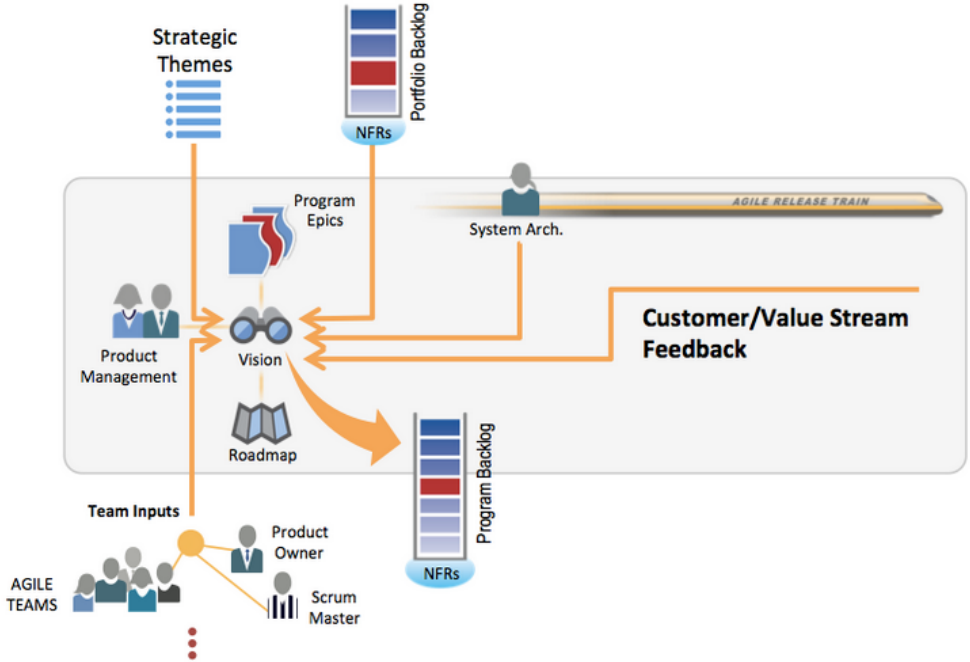
SAFe defines an **Agile Release Train (ART)** which are like Iterations but on organizational level. ART delivers a value stream for the organization.

Every Potentially Shippable Increment (PSI) contains items from Product Backlog(s). They are defined as hierarchy of Epics, Features and User Stories.

Between **5 and 10 teams** work together on a ART. They synchronize their release boundaries and their iteration boundaries.

Regular synchronization happens on the **Scrum of Scrums meeting**.

Every **10 weeks (5 iterations)** a ART delivers a PSI. After which demo and retrospective sessions are held, and planning begins for the next PSI.



SAFe

new **Program Level** Roles

- **System Team**
- **Product Manager**
- **System Architect**
- **Release Train Engineer** (RTE) or simply main ScrumMaster of ART
- **UX** and **Shared Resources** (e.g., security, DBA)
- **Release Management Team** (cross-functional team - with representation from marketing, dev, quality, ops and deployment – that approves frequent releases of quality solutions to customers)

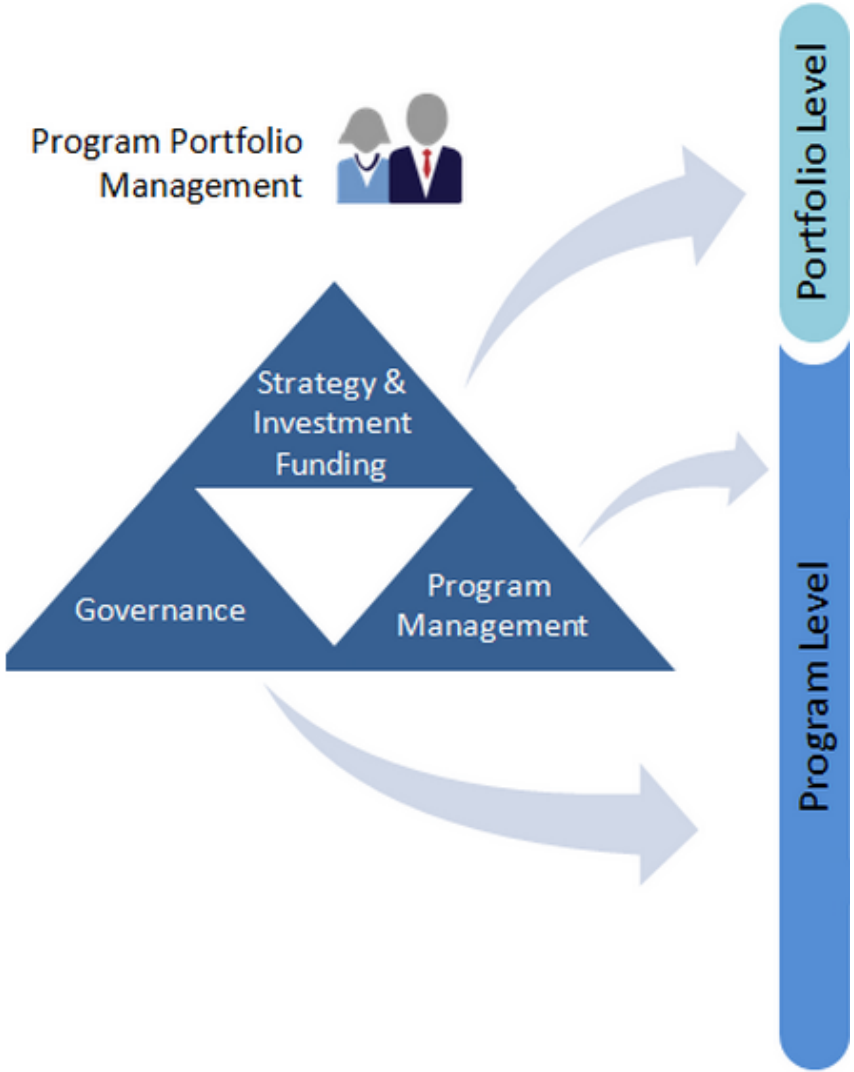
SAFe

Portfolio Level

The highest level of the SAFe is the Portfolio Level, where programs are aligned to the enterprise business strategy along **Value Stream lines**.

On this level **themes** for Agile Release Trains are defined and corresponding budgets are allocated.

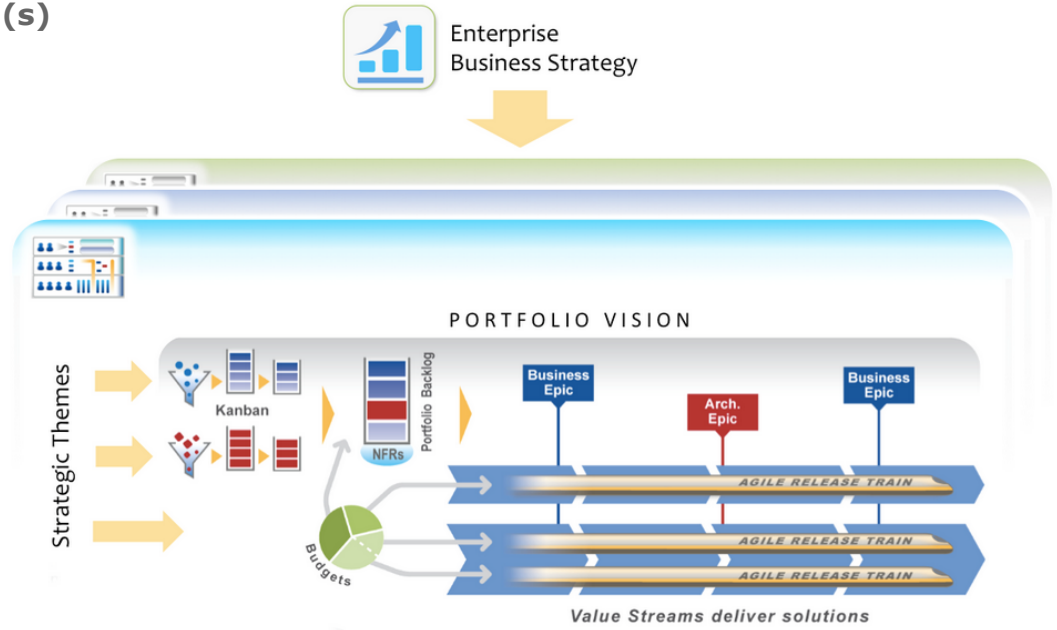
Kanban system is used **to insure the limit** of Portfolio Items and finishing Items which are "in progress" before starting new ones.



SAFe

The Portfolio Level contains **Portfolio Vision(s)** which consists of:

- Strategic Themes
- Program Portfolio Management
- Value Streams
- Budgets
- Epics
- Portfolio Backlog
- Kanban Systems
- Epic Owners
- Enterprise Architect
- Portfolio Metrics



Thats it. This Agile overview is over. Time to dive in :)

AUTHOR: Dzvinka Mytsyk

Email: dzvinka.mytsyk.contact@gmail.com
LinkedIn: http://www.linkedin.com/in/dzvinkamytsyk