LEANDOG AGILE

Discussion Guide
4th Edition

To build cultures and products that inspire and delight

LEANDOG AGILE

Discussion Guide
4th Edition

LeanDog, Inc.
1151 North Marginal Road
Cleveland, Ohio 44114

TABLE OF CONTENTS

PREFACE

This guide is a comprehensive introduction to modern product development practices and thinking. While we have had many customers ask us for a “checklist on Agile”, every team is different and no two teams practice the same way. This guide is meant to serve as a blueprint, not a roadmap. You’ll find outlined principles, concepts, and practices by role to help guide you to better ways of working.

Each practice falls into one of two categories: essential and advanced. Essential practices are core to adopting Agile while Advanced help as your team matures.

For each role on an Agile team, you’ll find practices mapped to each craft. You don’t have to implement every practice, but each practice adds value. We encourage you to explore practices, try them, and evaluate them yourself.

In the spirit of continuous improvement, this guide is evolving as we uncover better ways of working as modern product teams. Subscribe to the mailing list to stay up to date with the latest version - leandog.com/resources.

To build cultures and products that inspire and delight

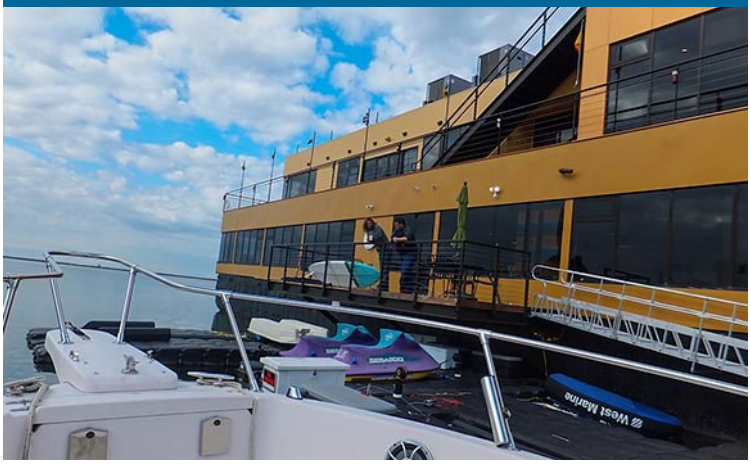
ABOUT LEANDOG

LeanDog was founded in 2008 with a simple mission: to build cultures and products that inspire and delight. For over ten years our team of coaches, trainers and developers have done just that. Our lean and agile experts work with change agents in a variety of industries and disciplines to instill the values and behaviors that drive innovation. Whether we're leading agile transformations, delivering software or training leaders on agile concepts, we remain committed to taking a people-first approach to process improvement.

We're proud to be known as the **"company on the boat."** Our offices are located on Barge No. 225, an 1892 steamship on the shores of Lake Erie in Cleveland, Ohio. The boat is a big part of our company culture. We take pride in our offices being a weird, quirky space that encourages outside-the-box thinking. Our offices are home to a host of Northeast Ohio technology meetups and events. We remain committed to supporting a thriving tech community in Cleveland and beyond.

We've worked with over 200 client organizations of all sizes, from the Fortune 100 to start-ups and small businesses. This Agile Discussion Guide has served as our playbook and the foundational document for our client engagements since it was first published in 2010. Creating a shared understanding of concepts is the first step toward building a successful and sustainable agile culture. We hope that this book helps to set a baseline understanding of the tools in the agile toolbox.

LEANDOG HEADQUARTERS




Our highest priority is
to satisfy the customer

CHAPTER 1

Introduction

IN THIS CHAPTER

- Agile Manifesto
- Principles Behind the Agile Manifesto
- Agile Practices
- Practices Mapped to Agile Values
- Agile History
- Agile Culture & Process Case Study: IDEO
- Oath of Non-Allegiance
- Company Ecosystem
- Agile Benefits Realized

AGILE MANIFESTO

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

- **Working Software** over Comprehensive Documentation
- **Customer Collaboration** over Contract Negotiation
- **Individuals and Interactions** over Processes and Tools
- **Responding to Change** over Following a Plan

That is, while there is value in the items on the right, we value the items on the left more[1].

AGILE MANIFESTO HISTORY

In 2001, a group of 17 people who are passionate about simple and minimalistic software practices met in Snowbird, Utah to discuss their approaches and best processes. These representatives from eXtreme Programming (XP), Scrum, DSDM, Adaptive Software Development and others "sympathetic to the need for an alternative to documentation driven, heavyweight software development process" uncovered better ways of developing software by doing it and helping others do it.

Through collaboration, the Agile Manifesto was created outlining four value statements and 12 principles. While most people only read the values, it is important that the principles are understood as well. The values define the overall objectives of the Agile methodology, making up the "WHY in Agile", while the principles define the tenets that feed into the values, becoming the "WHAT in Agile". Finally, the practices serve as the "HOW in Agile", providing the blueprint to follow [1].

AGILE MANIFESTO

THE 17 AUTHORS OF THE AGILE MANIFESTO:

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

vimeopro.com/leandog/manifesto

Agile Manifesto & Principles Poster
LeanDog.com/free/agilemanifesto.pdf

1 Kent Beck, et al 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 team 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.

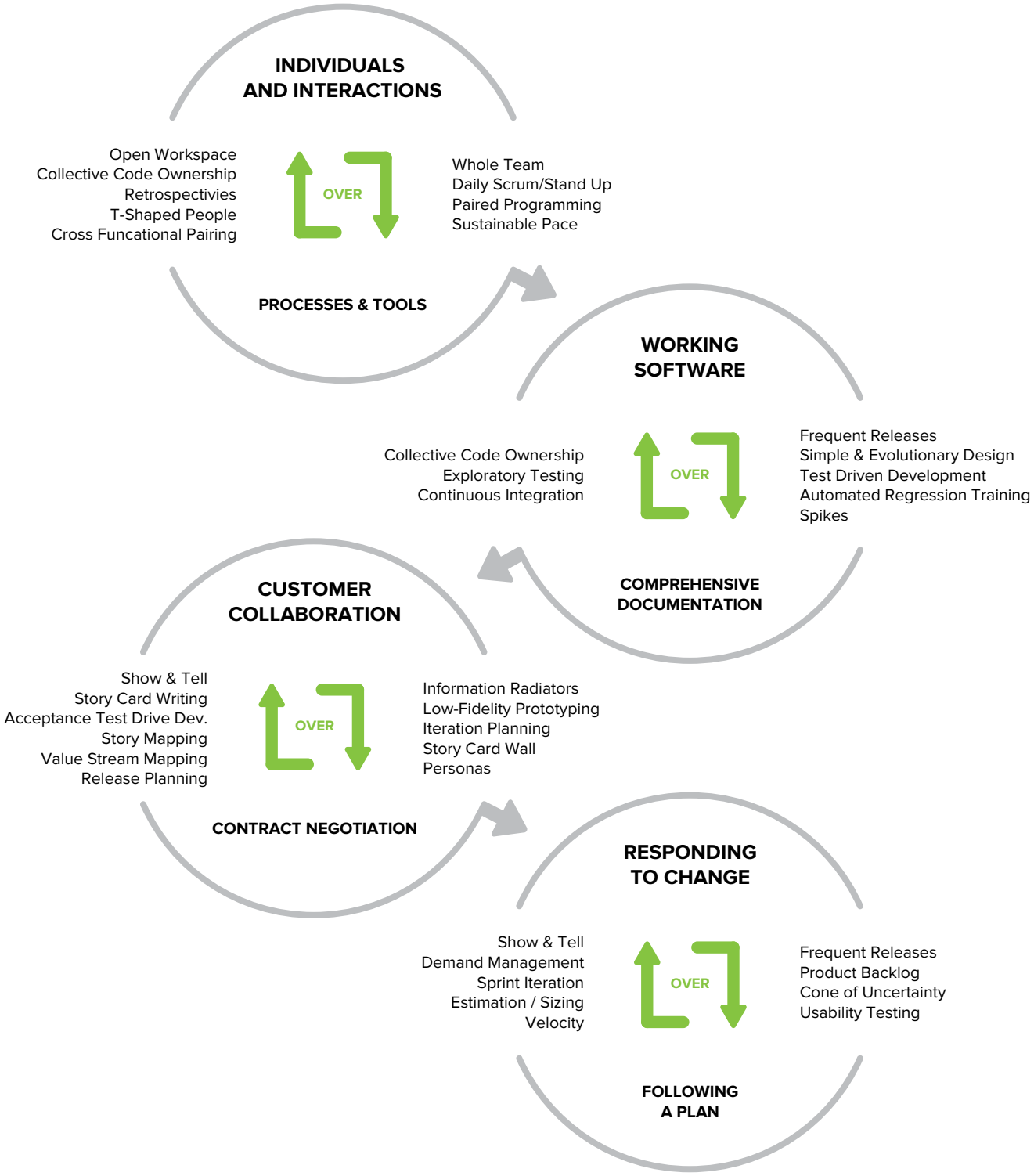
The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly[1].

1 Kent Beck, et al http://www.agilemanifesto.org

PRACTICES MAPPED TO AGILE VALUES

Practices Mapped To Agile Values



vimeopro.com/leandog/agilepractices

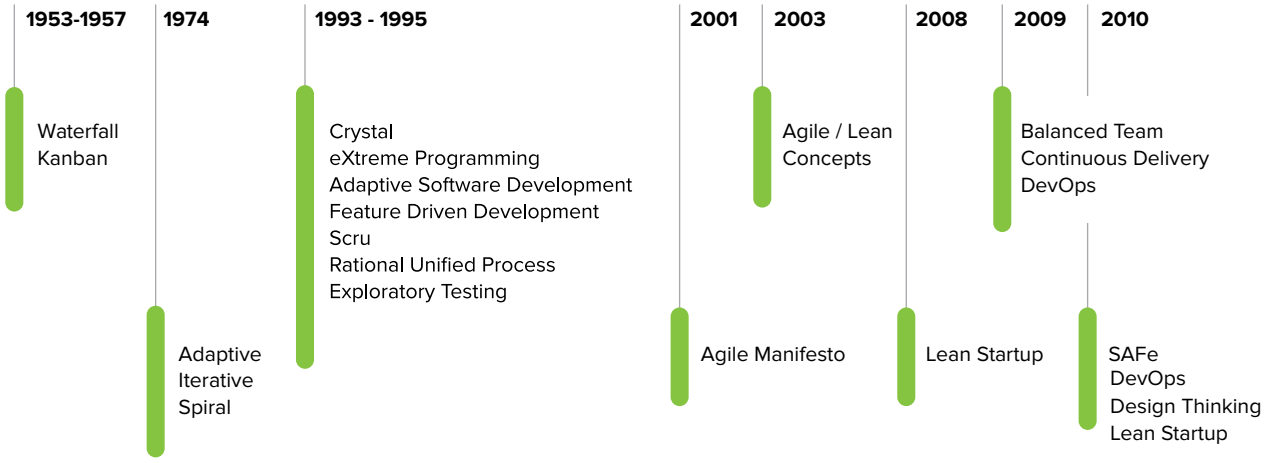
AGILE HISTORY

Incremental software development methods trace back to 1957. Lightweight software development methods first appeared in the mid-1990s in reaction to the heavyweight waterfall-oriented methods that had become commonplace, which critics called heavily regulated, regimented, micromanaged and over-incremental.

Proponents of these lightweight methods contended that they were returning to development practices that were present early in the history of software. These early implementations favored creativity, freedom, self-management and continuous improvement. These methods are now collectively referred to as Agile development after the Agile Manifesto was published in 2001.

The term "Lean" was coined to describe Toyota's Production System during the late 1980s by a research team at MIT. The core idea is to maximize customer value while minimizing waste. In the early 2000s, companies (especially startups) began applying both Lean and Agile principles together in order to develop new products (or even new companies) more efficiently and based on validated customer demand.

Agile History



AGILE CULTURE & PROCESS CASE STUDY: IDEO

The IDEO Deep Dive video on ABC Nightline is about a team at IDEO that reinvented the shopping cart in one week. While watching the video, ask yourself:

1. How does the process of designing a better product work?
2. What does a process and a culture look like?

Online at: youtube.com/watch?v=M66ZU2PCIcM [1]

IDEO
DEEP DIVE

INNOVATIVE CONCEPTS

- Fresh ideas come faster in a fun environment
- Similar ideas don't create innovation
- Being playful is important to innovation

PRACTICES

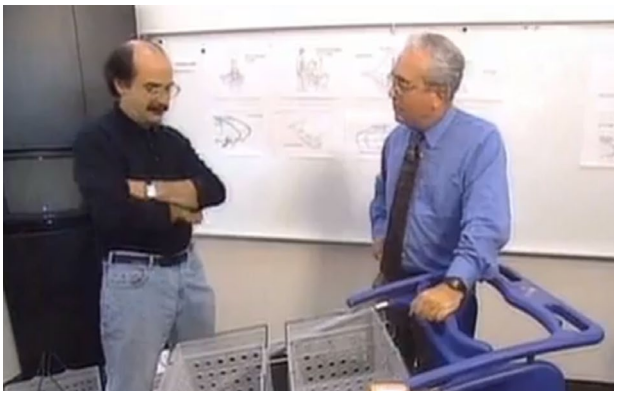
- Find experts to learn quickly, don't learn on your own
- People are encouraged to build their own work areas
- One conversation at a time, stay focused, signs everywhere
- Demonstrate, communicate, and share everything
- Not experts in any area, they are experts in a process
- Fail early and often to succeed sooner
- Don't put up with a failed process for too long
- Enlightened trial and error succeeds over planning

OUTCOMES

- Witnessed Culture, Process & Innovation
- Agile?
- What does your culture look like?

CULTURE / TEAMWORK

- Build a culture and a process
- The Boss always has the best ideas? Not likely!
- Open mind is crucial, boss encourages team work
- Try it without permission, ask for forgiveness later
- Status is from who has the best ideas, not tenure
- In an innovative culture, you can't have hierarchy
- Project leader must be good with groups
- No titles, no permanent assignments



1 ABC Nightline, IDEO Deep Dive, http://www.youtube.com/watch?v=M66ZU2PCIcM

OATH OF NON-ALLEGIANCE

"I promise not to exclude from consideration any idea based on its source, but to consider ideas across schools and heritages in order to find the ones that best suit the current situation."

An Agile approach focuses on empowered, self-managing teams; autonomy that doesn't need day-to-day intervention by management [2]. Instead, Agile management means protecting the team from outside interference and removing roadblocks that impede the delivery of business value and productivity. It is widely accepted that complex systems cannot be predicted [3] and are best managed using empirical process controls; therefore, management allows self-managing teams to build systems in an empirical manner [4].

WHAT DOES IT MEAN TO "BE AGILE?"

Considering that Agile is a set of values and principles, becoming Agile means upholding these values and principles. It's not about doing one practice (for example Scrum) and declaring that you are Agile. Instead, it's about continually seeking better ways of delivering value. We coach your company to adopt this culture and implement practices that are based on the needs of your teams and projects.

LEANDOG SIGNED THE OATH OF NON-ALLEGIANCE...SO SHOULD YOU

At LeanDog, we uphold the tenets of the Agile Manifesto by studying concepts like Lean, Scrum, Systems Thinking, eXtreme Programming (XP), Organizational Effectiveness, and most importantly, practicing and giving back to the Agile community. Once you have decided to adopt the Agile methodology, we hope that you too will sign the Oath of Non-Allegiance.

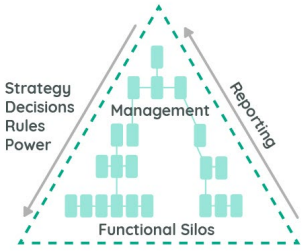
RECOMMENDED

vimeopro.com/leandog/oath

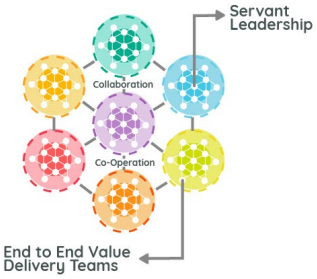
1 Alistair Cockburn, http://alistair.cockburn.us/Oath+of+Non-Allegiance 2 Kent Beck, et al http://www.agilemanifesto.org
3 Michele Sliger and Stacia Broderick. The Software Project Manager's Bridge to Agility, (Addison-Wesley: 2008)
4 Babatunde A. Ogunnaike and W. Harmon Ray. Process Dynamics, Modeling and Control. (NY: Oxford University, 1994)

BUSINESS AGILITY

The widespread adoption of agile methodologies within technology circles has spilled into other business units. Human resources, finance, marketing, sales, corporate governance...any organizational function can be positively affected by the iterative, transparent and value-driven cultures that agile practices support. In addition, many charged with leading transformations at scale are coming to understand that breaking down silos early and engaging stakeholders across the organization can be key to achieving sustainable change. Business agility is maximized when silos, hierarchy and bureaucracy are replaced are replaced with a flat organizational structure focused on empowered, cross-functional teams delivering value rapidly.



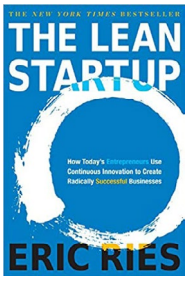
Traditional Organization



Agile Organization

Traditional Organizations	Agile Companies
Functional, silo-based structure, task delivery focus.	Team-based structure, combining different competencies, value delivery focus.
Command and control behavior.	Collaboration & teamwork emphasized.
Micro management, directive managers.	Servant leaders focusing on providing service.
Focusing on efficiency and operation.	Focusing on productivity and value.
Annual, big bang planning	Rolling wave, adaptive planning.
Think big, work on details, deliver the perfect.	Think big, start small, evolve continuously.
Bottom up reporting, decisions driven by assumptions and forecasts.	Radical transparency, decisions driven by data.
Long approval procedures and need for authority approval.	Empowered teams.
Limited tolerance for mistakes.	Based on experimentation and learning from mistakes.
Mechanical and cumbersome organizational structure.	A living organizational structure that changes shape in line with the company's strategies.
Focusing on individual performance.	Focusing on team-based performance within the scope of business objectives.
Investor oriented.	Employee and customer oriented.

RECOMMENDED



"The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses"
By Eric Ries

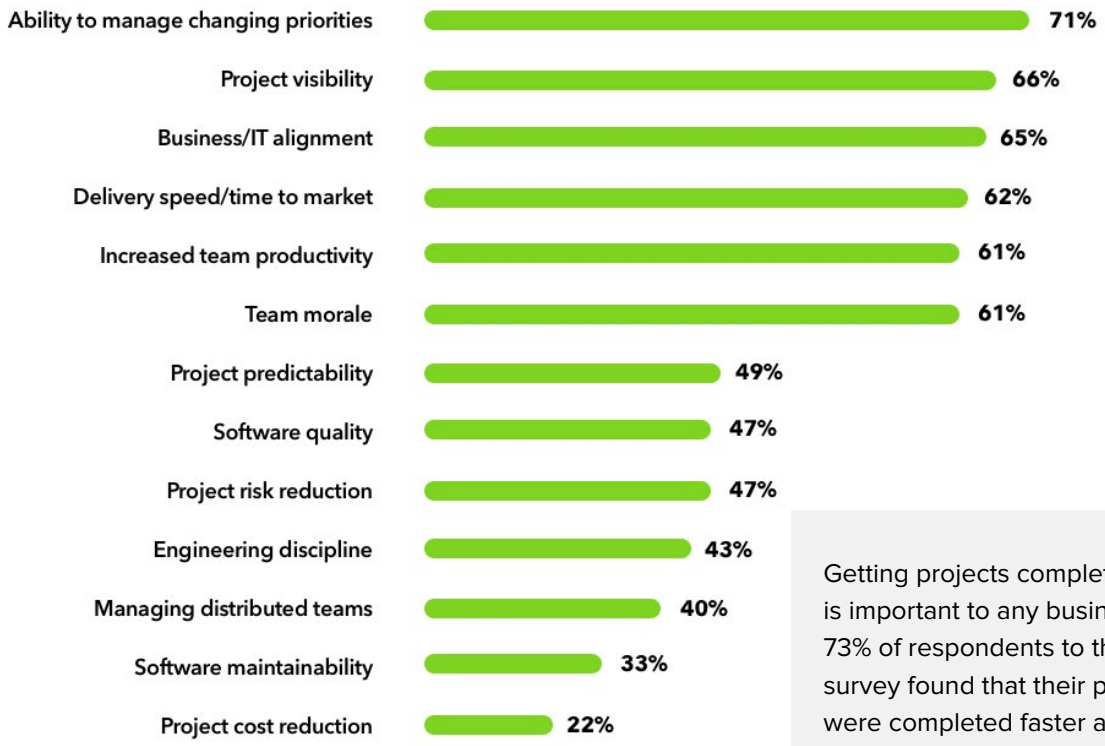
Source: Scrum.org, "Business Agility" www.scrum.org/resources/blog/business-agility

AGILE BENEFITS REALIZED

Since 2007 CollabNet | VersionOne has conducted yearly surveys on the state of Agile across industries and the globe. Consistently the top reasons why companies and organizations adopt Agile are to accelerate software delivery, better manage changing priorities, increase productivity, and many other improvements. The top benefits realized are the ability to manage changing priorities, increase project visibility, and align Business and IT.

BENEFITS OF ADOPTING AGILE

By implementing agile, respondents cited seeing improvements in the following areas:



Getting projects completed is important to any business, 73% of respondents to the survey found that their projects were completed faster after adopting Agile.

The challenges and barriers to adopting and scaling Agile are almost always cultural. You may find resistance to change, inconsistencies in process and practices, and lack of skills and experience with agile methods. This is common but can be overcome with practice and embracing a continuous improvement mindset.

RECOMMENDED

- **Source: CollabNet | VersionOne 12th State of Agile Survey 2018**
  https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report

Whole Team is the glue of Agile practices

CHAPTER 2

Concepts

IN THIS CHAPTER

- Lean-Agile
- Whole Team
- Team Agreements
- Open Workspace
- T-Shaped People
- Sustainable Pace
- Information Radiators
- Clear Communication
- Frequent Releases
- Story Card Wall
- Agile Triangle
- Cone of Uncertainty
- Design Thinking

LEAN - AGILE

Agile and Lean go hand-in-hand for modern software development teams. Agile is about fostering collaboration and communication while Lean is about eliminating waste and focusing on delivering the most value at just the right time.

The term Lean came out of lean manufacturing with the model Toyota Production System pioneered to streamline production and improve delivery. Mary and Tom Poppendieck correlated how Lean principles drive software development in their seminal book “Lean Software Development: An Agile Toolkit”. **Those driving principles are:**

- See the whole
- Embrace optionality
- Deliver effectively
- Amplify learning
- Empower people
- Build integrity in
- Eliminate waste

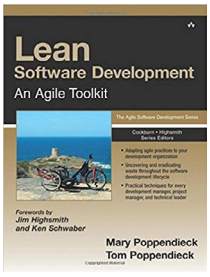
In 2011, Eric Reis took these concepts a step further with Lean Startup that emphasized tight feedback loops to learn and adapt as quickly as possible. The phases of the feedback loop are Build, Measure, Learn. More and more development teams are folding Lean Startup into the way they practice.

There is a global movement of people called Balanced Team. A collective of people who value collaboration, iterative delivery, and finding new ways to work well together as happy teams. Many of the folks are pioneering the next wave of processes and methodologies that innovate teams and deliver value.

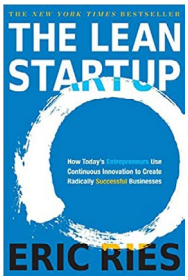
Building software using Lean-Agile is a “team sport” and it encourages exploration to find better ways of working together.

vimeopro.com/leandog/agile-concepts

RECOMMENDED



"Lean Software Development: An Agile Toolkit"
By Mary Poppendieck, Tom Poppendieck



"The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses"
By Eric Ries

WHOLE TEAM

The Whole Team approach states that everyone is accountable for the quality of the product. This approach brings the entire team together to work as a unit and share responsibility for producing high-quality software. Whole Team is the glue of Agile practices; it holds all the other practices together.

ESSENTIAL

- Whole team shares a workspace, whether co-located or remote, sitting with each other encourages collaboration and open communication
- Team members are 100% dedicated to the team's work and not allocated to other work outside the team
- Work comes to the team, the team doesn't form based on the funding of projects
- Avoid shared talent between teams. DBAs, QAs, SecOps and other technical specialists are available on demand but are preferably T-Shaped and on the team
- Cross-functional pairing is encouraged
- Team is led, not managed
- Team communicates, collaborates and improves continuously
- Team is accountable for results

ADVANCED

- Business partners are co-located with the team
- Team is not named after the project or department, instead they have their own team identity (example: Team Red Fish, Team Sweet Maple Bacon, Team CatDog)
- Remote pairing through collaboration tools that allow video conferencing, screen sharing, chat and more (tools like Zoom, Slack, Sococo)
- Keep communication in distributed teams highly effective by using virtual whiteboards, collaborative design tools, and collaborative project management tools (example: iObeya, UXPin, Trello)

RECOMMENDED

"The Art of Agile Development"
By James Shore
(Chapter 3)

WHY TEAMS?

Organizational boundaries typically increase cost by over 25%, creating buffers that slow down response time and interfere with communication [1].

vimeopro.com/leandog/whole-team

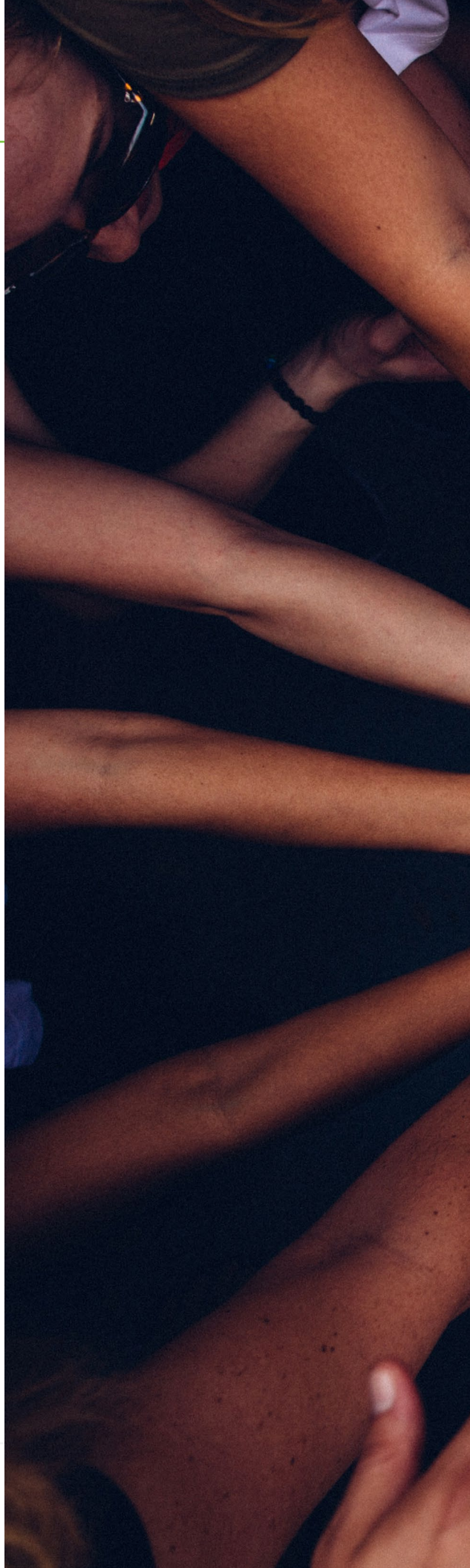
1 Mary Poppendieck, Poppendieck LLC

TEAM AGREEMENTS

The Team Working Agreements are important to establish when the team forms so that everyone on the team understands and agrees to the guidelines on how they must work together. These agreements are also known as Team Norms.

ESSENTIAL

- Everyone on the team contributes to defining the disciplines and behaviors the team should follow
- Pick one day a week, perhaps after the Friday daily Stand-Up to clean the work area for 15 minutes
- Music is OK and encouraged but the team agrees on the rules for it (example: anyone can change the music, it's on in the morning only)
- If something isn't working then members voice concerns at the retrospective - it's the team's responsibility to reflect and adapt to find the best way to work together
- All rules should be prominent in the team space




Everyone on the
team contributes

OPEN WORKSPACE

Face-to-face communication is extremely valuable. In an Open Workspace the team uses open seating to facilitate communication, shorten feedback loops, and foster ideation and collaboration. It is the hardest practice to implement but transparency and increased communication make it the most valuable.

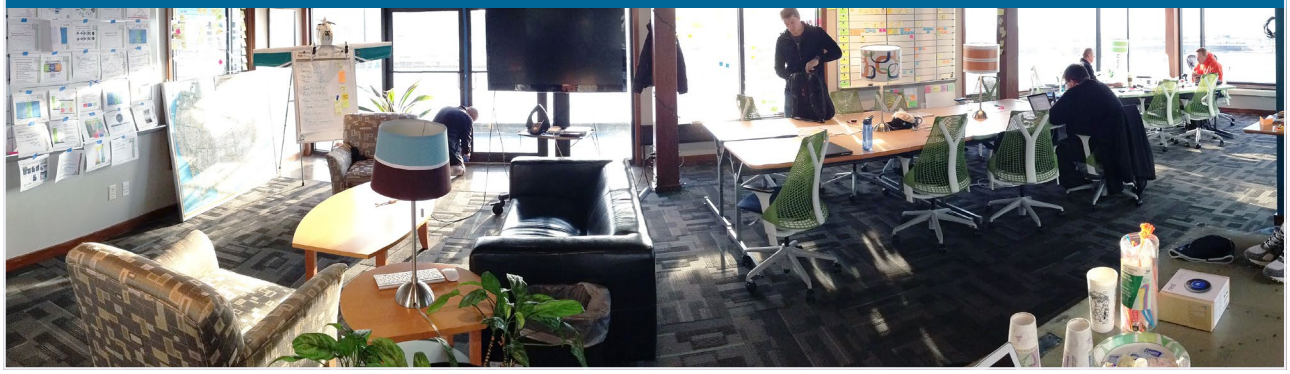
ESSENTIAL

- General rule is 150 square feet per team member with adequate space for moving, sitting, collaborating
- There are no assigned seats
- Teams do not have other personal space (like cubicles)
- Small breakout rooms are available for meetings or privacy
- Rolling whiteboards and movable furniture provide a dynamic and adaptable workspace configuration
- Team uses open seating to facilitate paired programming and communication
- Leadership sits with the team
- Team is proud to show the space to others and customers
- Allow team to be creative and introduce fun to the space
- Noise equals collaboration and communication, quiet is “bad”
- Requires facilities and leadership to understand the goals of the team and help them adapt the space to be effective for their needs
- Teams sit facing each other, not with their backs to each other
- “Hey Team” collaboration when roadblocks are discovered
- A high-quality speakerphone and TV is set up and ready to use for video conferencing
- Everyone has a laptop to improve mobility
- Large monitors set up to connect laptops
- Have hand-sanitizer readily available to prevent spreading germs

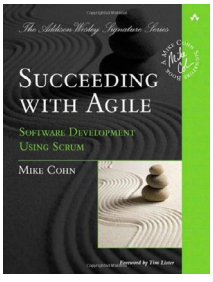
ADVANCED

- The Open Workspace is toured regularly by others in the organization to see and understand how such a space enables a high performing team
- Provide white noise generators
- Food/coffee station nearby but not on center island
- Rolling toolbox of office supplies (sticky notes, markers, painter's tape, scissors)
- Team works in a bright and desirable space and not relegated to closed conference rooms and basements
- One large breakout room for larger meetings
- Team members choose type of laptop (Mac or PC) if the team agrees it's okay to have both kinds on the team
- Windows are viewed as tools and the team is allowed to have things taped to them and use dry-erase markers on them
- Team has access to a plotter to print information radiators or TVs to radiate information digitally

OPEN WORKSPACE



RECOMMENDED



"Succeeding with Agile: Software Development Using Scrum"
By Mike Cohn

vimeopro.com/leandog/open-space

T-SHAPED PEOPLE

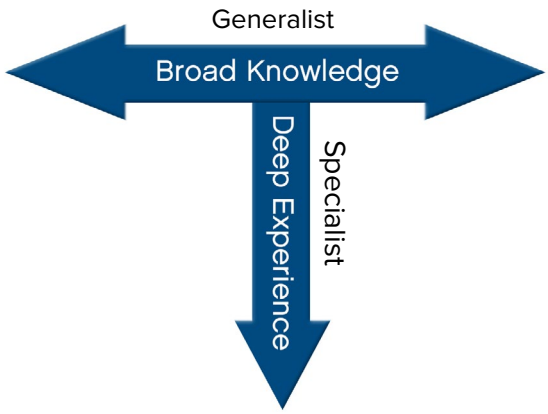
A T-Shaped person is an individual who has deep knowledge of a specialized skill set but also has acquired tangential related skills. T-Shaped people are also interested in continuously broadening their knowledge as well as deepening a core skill set. T-Shaped people are also known as generalizing-specialists or polymaths.

ESSENTIAL

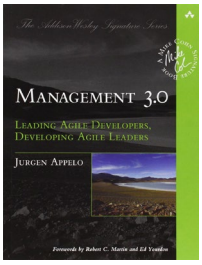
- People are encouraged to learn and pair in all roles
- Collaboration between team members to build T-Shaped People
- Team members cross-train each other in technical and domain specialized knowledge
- Team creates an environment that facilitates and encourages continual learning
- Activities such as pairing, job shadowing, lunch-n-learns, book clubs, and open discussions are used frequently

ADVANCED

- Anyone on the team can pick up any story card
- Team members are encouraged to pick story cards in areas with which they are unfamiliar as a learning challenge
- Team will focus on learning activities around bottlenecks
- Create cross-team communities of practice around technical specialties, domain knowledge areas, or any other area of interest



RECOMMENDED



"Management 3.0: Leading Agile Developers, Developing Agile Leaders"
By Jurgen Appelo
(Chapter 13)

vimeopro.com/leandog/tshaped-people

A Sustainable Pace is a constant pace that a development team should be able to maintain indefinitely, to ensure the team has time to plan, think, rest, and deliver effectively.

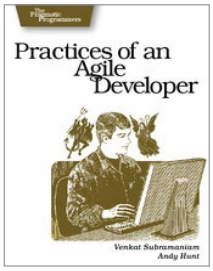
ESSENTIAL

- Team members typically work a 40-hour week
- Includes daily work responsibilities, training opportunities and sustainable fun
- Must give enough time to include all regular team activities such as initiation, planning, development, testing, and deployment
- Avoid the death march or the need for heroic efforts; instead, focus on repetitive delivery of high-quality software

ADVANCED

- Team pace should include enough slack, mental downtime, or development time to allow for other activities which foster creativity and innovation
- Team is evaluated on performance, not presence (Result-Only Focus)
- All regular workweek constraints are removed from the core team (i.e. All team members are required to be at their desks for workday by 8 a.m.)

RECOMMENDED



"Practices of an Agile Developer"
By Venkat Subramaniam and Andy Hunt

vimeopro.com/leandog/sustainable-pace

INFORMATION RADIATORS

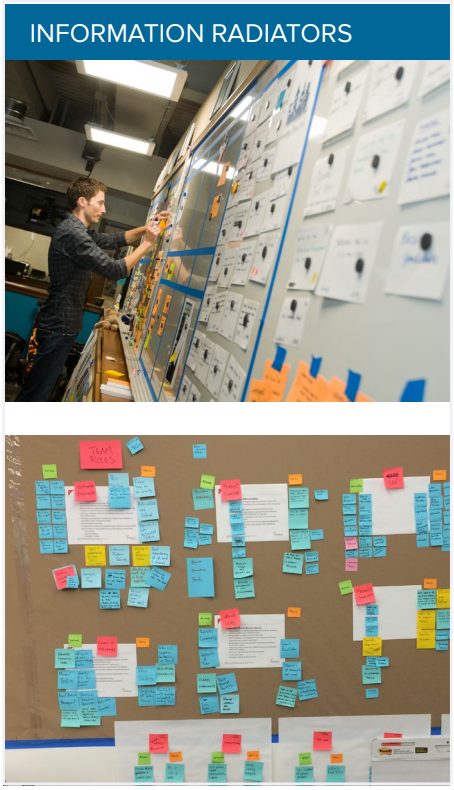
Information Radiators display important project information simply, communicating information even from across the room. Use Information Radiators to map project statuses so the team knows the status at all times: what's coded, what's verified and what's to come.

ESSENTIAL

- Anyone should be able to review the information displayed and easily understand it
- Key charts include:
  - Release Plan & Roadblocks
  - Product Value Map
  - Burn Up & Velocity
  - Story Card Wall
  - Unit Test & Acceptance Test Coverage
- Used for continuous communication and information radiation
- Presented in a casual way (no complexity, just charts on walls)
- Team Blocks & Escalation

ADVANCED

- Team decides what is visible
- Flow of information is centered around the workspace. The walls should tell a story to the team and customer
- Charts track everything from future project requests to fully-tested stories
- Other helpful items to be considered:
  - Organizational Value map
  - Program Demand Management
  - Persona Map
  - Ideation Wall (to capture and compare new ideas or potential work)



vimeopro.com/leandog/radiators

Clear communication is foundational for successful teams. It requires validation of shared understanding and is critical for aligning a team. When everyone is and stays on the same page, teams can continually improve the way they work and collaborate.

ESSENTIAL

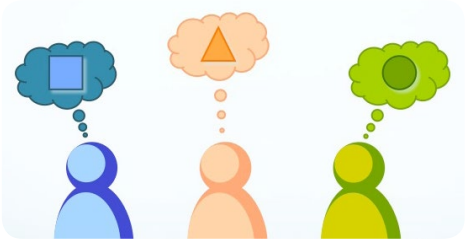
- Diagrams, flowcharts, and other visual aids are excellent ways to ensure that ideas discussed are understood in an intended manner and they spark conversation
- Use Information Radiators in the team space to communicate real-time system status, performance, and usage metrics

ADVANCED

- Use workshop-style collaboration sessions to bring the team together to ideate, solve issues, and dive deeper into the problem space
- Set an objective for each release so the team understands why stories are prioritized and what the impact is of each story. You can have 1-2 overarching objectives
- The whole team is involved in story writing, prioritization, and planning

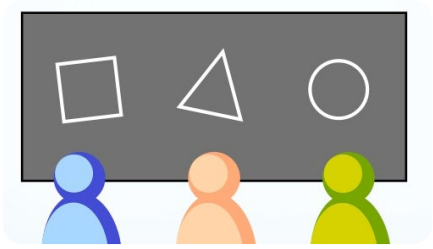
CLEAR COMMUNICATION

1. Clear communication is the foundation



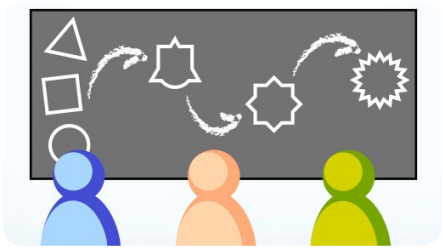
"I'm glad we all agree"

2. Get those mental models out on the table



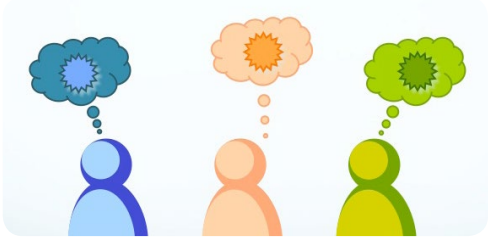
"Ah!"

3. An explicit model allows convergence through iteration



"Ah!"

4. A genuinely shared understanding



"I'm glad we're all agreed then"

FREQUENT RELEASES

Frequent Releases are intended to shorten feedback cycles and improve responsiveness by deploying code in short cycles. This improves productivity by providing more opportunities to handle new or changing requirements or adjusting priorities of planned work in response to business or user needs.

ESSENTIAL

- At the end of each iteration the software should be ready for production, the business determines if it is released
- Apply fixed capacity against a fixed date, prioritizing cards by business value The capacity and schedule are fixed, but the scope is flexible
- Fixed capacity and fixed scope allow for flexible dates
- The team (including the product owner) must agree on what a completed state means for individual features and releases

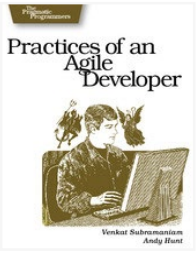
ADVANCED

- Working software is frequently delivered to business partners in small, functional chunks
- Releases are scheduled based on the input and needs of business partners
- Teams strive to reduce the risk of large, “big bang” releases by reducing the size of each release, and automating the release process to improve repeatability and minimize the cost of releasing frequently
- Create a release schedule to help shape the direction of the product or initiative

RECOMMENDED

vimeopro.com/leandog/frequent-releases

"The Art of Agile Development"



"Practices of an Agile Developer" *By Venkat Subramaniam and Andy Hunt (Chapter 4)*

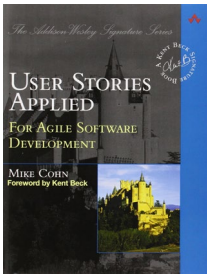
The Story Card Wall is an information radiator tool used in the team's workspace. It is an effective way to display the status of each card in its current iteration. The story card wall is broken up into columns reflecting each function necessary to the process. Story cards can be index cards or Post-its®.

ESSENTIAL

- Must be on a physical wall in the team's space
- Positioned in a clean, well-lit place
- All team members move cards
- Cards should be written, not printed (easy to change vs. edit tool/reprint)
- The definition of 'complete' should be clear
- Personal WIP limits: no more than one card per person can be in play at a time
- Everyone on the team can read and explain the story card wall
- Use color coding to differentiate cards of different work types - legend Minimal Marketable Feature Sets (MMF)

STORY CARD WALL

RECOMMENDED



"User Stories Applied: For Agile Software Development"
By Mike Cohn

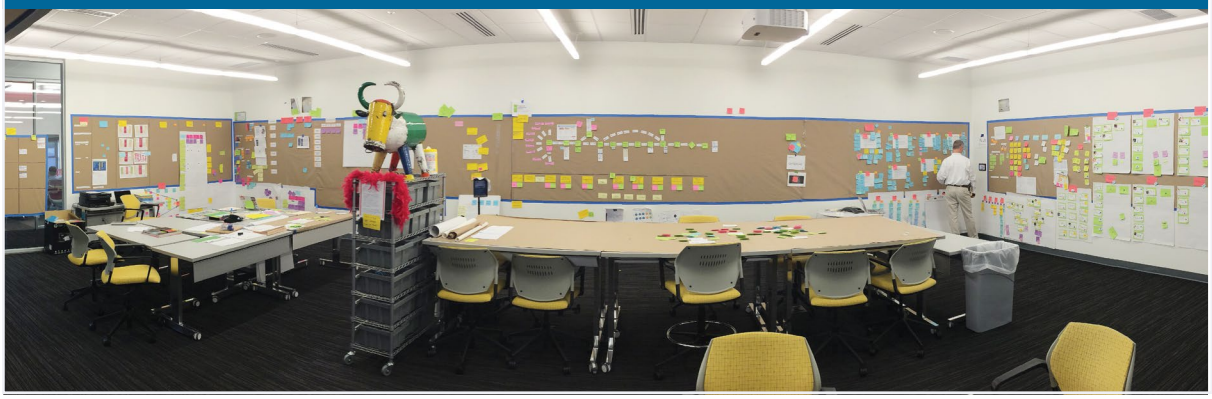
vimeopro.com/leandog/story-card-wall

STORY CARD WALL

ADVANCED

- Portable magnetic dry erase card walls are the best
- Enough room around the wall for the team to gather
- Pictures on magnets show card owner
- Retrospective items on wall are not necessarily part of a Story Card Wall. However, some teams do maintain an area to list topics that team members want to discuss at an upcoming retrospective.
- Stand up should be in front of a wall
- Visual sign explain specific transition criteria for cards to move between statuses on the board (e.g. what has to be done/verified for each card in "Development" before it can move to "Testing")
- Anything in progress has a name attached to it
- Cards are easy to read from a distance
- Team establishes an overall team WIP limit (in addition to personal WIP limits) to reflect team's sustainable capacity
- Team writes "Created", "Started", and "Completed/Accepted" dates on cards to facilitate tracking of lead time/cycle time
- On-Hold Metrics: Some teams keep track of "Hold" time - i.e. when a card cannot be worked on continuously once it has been started. Cards are annotated with hash marks to represent any full- or half-day increments during which the card had to be placed "On Hold" for some reason.
  (Some overlap with Roadblocks, but sometimes reveals/highlights different process problems)

PORTABLE MAGNETIC DRY ERASE CARD WALLS



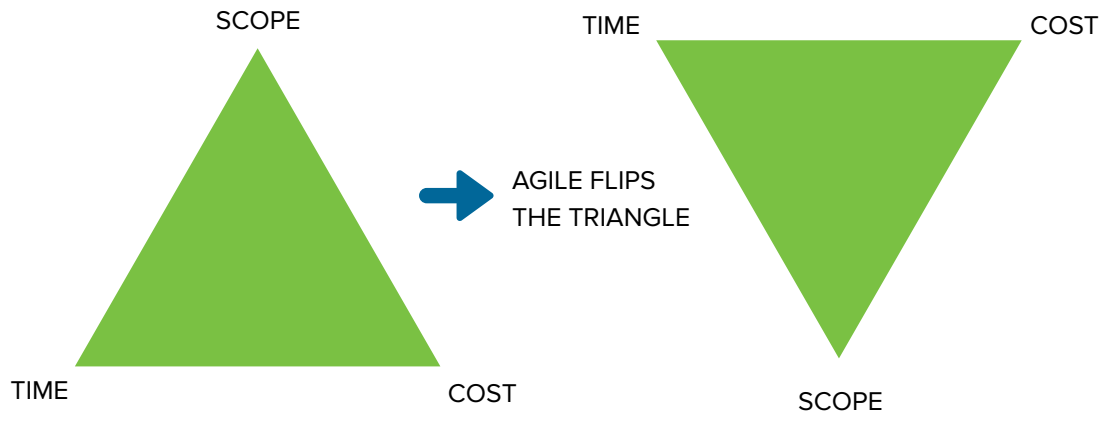
Agile teams are often asked to be "adaptive, flexible, or agile," but also asked to stick with a plan. A traditional plan is based on scope, schedule and cost. An agile triangle turns the triangle upside down and has very different goals of customer delight, building quality products while speeding up the development process. The Agile Triangle in simpler words changes the way we view success [1].

ESSENTIAL

Question should be: "Can we release this product now?"

- Look at the value of functionality over implementing all the requirements
- Quality today - current iteration or release of product
- Quality tomorrow - release that continues to respond to business changes both anticipated and un-anticipated
- Constraints - scope, schedule, cost - not unimportant, but not the goal of the project

AGILE TRIANGLE



1 "Agile Project Management: Creating Innovative Product" by Jim Highsmith

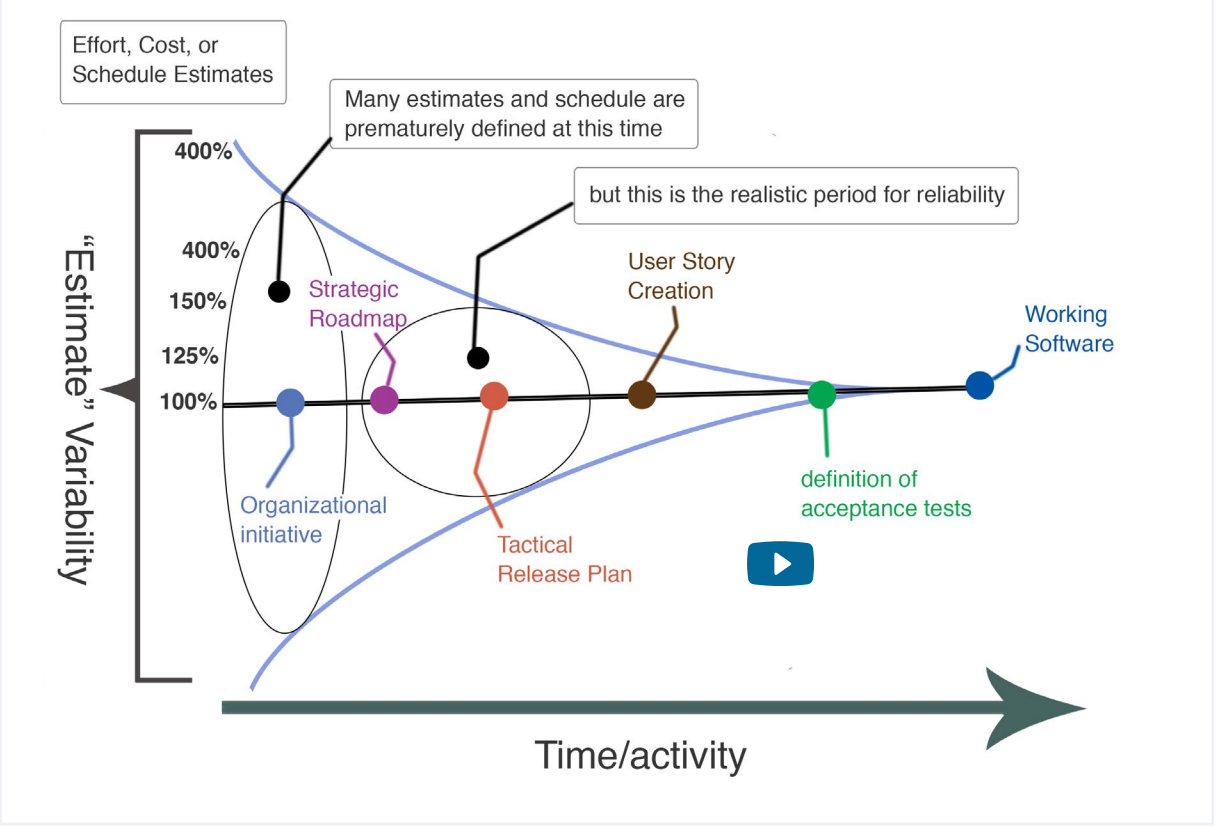
CONE OF UNCERTAINTY

The Cone of Uncertainty describes the amount of uncertainty during different time periods of a project. At the beginning of a project, it's hard to estimate activity with a high level of accuracy and confidence because little is known about the product or the results. As you begin researching and developing the product, the uncertainty level will decrease.

ESSENTIAL

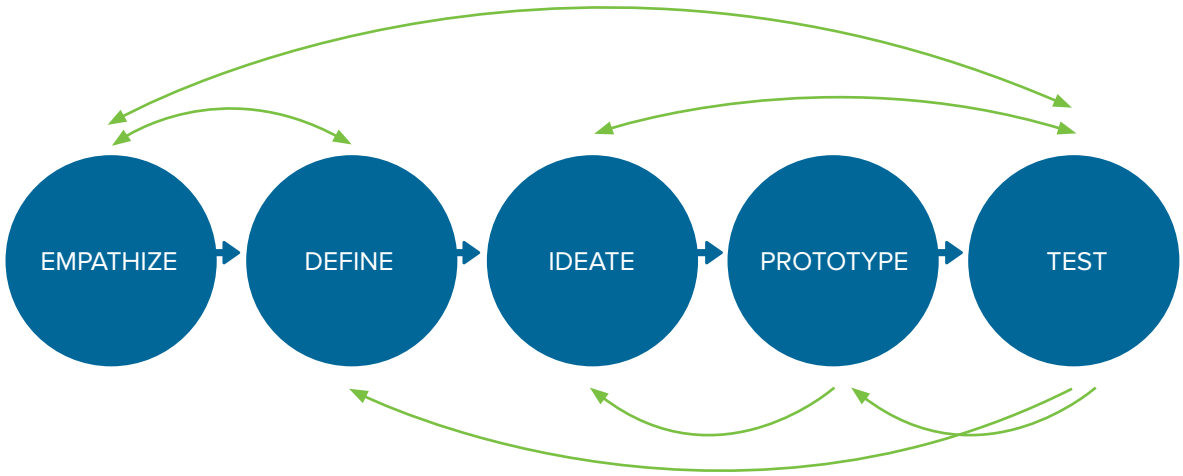
- Estimates at the beginning of the project are very vague
- Estimates and project plans need to be re-estimated on a regular basis
- Uncertainties should be built into the estimates
- Uncertainties should be visible in project plans

CONE OF UNCERTAINTY



A design methodology that helps define complex problems through understanding human needs. This human-centric approach is driven by empathy and root cause analysis, a process of understanding the real problem. Although design thinking is a non-linear process, there are typically 5 different stages associated and are as follows: Empathise, Define (the problem), Ideate, Prototype, and Test.

DESIGN THINKING PROCESS



DESIGN THINKING

1. EMPATHIZE

If the overall goal of Design Thinking is to gain an understanding of user problems, then the first phase is the most critical. In order to empathize with user needs, we must first observe and engage to understand their experiences and motivations. Often times immersing yourself in the user environment will yield a deeper personal understanding of the issues involved. Design thinkers must be able to set aside assumptions to gain pure insights into user needs. Information at this stage is compiled for phase 2: Define.

2. DEFINE

At this point, the information and observations need to be analyzed and synthesized to define the core problems. These problems should be defined in a user-centric, problem statement.

Example: "How can we provide aspiring developers with the resources they need to grow and learn?"

3. IDEATE

Now that you have a well-defined problem statement, it's time to "think outside of the box" with team members to propose solutions. Brainstorming sessions need to facilitate open-minded thinking but also focus to walk away with a plan to investigate and test your proposed solutions. Aside from brainstorming you can hold sessions such as, "Worst Possible Idea," which is designed to look at all the things you would never do to solve a problem.

Examples: "Spam Users, Log them out after 5 minutes", etc. If your team struggles at this stage to define the best solutions, don't worry about the final two phases (Prototype, Test) often form a natural feedback loop where you will be doing... Ideate, Prototype, Test a few times or until the best solution is discovered.

4. PROTOTYPE

Designers now produce scaled-down versions of the product features, so they can dive deeper into the problem solutions generated in previous phases. This is an experimental phase, meant to gather feedback within the team and sometimes even with external departments. The prototypes generated are investigated, and either accepted, improved and re-examined or rejected on the basis of users' experiences. By the end of this stage, the design team will have a must understanding of how users will interact with the end product.

5. TEST

Finally, it's time to test users to understand if you've delivered the best problem solution. Often times this phase will result in re-defining the problem, sometimes we miss the most important problem in phase 1: Empathise. Design Thinking is not a linear process and by prototyping and testing often you will learn faster, what the root cause of the user problem and the best way to deliver a solution!

DESIGN-LED ORGANIZATIONS ARE THE WINNERS

If you're questioning why you'd invest in design or you look at companies that seem to be innovating with every launch and wonder how they do it. The answer has been proven - design-led organizations to increase their revenue and shareholder returns at twice the rate of others in their industry.

A study done by McKinsey and published in 2018, looked at 300 publicly traded companies over a five year period and assessed the impact design has on those organizations. They analyzed two million pieces of financial data and 100,000 design actions. Four themes emerged and form the basis for the McKinsey Design Index (MDI), which rates companies by how strong they are at design and—for the first time—how that links up with the financial performance of each company.

Analytical leadership, user experience, cross-functional talent, and continuous iteration are the dimensions that propel organizations to the top. It takes time and investment but it truly helps organizations innovate. A case study in using design to lead an organization to greater heights is Disney. To delight their customers even more, Disney asked itself how it might remove the friction points in their visitors experiences. A small cross-functional team was assembled and set up right on the main street of the Disneyland Park. After years of research and iteration, the Disney MagicBand was ready for rollout. The MagicBand is a wearable device that connects up with sensors and systems throughout the park and eases the visitors interactions. With simply the wave of a wrist, visitors can access rides, purchase sodas, and check into their hotel room. No part of the visitor experience was left untouched in remaking a magical experience.

RECOMMENDED

"Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation"
By Tim Brown

"Innovation by Design: How Any Organization Can Leverage Design Thinking to Produce Change, Drive New Ideas, and Deliver Meaningful Solutions"
By Tom Lockwood

- **Design thinking origin story plus some of the people who made it all happen - Jo Szczepanska**
  https://medium.com/@szczpanks/design-thinking-where-it-came-from-and-the-type-of-people-who-made-it-all-happen-dc3a05411e53
- **Mckinsey Design Report**
  https://www.mckinsey.com/business-functions/mckinsey-design/our-insights/the-business-value-of-design
- **Disney's $1 Billion Bet On A Magical Wristband**
  https://www.wired.com/2015/03/disney-magicband/

Visualize workflow

CHAPTER 3

Process

IN THIS CHAPTER

- Agile Process
- Kanban
- Scrum Framework
- Story Card Writing
- Estimation & Sizing
- Sprints/Iterations
- Customer Collaboration

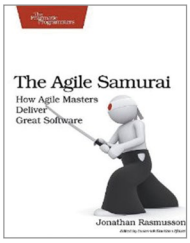
PROCESS

There are several popular models: eXtreme Programming (XP), Scrum, Crystal, Kanban, etc. All have a common solution for improving the outcomes of software development through concepts like “the minimum responsible amount” of many artifacts. Consider specific circumstances to determine which might be the best fit for each situation. The LeanDog Agile Discussion Guide gives you an idea of the many tools and approaches that are available for addressing specific challenges that your team may encounter.

ESSENTIAL

- Recognize that current processes, organizational structures, culture and approaches are usually the key causes of throughput/quality problems - not your team members
- Coach your team members on finding problems and challenges: finding, identifying and talking about problems is EXPECTED
- Based on the problems and challenges that are being highlighted, we can pick processes and approaches to help resolve them

RECOMMENDED



"The Agile Samurai: How Agile Masters Deliver Great Software"
By Jonathan Rasmusson

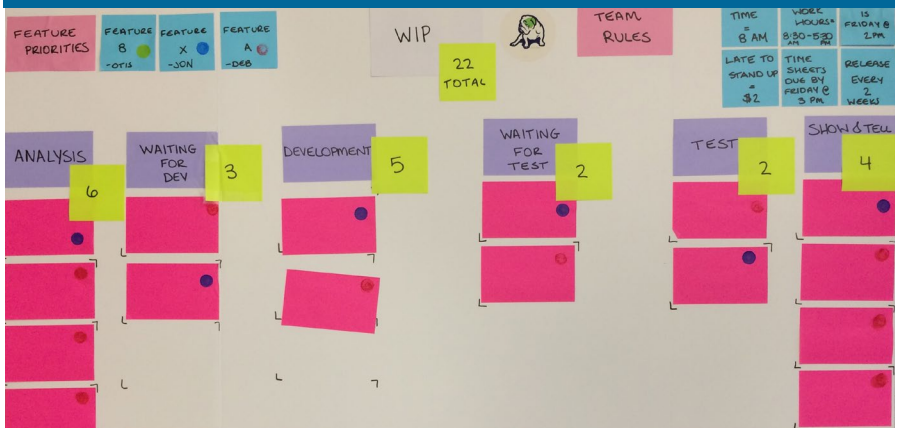
vimeopro.com/leandog/agile-process

In the 1950's, Taiichi Ohno began using Kanban in Toyota's primary machine shop. Translating to "signboard" or "billboard", Kanban is a visual scheduling system that helps a team determine what to produce, when to produce it, and how much to produce.

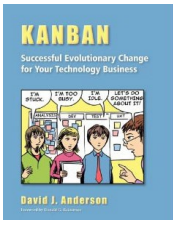
ESSENTIAL

- Visualize workflow and picture the product in each state—from concept to deployment
- Limit work in progress (WIP) to prevent your team from becoming overwhelmed, and keep progress continuous and steady (flow)
- Manage flow in order to allow and prepare for changes that will occur within the iteration
- Make team, queue and practice policies explicit so your team knows how to participate properly, ensuring a smoother iteration
- When a TEMPORARY surplus in capacity occurs, give priority to helping other team members complete any work already in progress, before taking on new tasks/cards

VISUALIZE THE WORKFLOW



RECOMMENDED



"Kanban: Successful Evolutionary Change for your Technology Business"
By David J Anderson

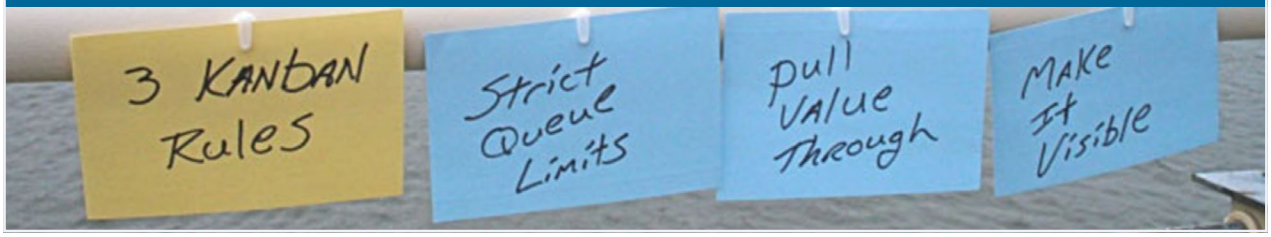
vimeopro.com/leandog/kanban

KANBAN

ESSENTIAL

- Implement feedback mechanisms like operations reviews, mentorship and daily production meetings to ensure demands are being met while identifying where improvements can be made
- Improve collaboratively with a scientific approach: continuous reflective analysis paired with small adjustments, encouraging timely evolution at sustainable pace
- Periodically review/revisit team WIP limits with the team, customers and stakeholders, especially when changes to the team's composition occur
- WIP limits are intended to facilitate the flow of work through the team by making it easy to decide whether or not the team is ready to take on new work. As such, WIP limits should never be used as a negotiation tool - only as a collaboration tool

3 KANBAN RULES



CYCLE TIME

$$\text{Delivery Rate} = \frac{\text{Work in Progress}}{\text{Lead Time}}$$

Decrease WIP in order to decrease cycle time (delivery rate). Smaller stories in process will come out faster.

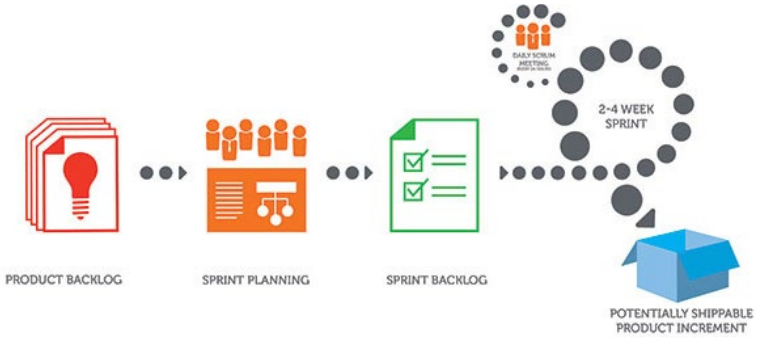
Scrum is an Agile development framework that consists of individual teams working interdependently. The teams work together, but focus on their own tasks and must be capable of self management and decision making. Scrum is based around a "sprint," (or iteration) which is generally a 1-4 week period for delivering a working part of the system. At the end of a sprint the results are delivered and reviewed, then the next sprint is started.

ESSENTIAL

- A product backlog is created and prioritized by the Product Owner
- The Product Owner selects story cards from the product backlog and adds them to the sprint backlog, deciding which cards will get implemented
- A sprint (or iteration) is a short amount of time to get work done and receive feedback. The team selects the duration, LeanDog recommends 1-2 weeks.
- The team meets each day to assess their progress; a "daily scrum"
- The Scrum Master keeps the team focused on their goals
- At the end of each sprint, the completed work that was selected from the product backlog is shown to a stakeholders, released to the customer, or shelved for a near term future release
- Every sprint ends with a sprint review, show and tell and team retrospective
- When a new sprint begins, the team selects another group of prioritized story cards from the product backlog
- The cycle repeats until there are no more story cards to be completed in the product backlog, the budget has been spent, or the deadline has passed. Using the scrum process ensures that the most valuable parts of the product are built first

GET THE FULL GUIDE FROM SCRUMALLIANCE.ORG

ScrumAlliance®
transforming the world of work



vimeopro.com/leandog/scrum-framework

1 Scrum Alliance, http://www.scrumalliance.org/pages/what_is_scrum
Picture source: Scrum Alliance, http://www.scrumalliance.org/pages/what_is_scrum

SCRUM FRAMEWORK

SCRUM FRAMEWORK

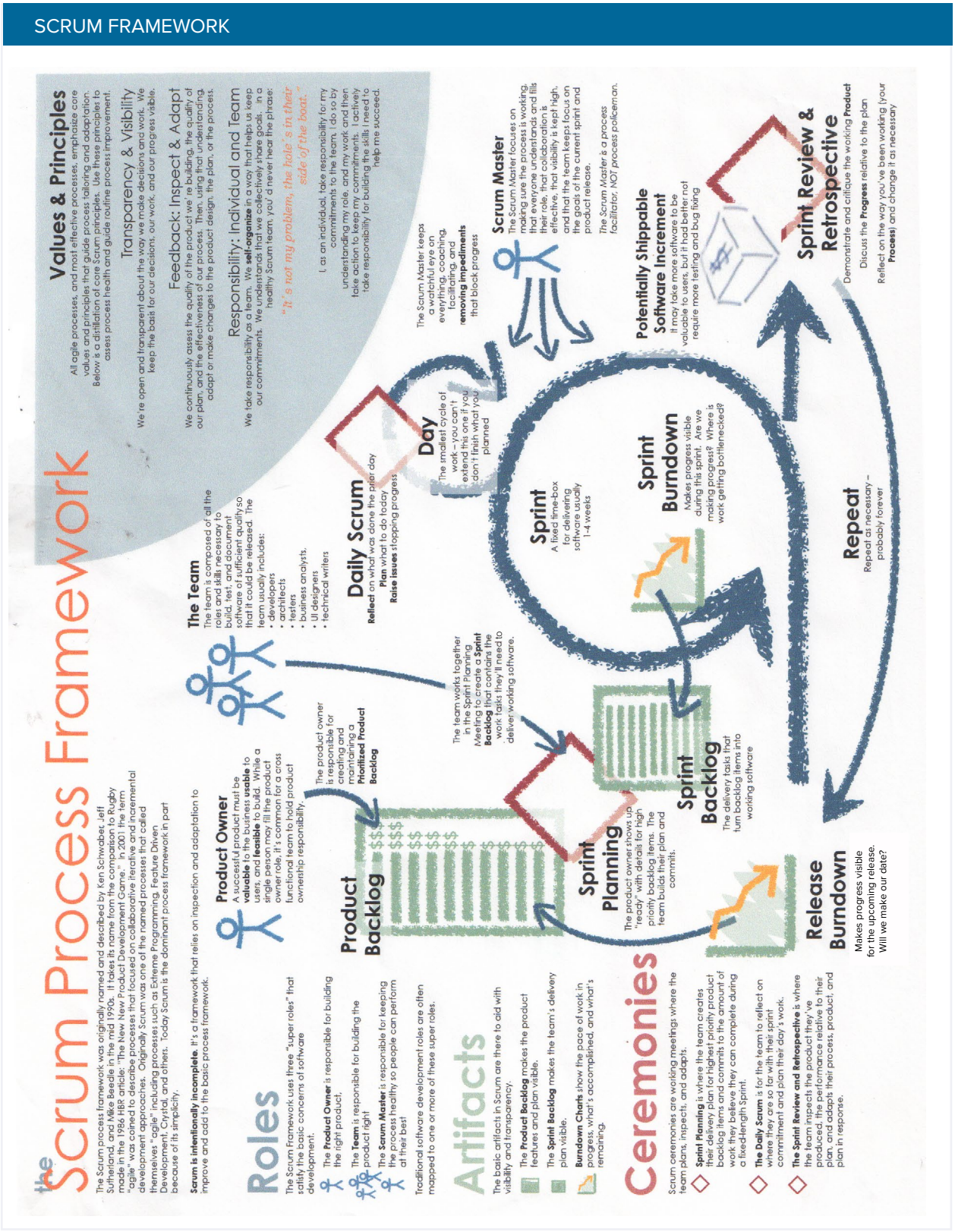


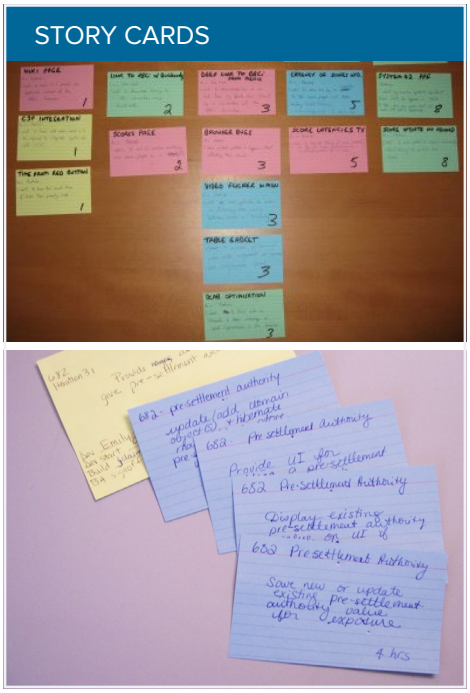
ILLUSTRATION BY JEFF PATTON

STORY CARD WRITING

A Story Card represents a user story and is the unit of each deliverable for an Agile team. Story Cards include a sentence or two describing a needed function. Rather than representing detailed requirements, story cards are a “placeholder for a conversation”. Story cards are testable and include acceptance criteria. The details are elaborated upon via a conversation between the customer, and the delivery team.

ESSENTIAL

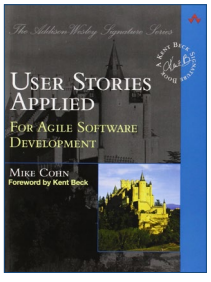
- Cards are handwritten
- Cards should follow the format: "In order to (X), As a (Y), I want (Z);" these are the Values, Roles, and Goals of the story cards
- Spike cards exist
- Cards follow the INVEST rule
- Story cards should represent a vertical slice of the application and deliver business value
- Story cards are sized to fit an iteration
- Use the 3 C's: the Card, the Conversation AND the Confirmation
- The definition of "done" must be clear



ADVANCED

- Developer pairs with BA/Product Owner to write cards
- Utilize Story Mapping as a tool to identify stories and priorities

RECOMMENDED



"User Stories Applied: For Agile Software Development"
By Mike Cohn

STORY CARD WRITING

INVEST RULE

- INDEPENDENT
  Story Cards should be independent of each other.
- NEGOTIABLE
  The cards should include a short description of the deliverable without too many details. The details should be worked out during the "conversation" phase.
- VALUABLE
  Each story must be of value to the customer and have a value given to it.
- ESTIMABLE
  In order to plan and prioritize, developers need to be able to estimate the effort to complete the story.

- SMALL
  Ideally the story should be small, typically taking no more than 2-3 days.
- TESTABLE
  A story needs to be testable for confirmation to take place.
  If you cannot test it then you will never know when you are done.

VALUES, ROLES, & GOALS

Business Value: In order to _____

User Role/Persona: As a _____

Goals/Perform Something: I want to _____

Or, Alternate Format:

As a __(who)__

I Want __(what)__

So that __(why)__

THE 3 C'S

User stories have three critical aspects:

- CARD
  Stories are written on cards. Each card does not contain all of the information, but just enough to show the requirements of the work to be completed. Cards are used during planning.
- CONVERSATION
  The requirements on each card are communicated from the customer to the delivery team members through conversation: exchange of thoughts, opinions, and feelings. These conversations happen many times before/during release planning, during iteration planning, and again when the story is ready for implementation.
- CONFIRMATION
  After running acceptance tests and confirming with the customer that the tests were successful, a card has been completed.

Estimating and Sizing are techniques for evaluating the size / complexity of delivering features in order to facilitate planning and guide investment decisions. A good planning process based on a reliable estimation and sizing approach reduces risk and uncertainty, supports better decision making, establishes trust and conveys information.

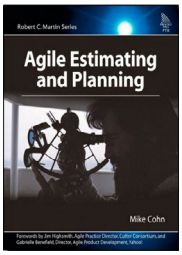
ESSENTIAL

- Use relative sizes (Planning Poker - Story Points, T-shirt sizes, etc)
- Team understands it's not just time, it's complexity and uncertainty
- Very few large cards
- Cards which the team feels are too large/complex to complete within an iteration must be broken down into smaller / less complex cards
- Re-estimate when significant new information is uncovered, as it changes complexity
- Management understands differences of size/estimating vs. hours
- Re-estimate in the event of team changes
- Make all cards as small as possible
- Consider all cards small "By Definition"
- Rewrite / breakdown / decompose a card when a smaller slice is found

ADVANCED

- Measure and account for "Card Growth" between planning and delivery (e.g. 1 card in the backlog may turn into, on average, 1.25 cards delivered, due to some fraction of the cards being able to be decomposed into smaller slices when they are discussed in detail with the Product Owner)

RECOMMENDED



"Agile Estimating & Planning"
By Mike Cohn

vimeopro.com/leandog/estimation

SPRINTS / ITERATIONS

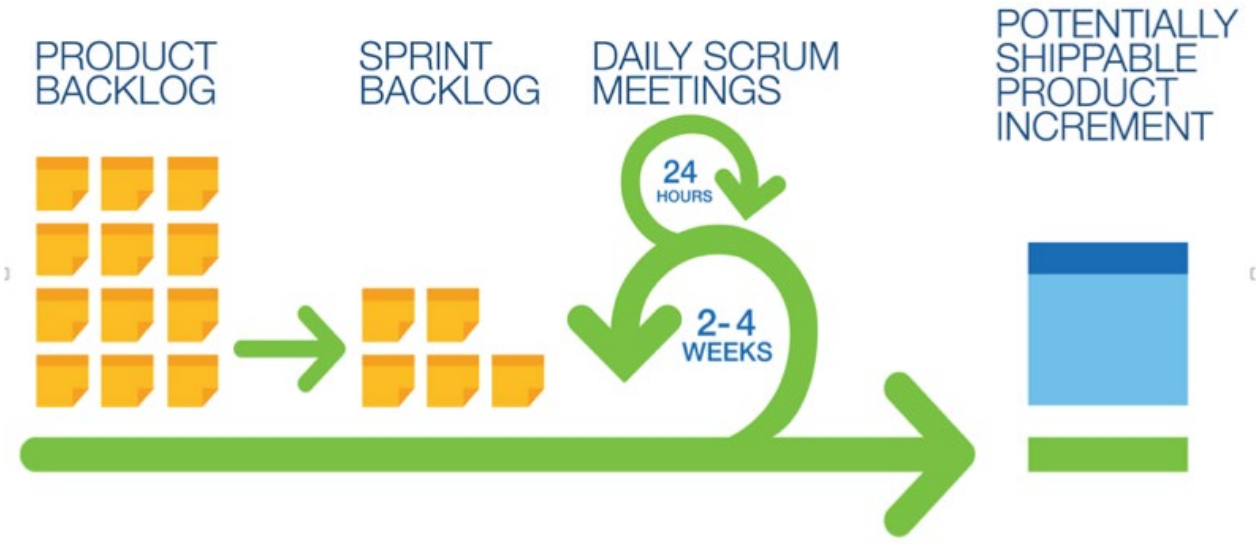
A Sprint / Iteration is a set period of time for measuring a development team's throughput. Sprints/ Iterations are one to four weeks in length with shorter iterations (one to two weeks) being the goal.

ESSENTIAL

- Work is organized to be completed in short time frames (preferably 1 - 2 weeks)
- Story cards worked on during an iteration must meet the team's definition of done and get Product Owner acceptance to be considered complete
- Team gets credit for functional, tested and approved code

ADVANCED

- Unplanned work is noted at retrospectives



vimeopro.com/leandog/sprints-iterations

CUSTOMER COLLABORATION

Customer Collaboration is the practice of including your customer throughout development. Only the customer can tell you what they want, and collaboration allows you to listen to their needs instead of guessing at what they need. The customer drives the requirements, prioritization, and review of work. While it is not always possible to have customers on site, you must make interaction with them a priority.

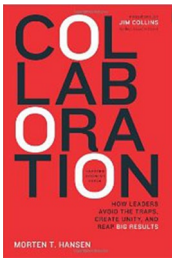
ESSENTIAL

- The customer must be clearly identified, and roles and responsibilities must be clearly defined
- The customer may take the form of customer proxy, Product Owner, business stakeholder, end-user or any combination of these roles. The customer owns the responsibility of driving work in the team that meets overall product goals and objectives for building and delivering software. The customer focuses the team's efforts on the highest priorities, highest business value features, and stories for release
- The customer works closely with the team to break features down into stories
- The customer determines minimal marketable features and signs off on completed stories supporting those features for release
- Grooming of product backlog is a collaborative effort between the customer and the team to adapt to changing product goals and objectives
- The customer listens to the team and the team listens to the customer
- The definition of "done" must be clear

ADVANCED

- Create personas as a tool to better understand the customers and their expectations
- Co-locate with the team to increase collaboration and communication
- Attend all team meetings (stand-ups, sprint planning, show and tell, etc)

RECOMMENDED



"Collaboration: How Leaders Avoid the Traps, Create Unity, and Reap Big Results"
By Morten Hansen

vimeopro.com/leandog/customer-collaboration

An organization's leadership needs to create a culture of continuous improvement

CHAPTER 4

Leadership

IN THIS CHAPTER

- The Role of An Agile Leader
- Act as a Servant Leader

THE ROLE OF AN AGILE LEADER

Leadership in Agile is different from traditional workforce management that was designed for laborers during the industrial revolution. While leaders can exist anywhere and at any level of an organization, this chapter will focus on the role of those that have direct reports in an agile environment.

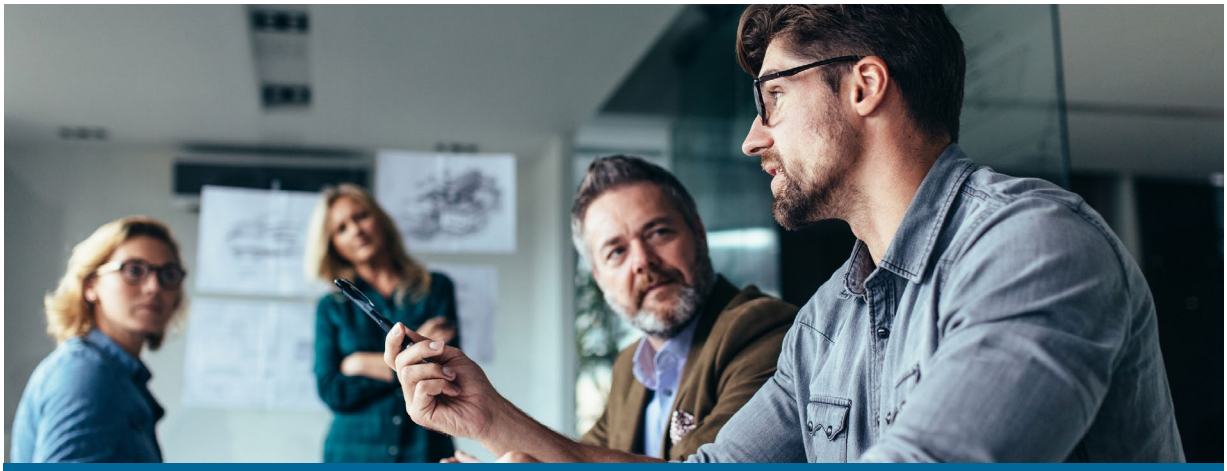
Agile leadership takes a humanistic approach (that motivated people will do the best work possible if given the tools, training, and support) rather than deterministic approach (that people will not work unless management motivates with rewards or punishment/fear). Research shows that a humanistic approach is required to maintain engagement of knowledge workers. This chapter outlines characteristics and actions that support a humanistic approach.

ESSENTIAL

- Leadership is a service, anything else is not leadership
- A leader's most important job is to create an environment where people bring their whole selves to work, have psychological safety and understand that it is OK to fail
- Leaders understand needs and support their people to maximize and grow their own abilities
- Leaders understand strategic outcomes and communicate those effectively
- Leaders operate within their level of hierarchy without intervening in lower tactical levels (E.G. The CTO does not interact with teams apart from socially)
- First level leaders engage with their teams as mentors, as developers of people and as trusted advisors
- Leaders protect their teams from external influences and distractions
- Leaders take ownership of blockers that the team cannot remove themselves and gives them priority until they are resolved
- Leaders hire skilled, motivated individuals that will be the best fit for their teams
- Leaders build trust and transparency so they can give and receive radical candor

PROMOTING PSYCHOLOGICAL SAFETY

Psychological safety is defined as a shared belief that the team is safe for interpersonal risk taking and being able to show and be one's self without fear of negative consequences of self-image, status, or career. This is one of the fundamental requirements for sustaining Agile as it is difficult to learn, grow, or innovate when individuals are not free to speak out. One of the most data-intensive research projects to support this was Google's Project Aristotle which had a goal to determine what made an effective team. This research project determined that the effectiveness of a team was not determined by who was on the team, but was determined by how the team worked together. The most important characteristic was if the team had Psychological safety.



THE EFFECTIVENESS OF A TEAM IS DETERMINED BY HOW THE TEAM WORKS TOGETHER

ACT AS A SERVANT LEADER

Per L. David Marquet, Servant leadership seeks to give control and create leaders, rather than taking control and creating followers.

THE FOLLOWING 10 CHARACTERISTICS ARE ADAPTED FROM THE WRITINGS OF ROBERT K. GREENLEAF:

- **Empathy** - A servant leader has the ability to recognize and understand feelings and emotions that are experienced by others. This understanding will drive actions that are motivated by a genuine desire to help others
- **Listening** - By actively and intently listening to what others are saying, what they are not saying, what one's own inner voice is saying, combined with periods of reflection, a servant leader will provide a more informed understanding of the situation
- **Awareness** - General awareness and more importantly self-awareness, helps inform decision making as well as growth. Servant leaders do not seek solace, but awakening
- **Healing** - People in need of healing can unknowingly express this need in unexpected, confusing, and hurtful ways. Servant leaders can use their empathy and listening to heal and fulfill some of that need while building trust and community
- **Conceptualization** - Servant leaders nurture their ability to conceptualize, or imagine the possibilities of future and reconcile it with current realities. They view problems from a Systems Thinking perspective with understanding beyond day-to-day realities
- **Persuasive** - Servant leaders seek to affect change through persuasion rather than to coerce compliance through positional authority
- **Stewardship** - A servant leader acts to hold their position in the organization in trust of the company, not as a reflection of their ego or as a positional castle to launch assertions of their power
- **Foresight** - Servant leaders have an intuitive ability to predict what is likely to happen in the future, based on their past experiences, knowledge, and the circumstances of the present
- **Community building** - Under a servant leader, people come together for a common purpose. They are able to create a feeling of belonging to something bigger than each individual, and foster team spirit and a sense of community. Servant leaders also care deeply for this community that they create
- **Commitment to the growth of others** - A servant leader is committed to helping grow individuals in their organization by doing everything in their power to nurture that growth. This can include acting as a trusted advisor, acquiring funds for their personal and professional growth, taking a personal interest in other's goals, involving them in decision making, or assisting those that have been laid off to find new employment

PROMOTES SAFE FAILURE

The humanistic approach takes the stance that people will do the best work that they can if they are given the proper tools, training, and support. Part of that support includes providing an environment in which they can fail safely. Failure is always an opportunity to learn and grow and should be celebrated as such. People do not fail, systems do. Agile leaders help to reduce risk so that teams can fail safely. This can include securing funds to set up an experimental environment to reduce risk in deployment, enabling the team to learn new methods to reduce risk like feature flags or limiting exposure to new features to subsets of users. Safe failure increases psychological safety and increases the likelihood of innovation - very few dare to speak up and try some new ideas if they have the potential to cause harm to client experience or negatively impact their career.

PROVIDE AN ENVIRONMENT WHERE PEOPLE CAN "FAIL SAFELY"

PROTECTS THE TEAM FROM EXTERNAL INFLUENCES

Agile teams should be given the opportunity to focus on completing the work from their backlog. Context switching and rapidly shifting priorities produce a condition called thrashing which reduces performance, effectiveness, and quality. An agile leader works to protect the team from these conditions by limiting who can disrupt the team's focus while developing. There should be channels for routing new work requests to a queue that the team can assess and refine into work items at predetermined intervals. Sometimes this is done by routing all work to a product owner to assess and add to the product backlog for the team to refine and decompose into work items during a backlog refinement session or user story mapping activity.

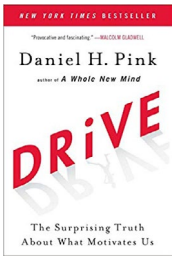
HELPS TO RESOLVE BLOCKERS THAT THE TEAM CANNOT REMOVE THEMSELVES

Agile leaders know how to balance guiding the team to resolve issues on their own, versus resolving issues for the team. Leaders may use their position to secure budget, educate or influence others who may be preventing the team from achieving their goals, or eliminate bureaucracy to facilitate movement.

HIRES MOTIVATED INDIVIDUALS

People are typically aligned to be either more intrinsically motivated, or extrinsically motivated. When we refer to motivated individuals, we are referring to those who are intrinsically motivated. They are more likely to look to themselves for change when something goes awry. These individuals will continue to grow and learn as they face challenges and undesired outcomes. For intrinsically motivated people, motivation comes from within, so doing a great job is its own reward. Contrast this to extrinsically motivated individuals that look to others for someone to blame when something goes wrong. These people do not see the returns from continually improving themselves and do not have as many triggers to increase their learning. Extrinsically motivated people are typically seeking raises, praise, or public recognition for their achievements. Research has shown that these rewards are but brief motivators and if not constantly fed a steady diet of motivation, these individuals can fall into a state of negativity and low engagement.

RECOMMENDED



"Drive" *By Daniel H. Pink*

PERSONALLY ENGAGE WITH THE TEAMS

Data and email rarely tell the whole story. An agile leader seeks to engage with the team in the team space. Taiichi Ohno, an executive at Toyota, developed a systematic approach for this called a Gemba Walk. This is translated as “go to where the value is created”. It was an activity in which the leaders would go to the workers space to observe and gain feedback on how the system could be improved. Observations should be shared with the team, and the team should have the opportunity to solve for the observation. HP also had a leadership approach for this called ‘Management by Walking Around’.

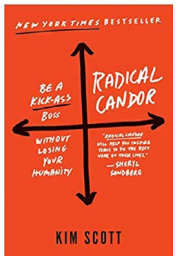
PROMOTES TRANSPARENCY

Provide the ‘why’ behind strategic decisions, organizational objectives, budgeting, and anything that impacts the team, their work, or their place in the organization. Withholding information can be disrespectful and condescending. Treating team members as partners rather than children will have a positive impact on morale and increase feelings of trust and trustworthiness.

GIVES RADICAL CANDOR

The radical candor approach is to care personally, build trust and challenge directly. Starting with your empathy and caring, being specific in how you provide positive feedback, and directly providing observations on opportunities to improve without judgement is a great approach. This has less potential for people to feel personally attacked, like they are victims of aggression, or that observations were sugar-coated and feedback was insincere. As long as trust is first built with the team, Kim Scott’s book “Radical Candor” is a great guide for agile leaders to use when providing feedback to their reports.

RECOMMENDED



"Radical Candor" *By Kim Scott*

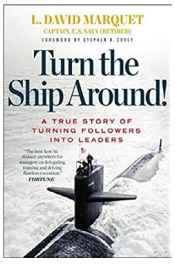
BUILDS AGILE TEAMS

Hire the right people for the culture, skillset need, roles, and size of team. Typical agile teams are 5-7 people, and include a Product Owner and a Scrum Master or a team coach. Team members should be 100% allocated to that team and only that team. There are no partially allocated people and they cannot claim commitment to more than one team. Foster teamwork through co-location and open workspaces. Distributed teams can function, but it is shown to be much less productive than co-located teams. Help the team to celebrate successes. Per Daniel H. Pink, in order to motivate heuristic (thought) workers, leaders must provide autonomy of task, team, time and technique, they must promote and foster mastery in what the team members do and they must maximize the team's purpose. Product Owners provide clarity of what the team is to build, but the leader supports the team in determining their approach, design and how to build the product. Per L. David Marqet, leaders must provide technical competence and organizational clarity in order to give control so that the team will be self-managing. For more on leadership, read 'Turn the Ship Around!' by L. David Marquet.

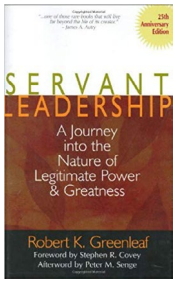
CONTINUOUS IMPROVEMENT THROUGH SYSTEMS THINKING

Per Dr. Russell L. Ackoff, in order to have successful quality improvement, leaders must view their organizations holistically with systems thinking. They must realize and operate under the following principles: (See next page)

RECOMMENDED



"Turn the Ship Around"
By L. David Marquet



"Servant Leadership: A Journey Into the Nature of Legitimate Power and Greatness:
By Robert K. Greenleaf

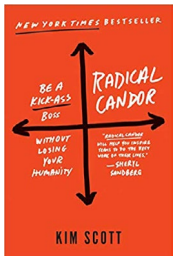
- A system is Not the sum of the behavior of its parts, it is the product of their interactions
- Improving the separate parts does Not improve the whole system
- Only improve the quality of the separate parts if it simultaneously improves the quality of the whole system
- When you fix defects you don't want, you don't necessarily get what you do want
- Finding defects and getting rid of them is not a way of improving performance of a system
- An improvement program must be directed at what you want, Not at what you don't want
- Ask yourself what you would do right now, if you could do whatever you wanted to
- Continuous Improvement is Not nearly as important as Dis-continuous improvement
- Creativity is a discontinuity
- One never becomes a leader by continuously improving - you only become a leader by leap frogging those ahead of you - and that comes about through creativity
- Peter Drucker made the distinction of doing it right or doing the right thing
- Doing the wrong thing right is not nearly as good as doing the right thing wrong
- Quality needs to contain the notion of value, not merely efficiency - that's the difference between efficiency and effectiveness
- Quality needs to be directed at effectiveness
- Until managers take into account the systemic nature of their organizations, most of their efforts to improve performance are doomed to failure

For more on systems thinking, read 'Thinking in Systems: A Primer' by Donella H. Meadows

While there are many things that are harmful for Agile Leaders to do, here is a short list of things specifically Not to do:

- Take Control
- Give Commands
- Set Delivery Dates for the team
- Determine What the team will deliver
- Assign tasks to team members
- Partially Allocate team members
- Determine How the team will deliver the product or solution
- Sit at your desk and wait for emails

RECOMMENDED



"Thinking In Systems: A Primer"
By Donella H. Meadows

The voice

CHAPTER 5

Product Owner

IN THIS CHAPTER

- Product Owner Role
- Value Stream Mapping
- Demand Management
- Product Backlog
- Release Planning

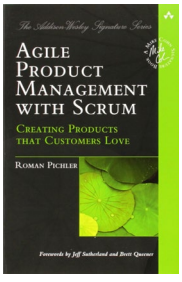
PRODUCT OWNER ROLE

The Product Owner represents the stakeholders and acts as the voice of the customer. They are the keeper of the vision for a particular product or service. They know why we are building the product, what problem it is going to solve, and for who. They are accountable for ensuring that the team delivers value to the business. The Product Owner writes customer-centric items (typically user stories), prioritizes them, and adds them to the product backlog. The Product Owner is not the manager of the team members, but rather, plays a critical role in ensuring that the "voice of the customer" is at the center of everything the team does. Thus, the PO needs to have a deep understanding of the business, its customers, and how the product brings value to the customer.

ESSENTIAL

- Interviews customers, solicits input from stakeholders and represents the Voice of the Customer
- Understands the business, it's customers, and how it's product brings value
- Performs customer journey mapping
- Maintains and communicates the vision of the product or service
- Collaborates with UX to create designs to incorporate customer input
- Creates the product roadmap
- Communicates updates on product features, roadmap and other changes with Program Lead and Stakeholders
- Defines the features of the product or desired outcomes of the project to maximize market value
- Decides what will be built and in which order (building the right thing at the right time)
- Owns the product backlog, writes user stories and orders them to best achieve goals and missions
- Identifies/Collects acceptance criteria for user stories
- Facilitates the release planning ceremony with the iteration manager
- Adjusts features, outcomes and priority as needed
- Accepts or rejects work produced by the team
- Member of the team, not a manager of the team members
- Protects the team from outside influences or interference
- Can say NO to unplanned work from upper management or other stakeholders
- Respects Sprint/Iteration boundaries
- Does not introduce new work, unless the team agrees that they can meet existing commitments plus the newly introduced work
- Ensures the completed user stories meet the acceptance criteria and accepts the work completed by the team
- Facilitates Backlog Refinement, story mapping and new capability deep dives
- Works in close partnership with the team's Scrum Master

RECOMMENDED



"Agile Product Management with Scrum: Creating Products that Customers Love"
By Roman Pichler

vimeopro.com/leandog/product-owner-role

Agile Project Management with Scrum, Ken Schwaber

Value stream mapping is a type of flowchart used to design and analyze the flow of information needed to bring a product to the customer. Mapping allows you to discover waste, then construct a plan to eliminate it.

ESSENTIAL

- Flow of activities begins with the customer's need and ends when the need is satisfied
- Identify the target product, product family or service
- Draw a current state value stream map illustrating current steps, delays and information flows required to deliver the target product
- Assess the current state value stream map in terms of creating flow and eliminating waste
- Draw a future state value stream map

FUTURE STATE VALUE STREAM MAP

vimeopro.com/leandog/value-stream-mapping

DEMAND MANAGEMENT

Demand Management helps organizations deliver the most value from their capacity by adopting continuous portfolio alignment, incorporating pull-based thinking, and limiting work in progress (WIP).

ESSENTIAL

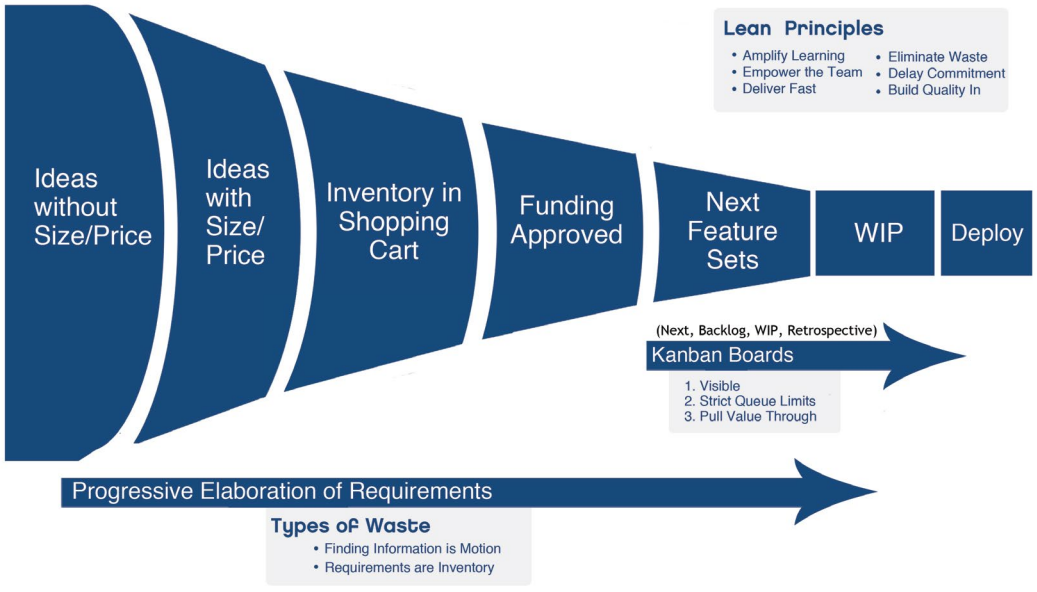
- Regular User Summits bring all users together in order by value to discuss their goals
- Project demand funnel is visible to the team
- Cards are ordered by highest value
- Everyone understands the concept of Minimal Marketable Feature (MMF) or Minimal Usable Feature (MUF)

ADVANCED

- Customer personas are created, present and considered
- Story Mapping technique is used and understood by everyone

SEE THE WHOLE - LEAN SOFTWARE DEVELOPMENT

Standard Workflow: From Idea to Production

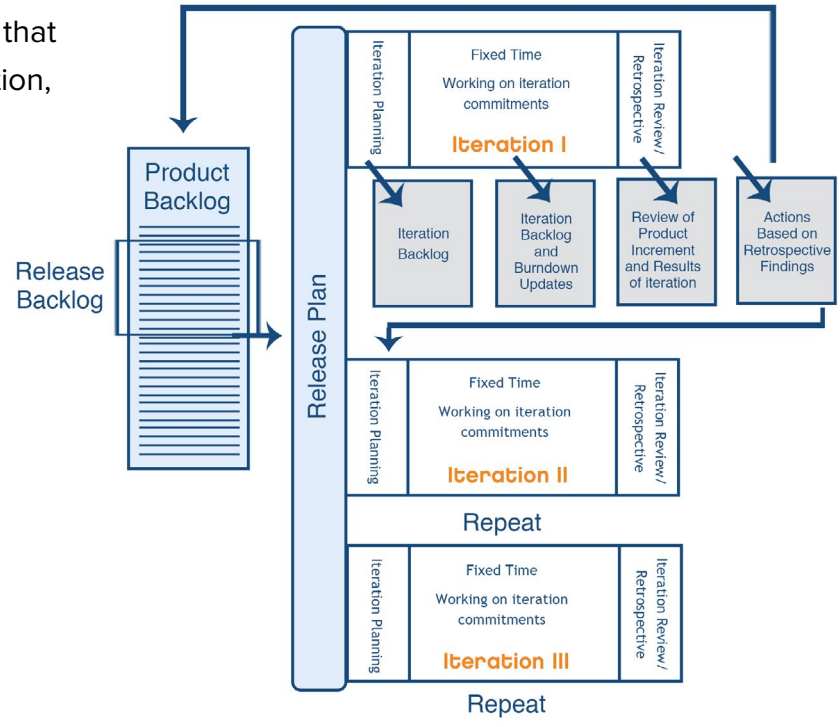


vimeopro.com/leandog/demand-management

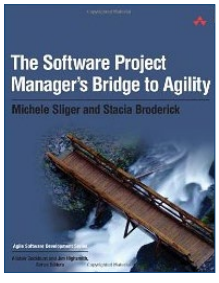
The Product Backlog is a high-level list of all potential features prioritized by business value for the customer.

ESSENTIAL

- Just enough product backlog to create the product without unnecessary features
- Must be visible
- Uses concept of Minimal Marketable Feature Sets (MMFS)
- Must prioritize and organize the backlog by business value
- Backlog is regularly reviewed and maintained
- Not a requirements list
- Team and Customer collaborate on prioritization, to ensure that technical complexity, duration, and cost of features are understood to the extent that they impact business value
- Customer has final decision authority around prioritization of features



RECOMMENDED



"The Software Project Manager's Bridge to Agility"

vimeopro.com/leandog/product-backlog

Michele Sliger and Stacia Broderick, "The Software Project Manager's Bridge to Agility"

RELEASE PLANNING

A Release Plan is an evolving roadmap that sets delivery goals for high-level feature sets. It will provide the team with an overview of the release and what is required to make the release a success. The plan should include the key features, goals, responsibilities, and risks.

ESSENTIAL

- Core team, business owner, and supporting roles drive out product features, details, and stories to be included in the release; anything not deemed part of the release will either be out of scope, or moved to a future release
- Planning the next release is a joint effort between business partners and the team
- Release plans map out several iterations to package releases and maximize business value
- Spikes are used to identify and mitigate risk; risk is pulled forward
- Produce burn up/burn down charts
- Lay out cards in order of feature importance
- Team agrees to a planning velocity which is used to create the Release Plan

ADVANCED

- Release Planning is a continuous activity for regularly scheduled releases
- Rolling release planning window
- Release Planning Meeting with Product Owners to prioritize the next set of features to work on for upcoming releases

THE TEAM SHOULD REVIEW PLAN AND ASK THE FOLLOWING QUESTIONS:

- Is there enough work for all pairs?
- Are any pairs stepping on others?
- Are highest risk cards being played first?
- Are the most valuable features coming out first?
- Are there any dependencies that are missing?

RECOMMENDED

"Product Release Planning: Methods, Tools and Applications"
By Guenther Ruhe

vimeopro.com/leandog/release-planning


SCOTT

The servant leaders of the team

CHAPTER 6

Scrum Master/ Iteration Manager

IN THIS CHAPTER

- Scrum Master / Iteration Manager
- Daily Scrum/Stand Up
- Sprint / Iteration Planning
- Sprint/ Iteration Review
- Retrospectives
- Roadblocks

SCRUM MASTER / ITERATION MANAGER

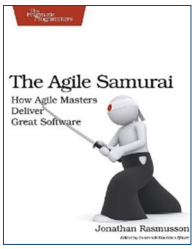
This person is the servant leader of the team sometimes known as Team Coach, Scrum Master or Iteration Manager. A key part of their role is helping the team adhere to the Agile values and principles. Once the team has agreed to certain approaches to their work, it's up to the Iteration Manager to remind the team of those commitments.

ESSENTIAL

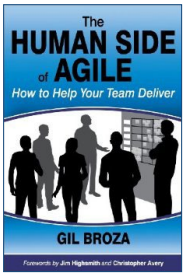
- Remove barriers between the development team and the product owner, ensuring development is directly driven by the product owner
- Help the product owner to effectively prioritize work using methods such as Weighted Shortest Job First (WSJF) to maximize the potential value the team can deliver
- Improve the lives and productivity of the development team by facilitating creativity and empowerment
- Champion the team process and improvements
- Influence practices to align with Agile values and principles without command and control activity
- Act as an internal team coach and protector of the team
- Keep information about the team's progress up to date and visible to all parties using information radiators
- Help team to identify, raise up and remove roadblocks
- Does not promise or schedule work on behalf of the team

RECOMMENDED

vimeopro.com/leandog/iteration-manager



"The Agile Samurai: How Agile Masters Deliver Great Software"
By Jonathan Rasmusson



"The Human Side of Agile"
By Gil Broza

Agile Project Management with Scrum, Ken Schwaber

The Daily Stand Up is an Agile ceremony where the team meets for no more than 15 minutes every morning. Each person briefly discusses what it will take for them to get their work done today, whether it would go faster if someone could help them, and any roadblocks they currently have. Daily Stand Ups reduce the need for team meetings. They encourage accountability, as team members are aware of all ongoing work. Stand Ups also allow for mid-course corrections, encourage the team to solve problems on their own and help to notify managers early if there are any roadblocks.

ESSENTIAL

- Collaborative activity with core team to share what they're working on and any roadblocks that are preventing forward progress on a story
- Occur daily at the same time for 15 minutes or less
- Everybody attends and shows up on time
- May include the customer or product owner so they can be informed and make adjustments
- There is no Stand Up leader - team facilitates

ADVANCED

- No stale roadblocks exist
- Happens daily, regardless of priorities
- Sound signal to start Stand Up (Alarm / cow bell / gong)
- Team selects Stand Up time
- Team member records the roadblocks on the whiteboard so they are visible and cannot be ignored
- Team members are allowed to “Pass” if they have no new information to share and no roadblocks

DAILY STAND UP

vimeopro.com/leandog/standup

Pictures source: Xavier Quesada Allue. Visual management. [Online] http://www.xqa.com.ar/visualmanagement/2009/04/daily-scrum-against-the-board/, 4/19/09.

SPRINT / ITERATION PLANNING

The planning ceremony is held at the beginning of each new iteration to forecast the work that the team believes they will be able to deliver in the upcoming iteration. During this session the team will review their expected availability, any work not completed in the prior iteration, and the highest priority new work in the product backlog. They will do any remaining refinement to ensure the work can be completed in the upcoming iteration and create an iteration backlog of work they believe they can complete in the upcoming iteration.

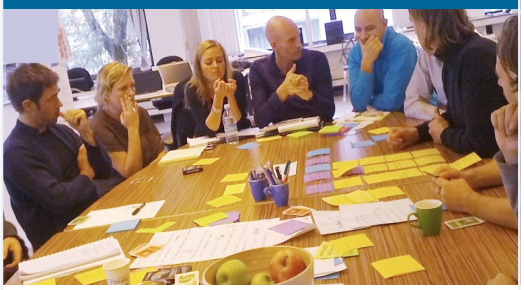
ESSENTIAL

- Meeting includes the Iteration Close
  - Review of work completed during iteration, including roadblocks
  - Review of cards not completed during iteration
  - Velocity

- Meeting includes the Iteration Open
  - Team dicusses availability and other expected impacts to velocity
  - Forecast velocity is established
  - New story cards are reviewed for business

- The product owner presents stories that may be played within the next iteration understanding that the team will commit based on their forecasted velocity and the readiness of the work

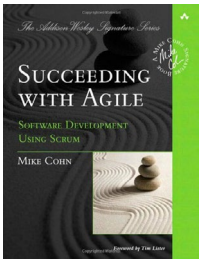
VELOCITY



"Over-emphasis on velocity causes problems because of its wide [use] as a productivity measure[ment]. The proper use of velocity is as a calibration tool, a way to help do capacity-based planning, as Kent Beck describes in Extreme Programming: Embrace Change." [1]

"The ultimate expression of agility from a software perspective is continuous delivery and deployment. Our goal should not be productivity, but to design and deploy a great customer experience quickly—again and again over time." [1]

RECOMMENDED



"Succeeding with Agile: Software Development Using Scrum"

vimeopro.com/leandog/sprint-iteration-planning

1 Jim Highsmith, http://jimhighsmith.com/2011/11/02/velocity-is-killing-agility/
Picture Source: http://d2lkacpp4m5oo7.cloudfront.net/wp-content/uploads/2010/05/AH-sprint-planning-meeting2.JPG

SPRINT / ITERATION REVIEW

The Sprint or Iteration Review is an opportunity for the development team to work directly with the customer to showcase the work the team has completed. This provides an opportunity for all customers, team members, and other stakeholders to review progress, provide feedback on new features, and prioritize any changes needed in the Product Backlog. By holding this session regularly the team maintains accountability and a close working relationship with the customer.

ESSENTIAL

- Customers and Whole Team attend
- All other interested parties are welcome
- The team demonstrates work completed during the iteration
- The customer provides feedback on the implementation
- The customer prioritizes any required changes immediatey within the Product Backlog

RECOMMENDED

"The Art of Agile Development"
By James Shore

vimeopro.com/leandog/show-and-tell

RETROSPECTIVES

Retrospectives are meetings that take place after each iteration and are designed to help the team find ways to improve their processes and output. It is critical that this is a safe environment in which to talk about what worked and what didn't work. The team votes on which of the items discussed during the retrospective should become cards to be played in the coming iteration.

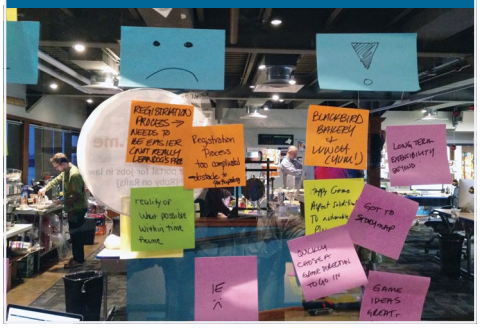
ESSENTIAL

- Core team applies continuous improvement principles
- Others outside of the development team, especially management, do not attend unless invited by the team
- The goal is to stress continuous improvement. Experimental changes, even those that fail, are encouraged
- The format is broken into five sections: Open, Gather Data, Generate Insights, Decide What To Do, Close
- The activities in each section are changed regularly and often include a version of "what worked" and "what didn't work"
- The team decides which action items become cards to be played in the next iteration
- They are conducted at a minimum at the end of each iteration
- The team consistently identifies improvement experiments for the next iteration and reviews the success of previous experiments

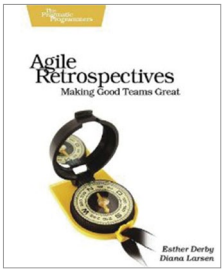
ADVANCED

- Use safety checks to indicate that team members feel safe to bring up difficult topics
- The team holds spontaneous retrospectives for significant issues
- The team experiments with alternative formats (e.g. Lean Coffee, Six Thinking Hats, etc.)

RETROSPECTIVE MEETING



RECOMMENDED



"Agile Retrospectives: Making Good Teams Great"

vimeopro.com/leandog/retrospectives

A roadblock is an obstacle that is standing in the way of the team. Management should be aware of roadblocks early so they are quickly resolved and the team can move forward in an effective manner.

ESSENTIAL

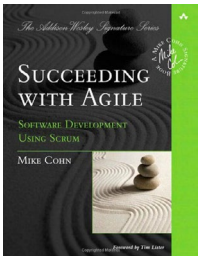
- Roadblocks are displayed on a wall in each team area
- They are prioritized and complated in order
- They are reviewed as a part of the Spring Review
- When a roadblock is being resolved, there is an owner listed on the card

LISTING TEAM ROADBLOCKS

ADVANCED

- Stale roadblocks are identified and escalated as priority issues
- Roadblock progress is tracked on a simple kanban board
- The purpose of the roadblock wall is made clear to anyone who sees it
- Senior management understands that roadblocks are visible and where to find them

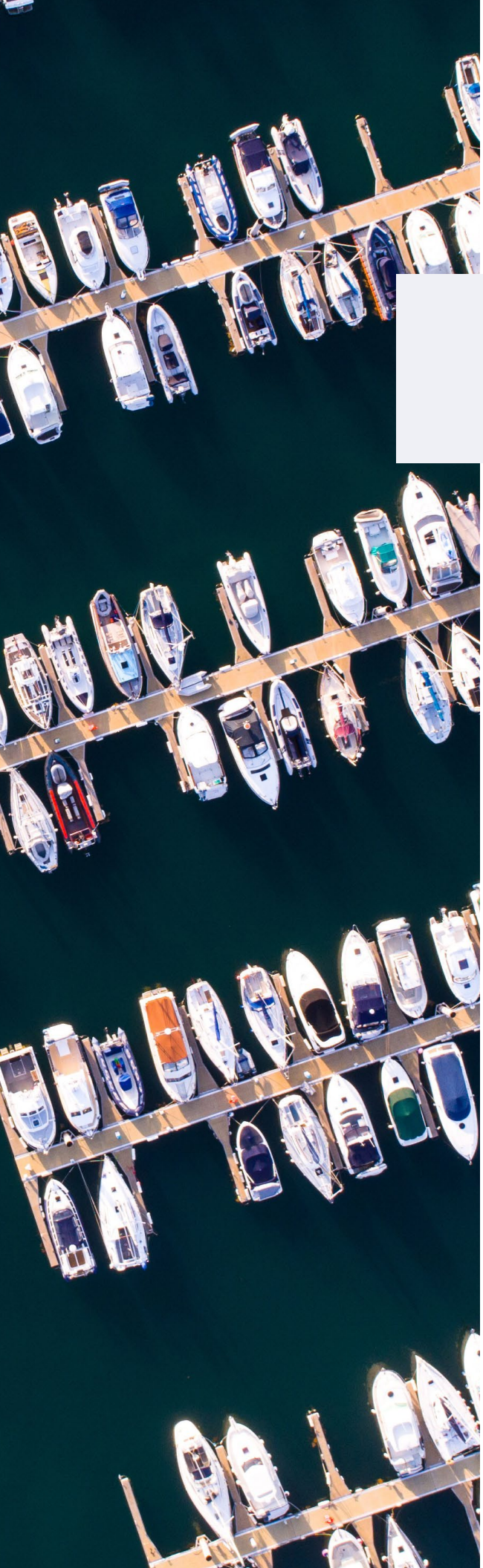
RECOMMENDED



"Succeeding with Agile: Software Development Using Scrum"
By Mike Cohn

vimeopro.com/leandog/roadblocks

Understanding people



CHAPTER 7

Product Design

IN THIS CHAPTER

- User Experience
- Roles
- Product Design Role
- Understanding the User
- Prototyping
- Feedback Loops
- How UX Fits into Agile Teams
- Beyond the Screen

USER EXPERIENCE

UX is not UI. It's about understanding the people who use our products and services. We must know what motivates them, why they need our product, what problem they are trying to solve, and how they do it today. When we truly understand our customer's articulated or unarticulated needs, we can deliver solutions that make their lives better.

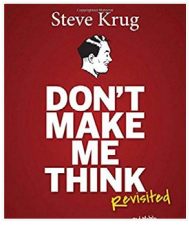
ESSENTIAL

- Starts with understanding the problem and business objective
- UX is recognized as an important role and practice to develop products
- The organization understand the value of design and how it can mitigate risk and save as products evolve
- Focus on solving the highest value items in the simplest way
- Research informs the decision making and prioritization
- Strategy cannot be done in 1 sprint
- Develop feedback loop with customers (interviews, surveys, calls, tests, in-person)
- Iterative design (refactoring is to be expected)
- Solve for 80% of users - don't get bogged down in the fringe edge cases to start
- Embrace rapid prototyping and validating with customers to tighten the feedback loop
- Define and measure success metrics
- Share the learnings throughout the team
- Do just enough design - it's not pixel perfect, it's enough to have a conversation

ADVANCED

- UX is 100% dedicated to the team, ideally sitting with the whole team
- Others on the team collaborate, co-create, and pair with UX regularly
- Everyone on the team participates in user research and user testing
- Ongoing and frequent interactions with customers
- Inviting customers into the design process
- Cross-functional pairing
- Teams take the thinnest valuable and vertical slice across the story map
- Release plans are user outcome focused
- Make decisions based on data - not preferences or the highest paid person in the room
- Radiate real time KPIs and product metrics (i.e. performance, churn, abandonment/conversion)
- Use OKRs to guide the team's priorities
- Business value and growth can be directly correlated to design thinking

RECOMMENDED



"Don't Make Me Think"
By Steve Krug

"Lean UX: Designing Great Products with Agile Teams Hardcover"
By Jeff Gothelf, Josh Seiden

ROLES

User Experience is often misunderstood and titles are nebulous at best. Depending on the size of an organization and the amount of teams, these roles can be more specialized. For example, a large financial institution may have people entirely dedicated to research and other focuses on crafting the interfaces. On the other hand, a small startup may need one person to do it all.

PRODUCT DESIGNER/UX DESIGNER

The most important quality of this role is to balance the needs of the customer with the business outcomes. This requires a multi-disciplinary background, which can include design, research, engineering, business, analytics, finance, and marketing. They must be able to seamlessly weave customer focused thinking within an organization's culture. A successful UX designer has the ability to provide customers with a joyful, elegant, and simple product experience. Accomplishment is factored through understanding the articulated and unarticulated needs of customers, understanding business requirements, generating direct customer feedback, iterative design processes, and learning from measurable indicators. This role is versed in process flow mapping, content strategy, information architecture, and sets the experience vision and strategy.

UI DESIGNER

Think of UI as the bridge between UX and Engineering. It's vital to understand there is distinct difference in UX and UI. Their core responsibilities will include, prototyping, providing visual designs, interactive design, information architecture, and more. Pairing research gathered from the UX Designer, a good UI Designer should be able to translate findings into an immersive user interface. UI is not front-end engineering.

VISUAL DESIGNER

Creating a strong brand and visual language is the primary responsibility of the Visual Designer. They are in charge of establishing the design systems, style guides, communication templates and graphical assets. Often this role will support other teams like Marketing and Sales.

At the core of any product or service we create, there is a person. Many organizations think they know what their customers want but they never actually talk to them and have the “if we build it, they will come” mindset. Building something and hoping customers use it or buy it is extremely risky and potentially very wasteful. None of us wants to build something that misses the mark or worse yet, doesn’t get used.

It’s easy to think we know what our customers want because we are people too and we know what we like and need. However, we have to set aside our preconceived notions and remind ourselves that we are not the user. We have to get out of the building and talk to real users. We need to learn what our customers need, why the need it, what motivates them, and what pain points they have in their current solution.

When we start with the user at the center, we stay focused on creating something that meets our customers needs, exceeds their expectations and makes their lives easier.

ESSENTIAL

- **Personas** - help you design and build features, products and services that meet your customers need. Personas are based on research with real people. You can start with a hypothetical proto-persona that articulates what you know or think you know about a user.. Then as you learn and gain insights through research, you can evolve the personas based on real data. Personas typically include user behavior, goals, motivations, needs, and key scenarios. Using Personas, help you ensure you’re creating something that your customers really need and value.

- **Journey Map or Experience map** - an activity that plots out the key touchpoints a user has with your product and/or service. Often maps will start with the discovery step - when and how does a user learn about the product and why do they decide to move further along with engaging with it. For many companies, visualizing the journey surfaces the gaps between systems or channels.

UNDERSTANDING THE USER

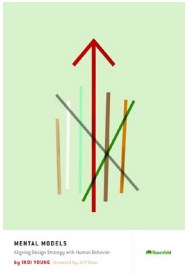
ADVANCED

- **Empathy Maps** - a framework for distilling what you learn about users. Empathy maps are meant to help you and your team gain empathy for your users tasks, feelings, influences, pain points and goals. Using Empathy Maps is a quick way to drive out personas from your research. David Gray and Paul Boag both have excellent maps to use.

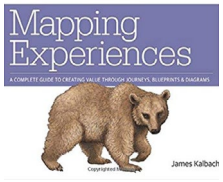
- **Value Proposition Canvas** - a tool created by Alexander Osterwalder that helps frame the product or service around the value you provide and the what the customer need.

- **Opportunity Solution Tree** - a reframing exercise to help teams step back and look at the desired outcome and then find opportunities for innovation.

RECOMMENDED



"Mental Models: Aligning Design Strategy with Human Behavior"
By Indi Young



"Mapping Experiences: A Complete Guide to Creating Value through Journeys, Blueprints, and Diagrams"
By James Kalbach

- **Value Proposition Canvas**
  https://strategyzer.com/canvas/value-proposition-canvas
- **How to use Value Proposition Canvas**
  https://designabetterbusiness.com/2017/10/12/how-to-really-understand-your-customer-with-the-value-proposition-canvas/
- **David Gray's Empathy Map**
  https://gamestorming.com/wp-content/uploads/2017/07/Empathy-Map-Canvas-006.pdf
- **Paul Boag's Empathy Map**
  https://boagworld.com/usability/adapting-empathy-maps-for-ux-design/
- **Teresa Torres - Opportunity Solution Tree**
  https://www.producttalk.org/2016/08/opportunity-solution-tree/

Teams use prototypes to quickly test out ideas. Sometimes it's to prove it's technically possible and other times it helps teams learn how people use it and if they want it. Prototypes are not meant to be perfect or even beautiful - they are meant to be a tool to test broad concepts.

ESSENTIAL

- Choose the simplest level of fidelity to test your ideas. Early on, you can make a prototype out of paper. As the concept matures, you will likely need to increase the fidelity to help your users give you more detailed feedback on how it meets their needs
- Rapidly iterate your concepts as a team based on what you learn
- Prototypes can help prove out technical feasibility
- Building a prototype helps the team tell the story to stakeholder for additional buy-in
- There are plenty of tools that help you build a prototype - some are more complicated than other but can offer testing on mobile and leveraging logic to simulate app behavior. Cooper has done an excellent review of the tools through a variety of dimensions including the ability to export code
- If anyone on the team is reluctant to draw, tell them that most ideas can be quickly sketched to convey the concept using squares, rectangles, circles, triangles, lines and dots
- Test early to validate direction and test your product's riskiest assumptions. It's better to learn sooner than later if we are right or wrong. And it's cheaper to fix issues or missed requirements in early phases like concepting - changes at later phases increase the cost by a factor of 10

PROTOTYPING

ADVANCED

- Animated prototypes are possible. With the advancement of some tools like Adobe XD, Figma, and Invision Studio, you can simulate motion, gestures, and native app behaviors directly in the prototype
- Use Storyboards to articulate the concept and get feedback on how users react to it.
- Leverage 3D printing and sensors to test out connected experiences
- Use Design Studio Method to involve the entire team in ideating on how to solve the problem. Start by individually sketching concepts and then go through a round of pitch and critique where each person pitches their concept for 2 minutes. Then for 3 minutes, the rest of the team provides feedback on how the concepts met the users goals. Then build on the best ideas and collaboratively generate new concepts. You'll be surprised at how rapidly ideas come and that great ideas come from everyone on the team!
- Build directly with the customer, invite the customer in or go where they are at and iteratively build concepts that they can test as you go

RECOMMENDED

- **Cooper's Designer's Toolkit: Prototyping Tools**
  https://www.cooper.com/prototyping-tools
- **Facilitating an Effective Design Studio Workshop**
  https://www.nngroup.com/articles/facilitating-design-studio-workshop/
- **Nordstrom Innovation Lab - Sunglass iPad App**
  https://vimeo.com/83614797

Feedback loops are important for teams to learn and mitigate risk. When you get feedback from customers, users, and the market, you are able to make informed decisions. Adopting gathering data and insights as a regular practice allows you to create a continuous feedback loop. The tighter the feedback loop, the faster you can hone your product's direction and evolution. Embrace the Build, Measure, Learn mindset.

ESSENTIAL

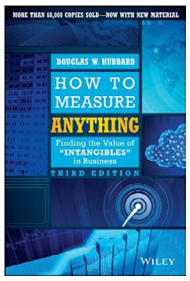
- **Find your data.** Identify your sources of data that you have now - what do you know about your customer. Typically, support and sales will have great insights into what customers are asking for or what they are having trouble with.
- **Add analytics.** There are plenty of tools that you can add to your digital products that help you understand your users behavior (i.e. Amplitude, Google Analytics, Keen). When you add in an analytics tool carefully plan out what you want to track and what you want to answer.
- **Define actionable metrics.** Articulate what you are trying to learn and your goal. Good metrics help you see or understand what is going on with your users. Stay away from vanity metrics don't give you any real actionable insight (i.e. number of downloads or page views).
- **Conduct regular user research to get rich insight.** Interviewing and observing your users (current, past and potential) are the best ways to understand their needs, motivations, and behaviors. Both methods let you have a conversation with your users to understand what they are thinking, why they did something, why they need your product and how the product can exceed their expectations. Every time you talk with a customer, you will learn. Additional methods teams use to get feedback include surveys, A/B tests, pilots, and beta releases.
- **Make sense of what you are learning.** A quick way to distill what you learned in your user research activities is to create an affinity map. Write each insight onto a sticky note and put on a wall. Review and cluster for themes. Some tools are emerging to help teams capture, store and analyze (i.e. Product Board, AirTable, ProdPad, Shipright).

FEEDBACK LOOPS

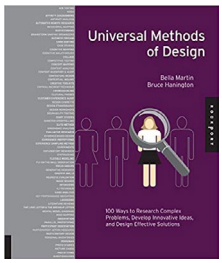
ADVANCED

- **Use a Dashboard.** Set up a place to radiate key customer metrics like funnel analysis, performance, active users, and churn rate.
- **Connect with your customers in real-time.** There are tons of tools that let you talk to your customers and collect feedback at just the right moment (i.e. Intercom, Mixpanel, Smooch)
- **Set up a User Research or Experience Panel.** This gives you a channel to reach customers and get feedback quickly. Let your customer opt-in to join the panel. Tell participants that this panel lets them help make the product better Use this panel to conduct surveys, interviews, field studies, user tests, and even focus groups. You'll save time and money on recruiting for your research needs and your customers will feel that you care.
- **Review data and feedback as a team.** Each sprint, review what you learned and how that influences the product direction. Then decide what feedback you need next and what question you are trying to answer.
- **Use OKRs.** Objective and Key Results (OKRs) are used by high performing companies like Google and Walmart to align the organization, teams and individuals on measurable goals. John Doerr, who introduced OKRs to Google, has a simple formula to follow: I will (Objective) as measured by (this set of Key Results). Adopting OKRs takes time and adjustment but it makes sure everyone is going in the right direction.

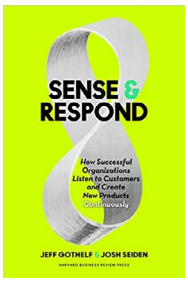
RECOMMENDED



"How to Measure Anything: Finding the Value of Intangibles in Business"
By Douglas W. Hubbard

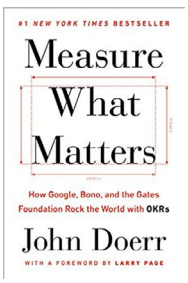


"Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions"
By Bruce Hanington, Bella Martin



"Sense and Respond: How Successful Organizations Listen to Customers and Create New Products Continuously Hardcover"
By Jeff Gothelf, Josh Seiden

"Validating Product Ideas: Through Lean User Research Paperback"
By Tomer Sharon



"Measure What Matters: How Google, Bono, and the Gates Foundation Rock the World with OKRs"
By John Doerr

"The Beginner's Guide to OKR"
By Felipe Castro

HOW UX FITS INTO AGILE TEAMS

UX is a key role on the Agile team. Mature teams have UX embedded on the team and 100% dedicated to the work that the team is doing. The ideal ratio of UX to Engineers is 1:5. This allows for regular cross-functional pairing, increased empathy, and knowledge transfer across roles. Having UX on the team helps the team narrow in on what to build so the team can have greater certainty that they are solving the most impactful problems.

ESSENTIAL

- Dual Track is a way to manage the work for UX and the Engineers
  With this approach, UX is at least 1-2 sprints ahead of development. Each track has a separate backlog. If you're using a tool like Jira or Trello, each track is a separate board
- Represent the work. Stories are written for the UX work and tracked just like the developer story cards. UX work is planned with each sprint and reviewed in Show and Tells
- Make the work visible. Any learnings or artifacts that come of the UX efforts should be posted up for the team to see, review, and discuss. Having work up on walls sparks great discussion, questions and ultimately, collaboration
- UX collaborates with the rest of the team to work through discovery, research, design, and execution

RECOMMENDED

- **Integrating Agile and UX design: Evolving the process of digital product creation.**
  https://www.oreilly.com/ideas/integrating-agile-and-ux-design
- **Dual Track Development is not Duel Track**
  https://www.jpattonassociates.com/dual-track-development/
- **Google Ventures Design Sprint resources**
  http://www.gv.com/sprint/
- **Balanced Team**
  http://www.balancedteam.org/

ADVANCED

- Single Track folds the UX stories into the one workstream for the team. Stories are assigned each sprint and reviewed at Show and Tell. There is only one backlog/board for the whole team
- Discovery is a team sport. The whole team plans and does the work necessary to reduce uncertainty and learn. This happens when vision and product strategy need to be further defined or when we don't know the best way to solve the problem. It can be as short as a Spike or as long as a few sprints
- Design Sprints are just 5 days and the whole team is involved. The goal of th sprint to to understand the problem, create simple solutions and get validation with real users at the end of the week. The agenda for the sprint is roughly: Monday - structured discussions, Tuesday - sketch/design studio method, start recruiting users for testing, Wednesday - critique, refine, Thursday - prototype, Friday - test. At the end of the sprint, the team will have learned how user needs were met (or not) and what the next question the team needs to answer
- Co-creation is a beautiful thing. UX and an engineer pair to create the design. This can be paired up scribbling at a whiteboard or designing directly in a design tool like Sketch, Figma or Adobe XD
- Embrace and live the Balanced Team mindset. Balanced Team is a global movement of people who value multi-disciplinary collaboration and iterative delivery focused on customer value as a source for innovation. Focused on continuously improving processes, team and selves so we can do good work as happy teams

RECOMMENDED

"Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days"
ByJake Knapp, John Zeratsky, Braden Kowitz

BEYOND THE SCREEN

As technology becomes more and more ubiquitous, the experience is moving out beyond the screen. The customer experience needs to be orchestrated across devices and channels. An end to end holistic lens must be applied to continue to innovate and deliver remarkable experiences.

END TO END EXPERIENCE

Understanding the full scope of how a user experiences your product is the only way to truly deliver a complete positive user experience. To gain a full scope of understanding you must look at marketing, sales, customer support, and other unconventional ways a human may interact with your product and company. For Example: AirBnB must not only consider how a host and traveler experience their service, but they also must consider how they interact online and in the real world. Additionally, with an open-market product and their business model, the cost of doing business for them will require them to face controversy in the media. Damage control like how they handle lawsuits, PR, and marketing will result in future poor or positive customer success.

SERVICE DESIGN

As more and more companies provide services, either as their main business or as a complement to their products, it's becoming increasingly important to craft great experiences outside of digital. Applying the same methods, principles and practices we use to develop technology products helps these organizations gain an edge over their competition and delight their customers. A service is experienced. If it's poorly orchestrated, customers are frustrated and will take their business elsewhere. On the other hand, if it's an unexpectedly great experience, customers will tell others. Designing great services requires deeply understanding the touchpoints a customer has with the service and what it takes to support those touchpoints. Whether it's communication or getting support, a great experience can be designed.

NO UI - CHATBOTS, AI, VR/AR/MR, BLOCKCHAIN

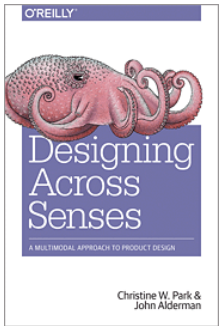
Humans have always had abstract interactions with machines. As technology continues to evolve into User Interfaces, like AI & Voice Commands, UX Designers will have to adopt new skill sets. Data will be the driver of most product decisions, which will require designers to dig deeper into data patterns and even query more complex sets of information. Mastering disciplines like psychology, biology, and science will become necessary. For instance, if a product is focused on voice commands, tone, cadence and timbre of voice become the mediums for building an effective UX.

INCLUSIVE DESIGN

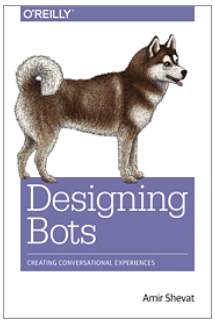
A product team must factor in all customers personas, regardless of race, culture, beliefs, or sex to truly be inclusive. Designing for all types of customers is vital for maintaining a positive audience sentiment. Leaving out a customer segment can lead to poor customer experiences, offending a segment of your customer base, brand/product negative public opinion, and sometimes endangering end users. With an inclusive design approach, everyone benefits and the design is stronger. For example, the curb cut-outs at intersections allow for anyone with impaired mobility to easily navigate and someone pushing a stroller or bike also appreciates the ease of crossing the street.

RECOMMENDED

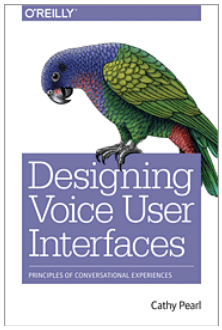
"Service Design: From Insight to Implementation"
By Andy Polaine, Ben Reason & Lavrans



"Designing Across Senses: A Multimodal Approach to Product Design"
By Christine Park, John Alderman



"Designing Bots: Creating Conversational Experiences"
By Amir Shevat



"Designing Voice User Interfaces: Principles of Conversational Experiences"
By Cathy Pearl

- **Microsoft - Inclusive Design**
  https://www.microsoft.com/design/inclusive/

The process of creating value

CHAPTER 8

Developer

IN THIS CHAPTER

- Developer Role
- Collective Code Ownership
- Continuous Integration
- Simple & Evolutionary Design
- Paired Programming (Pairing)
- Test Driven Development
- Technical Debt
- Spikes

DEVELOPER ROLE

A developer is responsible for creating high-quality working code that meets the expectations of the product owner.

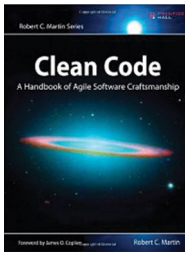
ESSENTIAL

- Understands the business intent of the story card
- Works with the product owner and tester to come up with an approach on how to provide the value
- Save both test and code in a source control
- Build code to satisfy the tests and conditions of the story
- Write and code with the purpose of clearly communicating the intent to the next developer that will look at the code, and be satisfied that what you have coded is ready for production use
- Build tests that measure whether or not the implementation is working
- Continually improve the test and code base with refactoring
- Pair with a fellow developer to ensure someone else knows this section of the application, and hold each other to high levels of design and code quality

ADVANCED

- Undertake software craftsmanship to help maintain development speed, quality and efficiency
- Commit to be one of the “Three Amigos”/ “Fantastic Four” and encourage collaboration between product owner and quality assurance
- Build code to get acceptance tests to pass (ATDD - Acceptance Test Driven Development)

RECOMMENDED



"Clean Code:
A Handbook of
Agile Software
Craftmanship"
By Robert C. Martin

COLLECTIVE CODE OWNERSHIP

Collective Code Ownership is a practice where all team members share responsibility for code quality. It also allows each developer to change any piece of the code at any time. Collective Code Ownership also helps to eliminate “specialization”, as everyone is expected to fix problems when they are discovered.

ESSENTIAL

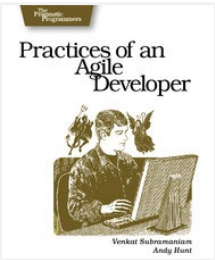
- All developers are responsible for the code; everyone has the ability to change the code
- Developers commit to writing outstanding code from the start, instead of doing a half-hearted job and expecting someone else to fix the mistakes
- Check egos at the door
- Works best when coupled with Paired Programming and Test Driven Development
- Eliminate knowledge silos by encouraging developers to work in unfamiliar areas of code

COLLECTIVE CODE OWNERSHIP

We are members of a community that owns the code

Anyone can modify any code at any time
Team has a single style guide and coding standard
Original authorship is immaterial
Plentiful automated tests increase confidence and safety

RECOMMENDED



"Practices of an Agile Developer"
By Venkat Subromanian & Andy Hunt

vimeopro.com/leandog/collective-code-ownership

CONTINUOUS INTEGRATION

Continuous Integration is the practice of ensuring all code changes are compatible with the existing shared codebase by assimilating each change into the codebase as soon as possible after a developer submits the change. The entire system is tested in order to confirm the health of the code and verify that the changes made are compatible with the shared codebase. Code compilation and testing can be automated.

ESSENTIAL

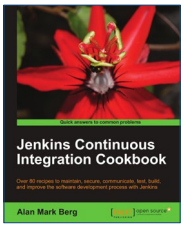
- System is built and all unit tests run at least once a day
- The entire build and test run takes less than ten minutes
- Team is notified whenever a build fails
- Fixing a broken build is the team's highest priority

ADVANCED

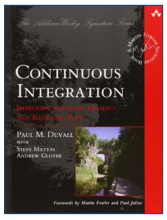
- A system is built and all tests are run every time a developer checks in code
- Visual indicators in the room that show the build status

vimeopro.com/leandog/continuous-integration

RECOMMENDED



"Jenkins Continuous Integration Cookbook



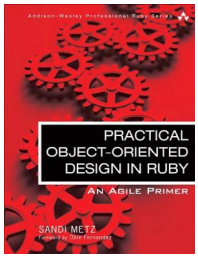
"Continuous Integration: Improving Software Quality and Reducing Risk"
By Paul M. Duvall

Simple & Evolutionary design is a system which minimizes the amount of up-front design done before coding begins. Instead, the design of the system grows as it develops, emphasizing simple architectures and designs that provide adaptability to change. Simple designs also allow teams to have increased velocity as they spend more time providing customer value.

ESSENTIAL

- “Simple is best” approach to software design where refactoring is an integral part of the code simplification process
- Avoid building unnecessary architecture
- Design is done just in time and evolves with the app
- Spikes are used to mitigate risk in design or investigate a new domain
- Application shouldn't contain any knowledge duplication – Don't repeat yourself
- Code should express intent – Principle of least astonishment

RECOMMENDED



"Practical Object-Oriented Design: An Agile Primer Using Ruby"
By Sandi Metz

"Clean Architecture"
By Robert C. Martin

PAIRED PROGRAMMING (PAIRING)

Paired Programming is the technique of collaborating two team members at one computer. With two minds in constant collaboration, paired programming encourages sharing knowledge and catching bugs early by continual code review and design. The knowledge moves across the team more quickly and ensures that the best practices are followed.

ESSENTIAL

- All code is produced by a pair of developers working together on one story at one workstation
- It's encouraged for all roles to pair with anyone, anytime - switching every few hours is optimal
- Pairs change frequently, and very few knowledge silos exist
- No coding without a pair partner
- Don’t underestimate good hygiene
- Environment supports pairing: desks are easy to access, office doesn't have obstacles or obstructions
- Developers do not have their backs to the team, they sit beside or across from each other

ADVANCED

- Frequent pair switching - switch pairs every 4-8 hours
- Remote Pairing via video conference
- Pairs use focusing techniques (e.g. Pomodoro) to stay on task

RECOMMENDED

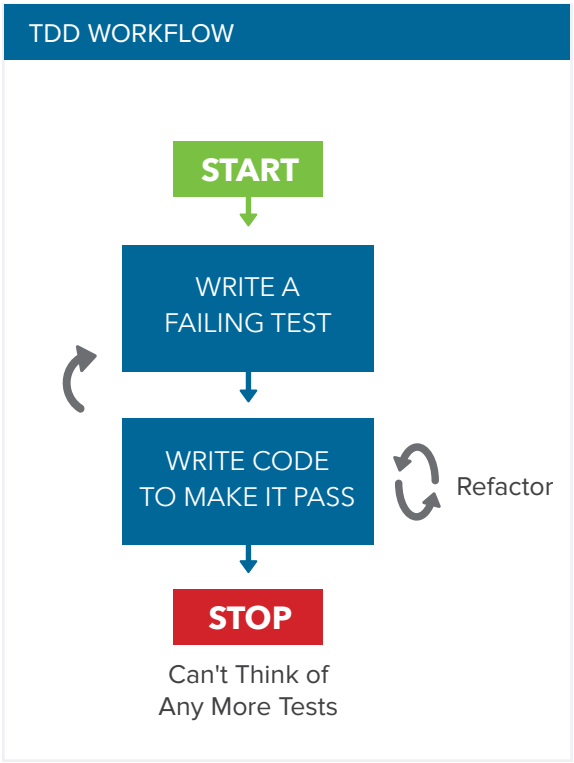
"The Art of Agile Development"
By James Shore

"The Cost & Benefits of Pair Programming"
By Alistair Cockburn
vimeopro.com/leandog/paired-programming

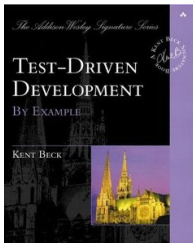
Test Driven Development is a software development technique where developers write a test prior to any new application code and use short development cycles to design the product. There are three stages: (1) Write a failing test, (2) Write code to make it pass, (3) Refactor and continue cycle until the developers cannot think of any more tests. Using TDD results in fewer defects, and provides a fast feedback loop, allowing for fearless refactoring of the system.

ESSENTIAL

- Tests are written prior to any code
- TDD workflow:
  - Write a failing test
  - Write code to make the test pass
  - Refactor and continue cycle until the system fulfills all required functionality and satisfies all known use cases
- Forces testable designs/architectures
- Produces clean code
- Tests to provide a safety net as application design evolves



RECOMMENDED



"Test Driven Development: By Example"
By Kent Beck

vimeopro.com/leandog/tdd

TECHNICAL DEBT

Technical Debt is the gap between the right solution and the solution that currently exists. As technical debt accrues, the more difficult it becomes to reconcile the differences and deliver what is needed. Technical debt can be incurred as a strategic initiative but needs to be repaid within only a few iterations.

ESSENTIAL

- Clean code
- Good test coverage
- A learning objective
- Payback plan
- A truly informed product owner

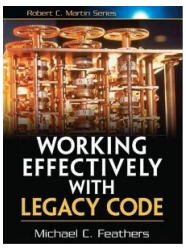
ADVANCED

- Technical debt is tracked / acknowledged / disclosed
- Technical debt is continuously minimized

SEE THE WHOLE - LEAN SOFTWARE DEVELOPMENT

RECKLESS	PRUDENT
"We don't have time for design"	"We must ship now and deal with the consequences"
DELIBERATE	
INADVERTENT	
"What's Layering?"	"Now we know how we should have done it"

RECOMMENDED



"Working Effectively with Legacy Code"
By Michael Feathers

vimeopro.com/leandog/technical-debt

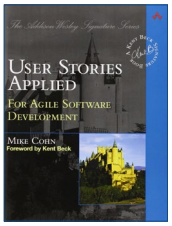
A Spike is a time-boxed experimental story card that is included in the iteration cycle to determine an estimate but stops short of completing the story. Spikes are an excellent way to try out something quickly and are usually a few hours to a couple days in length.

ESSENTIAL

- Time-boxed
- Few in number
- A learning / discovery objective
- Code produced should be thrown away
- Results should be communicated back to the team and product owner along with demos
- Pulled forward in the release plan to reduce risk
- Played with enough leeway to adapt the plan
- Incorporate spikes into planning
- Estimate the spike like anything else

RECOMMENDED

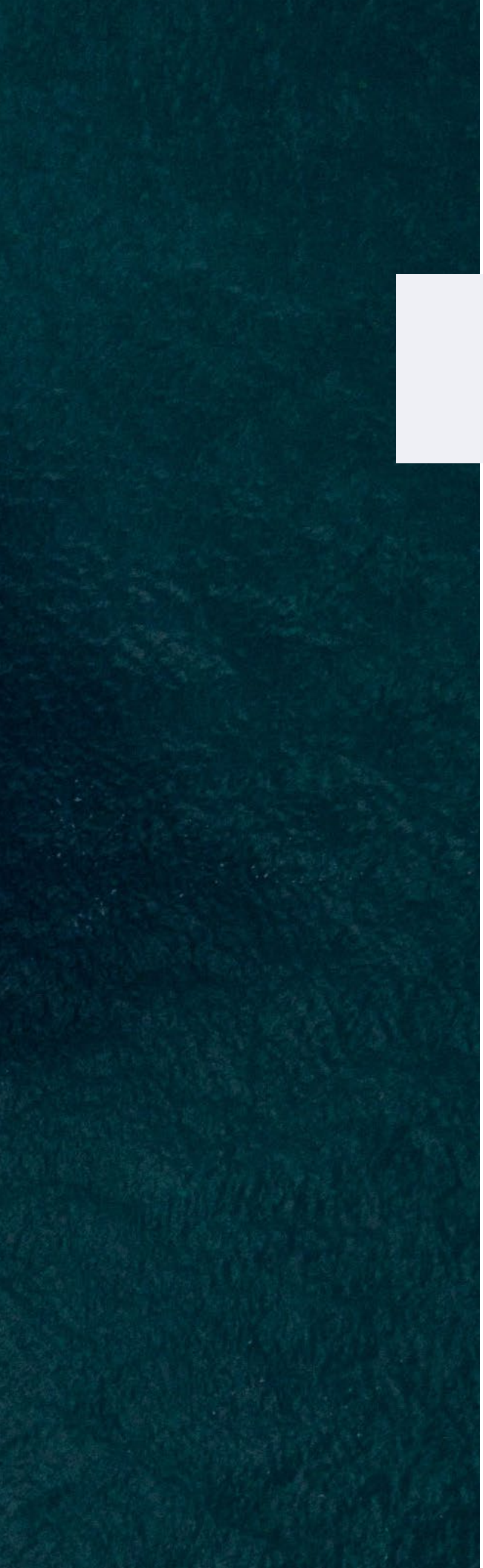
"The Art of Agile Development"
By James Shore



"User Stories Applied: For Agile Software Development"
By Mike Cohn

vimeopro.com/leandog/spikes

The process of clarifying value



CHAPTER 9

Testing

IN THIS CHAPTER

- Delivery Quality
- Tester Role
- Exploratory Testing
- Behavior Driven Development

DELIVERING QUALITY

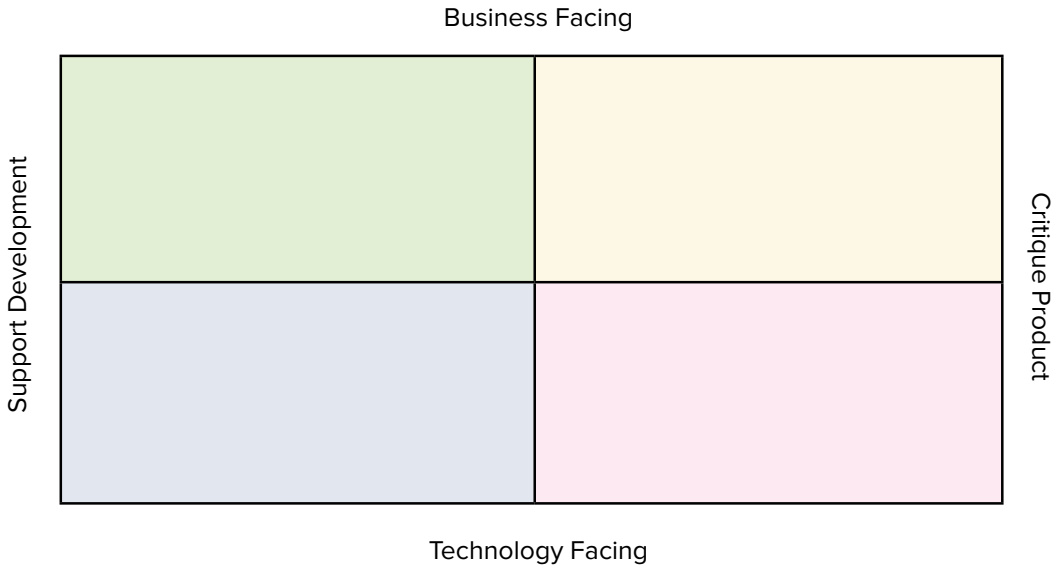
On an agile team, delivering quality is a team sport. There is no one person or role who is responsible for testing or assuring quality. Throughout this guide, we highlight many ways that the team members and processes help to provide insight into the quality of the system - through collaboration and pairing, direct inspection, automation, and targeted feedback activities.

Some of the many agile testing and quality-related team activities described in this guide include:

Acceptance Testing and Validation	• Story card writing, INVEST, 3 C's • Customer collaboration, Sprint Reviews & demos • Design Thinking, UX, prototype testing, feedback loops
Functional verification	• Developer role, pairing • Continuous Integration (CI) (for automated regression testing) • Test-Driven Development
Code Quality	• Collective code ownership • Technical Debt
System Quality	• DevOps, Continuous Delivery • Infrastructure as code • Continuous Improvement

Agile teams do not need dedicated testers to assure quality, so the title “Quality Assurance” does not fit because the whole team is responsible for the quality and value they deliver.

Brian Marick, one of the Agile Manifesto signatories, offered a way to visualize the different kinds of testing performed by agile teams using a set of quadrants. This matrix was popularized by Lisa Crispin and Janet Gregory in their Agile Testing books.



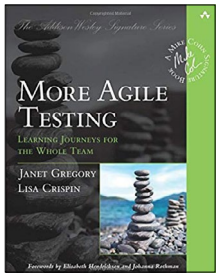
As an activity to you, the reader, use the testing matrix above to answer the following questions:

1. Which team members/roles generally fit into which quadrants above?
    - Which quadrant are you less certain about? Who else in your organization may help you with these kinds of feedback activities?

2. Where do the agile testing and quality team activities listed above fit?

Find a LeanDog coach to check your answers!

RECOMMENDED



"More Agile Testing"
By Crispin and Gregory

TESTER ROLE

Every person on an agile team is responsible for performing testing activities and providing valuable quality feedback. The tester role helps everyone on the delivery team understand the current quality and production-readiness of the software.

ESSENTIAL

- Work with the product owner and analysts to define acceptance criteria for stories and requirements
- Pair with developers to write automated functional and acceptance tests
- Cooperatively create test strategies to help team achieve required testing coverage
- Perform exploratory testing to gain insights into additional aspects of system quality
- Participate in team activities to continually improve the team and delivery processes

ADVANCED

- Actively seek out “Three Amigos” opportunities to discuss ambiguous requirements and to reduce uncertainty among team members
- Help lead collaborative test design activities such as Discovery and Specification workshops in Behavior-Driven Development (BDD)
- Demonstrate testing techniques, tools and approaches to team members to elevate their testing skills and effectiveness
- Assist with automation efforts
- Provide assistance and insights to improve test environments and test data management

EXPLORATORY TESTING

Exploratory Testing is a method of rapidly learning something specific about a product. Because the end state is unknown, this style of testing cannot be scripted or automated, and instead relies upon a structured learning approach to quickly iterate to a new insight or understanding.

Elisabeth Hendrickson defines Exploratory Testing as "simultaneously learning about the system while designing and executing tests, using feedback from the last test to inform the next." This approach leverages human ingenuity and creativity to rapidly explore different parts of a system to achieve clarity on some aspect of quality.

ESSENTIAL

- Manual tests that may occur before, during or after a story is developed
- Need a specific guiding question, hypothesis, purpose, mission or charter
- May be time-boxed (~ 30 - 120 minutes) to focus the effort
- Requires some reviewable output to share with team members - e.g. test notes, screen shots, video, log files, etc.
- Outcome is generally a specific understanding of a feature or system that may lead to new automated tests or other product improvements
- Useful to provide insights beyond the automated testing coverage already in place
- Whole-team exploratory testing activities have different names - bug bashes, smash-athon, etc.

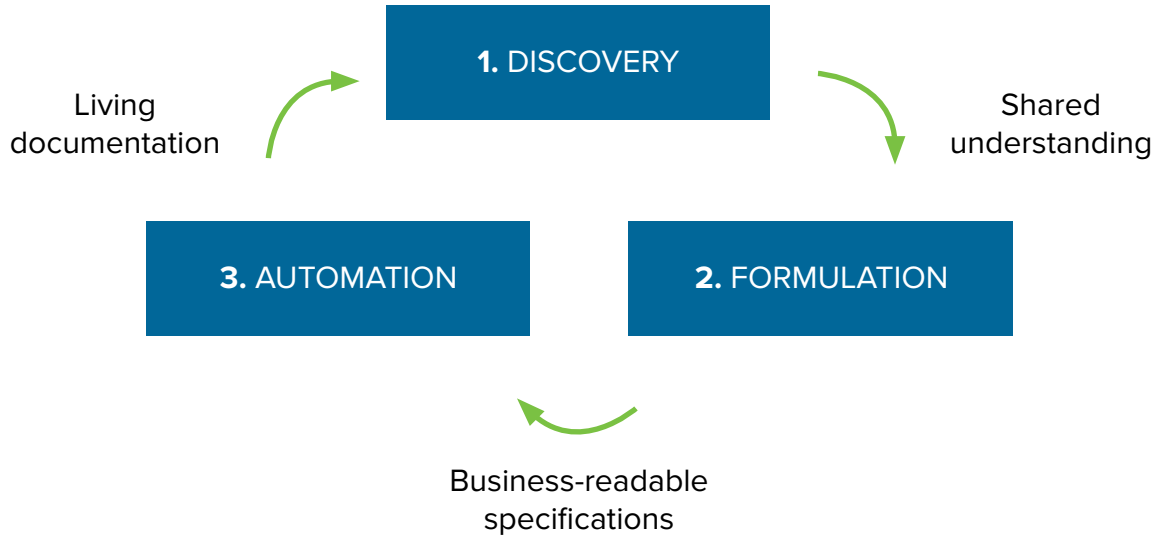
ADVANCED

- Use a specific format, structure or tools to create and manage test charters and test outputs
- Leverage automation to help generate more system information quickly
- Apply Exploratory Testing skills during "Three Amigos" discussions and during team BDD workshops

BEHAVIOR-DRIVEN DEVELOPMENT

Behavior-Driven Development (BDD) is a collaborative software development approach that seeks to simultaneously discover and deliver customer value. BDD encourages teams to use concrete examples to describe the desired application behavior and to surface uncertainty so we may deliver software that matters. It requires a shift in how team members interact with each other. The transition to BDD can be a challenge, but the payback is significant.

BDD is made up of three main practices and three outcomes:



DISCOVERY

- The Discovery Workshop is a time-boxed activity where team members get together and collaboratively illustrate the desired application behavior using specific examples
- It takes place just in time, before any development work begins on a particular user story. It may happen during Backlog Refinement, or as a separate activity to help get stories ready for development.
- This is a facilitated discussion similar to the Three Amigos that involves the business (PO and any subject matter experts), developer, and tester who may help explore the topic with relevant What-If questions to uncover unknowns.
- The primary outcome of this activity is a shared understanding of the desired application behavior. Additional outputs of the discovery workshop include: questions for follow-up, smaller stories, additional business rules, specific examples, and some light design artifacts.

FORMULATION

- This activity follows the discovery workshop. The team formulates the concrete examples into business-readable specifications to ensure everyone on the team is speaking and using the same language. These become the user story acceptance tests that guide the feature development.
- Gherkin (Given-When-Then) notation is a popular human-readable format for these specifications, although tables and other formats may also be used. (See example below)

FORMULATION: GHERKIN EXAMPLE FOR A BUSINESS-READABLE SPECIFICTION

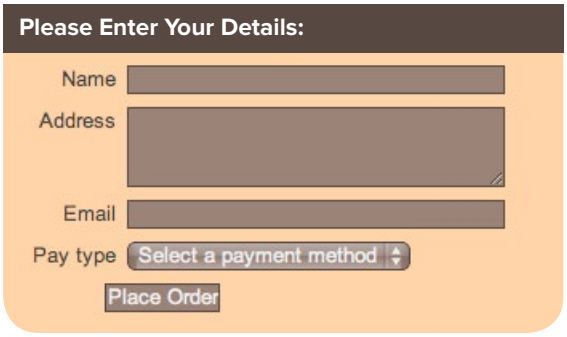
```
FEATURE: Checkout after selecting a puppy

In order to complete my online adoption
As a customer of the puppy adoption site
I want to be able to checkout by providing
necessary information.

It is important to capture the necessary
information we require to process
the adoption. We need to create a screen that
captures the name, address,
email address and payment type of the customer.

Scenario: Name is a Required field
  Given I am on the puppy adoption site
  When I adopt a puppy leaving the name field blank
  Then I expect the error message "Name can't be blank"

Scenario: Thank You message displayed when adoption completed
  Given I am on the puppy adoption site
  When I complete the adoption of a puppy
  Then I expect to see "Thank you for adopting a puppy!"
```



- The PO owns these specifications although other team members may help update them. The scenarios described here should be rooted in the problem domain and make sense to everyone so they may provide a shared source of truth throughout development.

BEHAVIOR-DRIVEN DEVELOPMENT

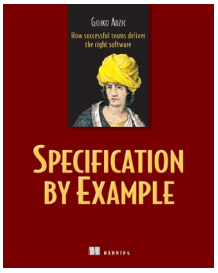
AUTOMATION

- Automation is a development activity, not a testing one. This BDD step leverages the programming expertise from the developers to select the right automation tool and to automate the formulated business (gherkin) scenarios generated above
- There are many BDD automation frameworks that may be used here, including these open source tools: Cucumber, Serenity, Behave, SpecFlow, and JBehave
- The primary outcome for this activity is to turn the formulated acceptance tests above into living documentation. Living documentation provides a shared view of what has been delivered and always tells you when it is out of date
- BDD automation works best when: it is integrated into your Continuous Integration practice, all the acceptance tests are automated prior to the completion of development (i.e. as part of Definition of Done), and these behavior tests are run continuously to provide fast regression test feedback

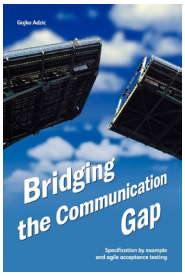
ADVANCED

- An experienced tester participates in each of the activities above in new ways to help the team members rapidly understand and deliver software that matters
- During the Discovery workshop, a tester may apply exploratory testing skills to help everyone rapidly learn and discover more about the feature. This may take the form of asking appropriate What-If questions or using whiteboarding skills to explore user or application behaviors
- Formulated behavior (Gherkin) scenarios are NOT test cases. It is important to keep the language focussed on the desired user behavior and not on detailed steps through an application interface. Writing good behavior scenarios is a team sport that takes time and practice
- Testers used to automating tests may have experience with certain tools such as Selenium. The BDD automation here is a programming activity owned by developers, so testers should pair with developers to ensure the maximum automation effectiveness achieved by the team

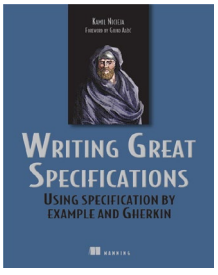
RECOMMENDED



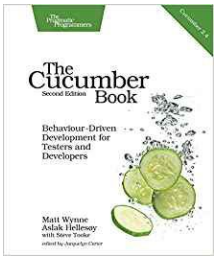
"Specification by Example"
By G. Adzic



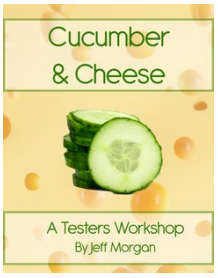
"Bridging the Communication Gap"
By G. Adzic



"Writing Great Specifications"
By K. Nicieja



"The Cucumber Book: Behaviour-Driven Development for Testers and Developers"
By M. Wynne and A. Hellesoy



"Cucumber & Cheese: A Testers Workshop"
By J. Morgan

Encouraging continuous improvement

CHAPTER 10

DevOps

IN THIS CHAPTER

- DevOps
- Continuous Improvement
- Continuous Delivery
- Infrastructure as Code
- Shift-Left on Security
- Test Automation

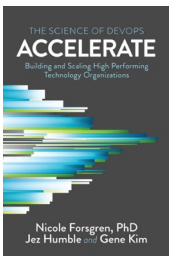
DEVOPS

Agile software development aims to break down the silos between requirements analysis, testing and development. Deployment, operations and maintenance are other activities that suffer a similar separation from the rest of the software development process. The DevOps movement is aimed at removing these silos, encouraging continuous improvement at all levels, and leveraging automation to work smarter, not harder.

ESSENTIAL

- Continuous Improvement
- Continuous Delivery
- Testing Automation
- Infrastructure Automation
- Integrate Security earlier into Development process

RECOMMENDED



"Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations"
By Nicole Forsgren PhD, Jez Humble, & Gene Kim

CONTINUOUS IMPROVEMENT

We are not perfect, and technology is constantly changing. Part of changing your culture is improving yourself, your team, your processes and your organization. Since failure is inevitable, it's important to set your team up to be able to absorb the failure, recover quickly and then learn from it. Most successful organizations will say that if you aren't failing every now and then, you aren't trying hard enough. With DevOps, failure isn't always a negative thing, it's a learning experience. High performing teams assume things are going to hit the fan here and there, so they build for fast detection and rapid recovery.

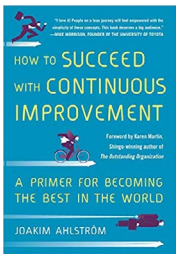
ESSENTIAL

- Continuous Improvement and failure go hand in hand
- Share knowledge from individuals and teams via such methods as Lunch 'N Learn, Centers of Excellence, internal conferences, etc. Don't have knowledge silos within the organization
- Look for waste and opportunities to improve your processes. Streamlining and automation can go a long way

ADVANCED

- Work towards having a **Just Culture** – It means that you're making an effort to balance psychological safety and accountability
- Start holding Blame Aware Postmortems as part of incident response – actively foster an environment that accepts the realities of the human brain and creates a space to acknowledge blame in a healthy way. Also referred to as Actionable Retrospectives, they collectively acknowledge the human tendency to blame, they allow for a productive form of its expression, and they constantly refocus the postmortem's attention past it. Look for waste and opportunities to improve your processes
- Continuous Improvement and the Postmortem Philosophy are core tenets of Google's Site Reliability Engineers

RECOMMENDED



"How to Succeed with Continuous Improvement: A Primer for Becoming the Best in the World"
By Joakim Ahlstrom

"Continuous improvement is better than delayed perfection"
– Mark Twain

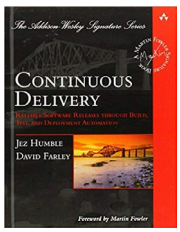
CONTINUOUS DELIVERY

Continuous delivery reduces the risk of release and leads to more reliable and resilient systems. This is achieved through automation of the build, deploy, test and release process, which reduces the cost of performing these activities, allowing us to perform them on demand rather than on a scheduled interval. This, in turn, enables effective collaboration between developers, testers, and operations.

ESSENTIAL

- Source Control not only code, but configuration management of infrastructure
- Deploy code and servers almost completely automated
- Every check in, compile, test, and validate via Continuous Integration
- Long-lived source control branches degrade quickly and cause problems
- Don't manually setup test data every time
- Security review should not be late in, or after, the development cycle
- One application's issues should not disable other apps in the system
- Teams should be empowered to introduce change and not fear of retribution when some innovative idea doesn't work out as planned
- Application monitoring efficiently and quickly in a measurable and scalable way
- Proactive monitoring of the infrastructure, platform, application and business objectives layers

RECOMMENDED



"Continuous Delivery: Reliable Software Release through Build, Test and Deployment Automation"
By Jez Humble & David Farley

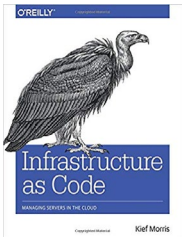
INFRASTRUCTURE AS CODE

An essential part of Continuous Delivery is being able to manage your infrastructure in an automated fashion. Organizations looking for faster deployments should treat infrastructure like software: as code that can be managed with the same processes software developers use. These let you make infrastructure changes more rapidly, safely and reliably.

ESSENTIAL

- IaC ensures that all infrastructure environments are setup the same
- Can be used on bare metal, virtual machines and on cloud infrastructure
- Many tools out there to facilitate this – Ansible, Puppet, Chef, Salt, etc

RECOMMENDED



"Infrastructure as Code"
By Kief Morris

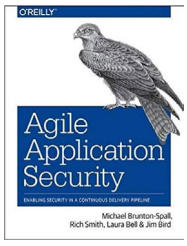
SHIFT-LEFT ON SECURITY

As we become a more connected society and our reliance on the Internet grows exponentially, the threat of security breaches only increases. We need to stop doing security reviews at the end of our development cycles and instead bring security professionals into the cross-functional teams we are building.

ESSENTIAL

- Security is everyone's responsibility
- We want to bring individuals of all abilities to a high level of proficiency in security in a short period of time
- An internal security team is not an adversary
- Developers build value for an organization; Security builds trust
- Security is a context bound job with behavioral analysis and forensics as a foundation

RECOMMENDED



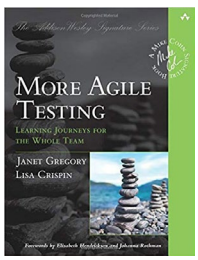
"Agile Application Security: Enabling Security in a Continuous Delivery Pipeline 1st Edition"
By Laura Bell, Michael Brunton-Spall, Rich Smith, & Jim Bird

DevOps can simply not succeed if it still requires a large number of test cases to be run manually. Since we are only human, we make mistakes no matter how hard we try. Testing automation not only encompasses running Unit Tests as part of a Continuous Integration system, but also Acceptance Tests (also called Behavior Driven Development), Integration Tests and Regression Tests.

ESSENTIAL

- Reduce errors in specifications / understanding
- Mitigate Risk by being able to repeat the tests on demand instead of scheduling resources to do that
- Computers perform repetitive tasks; people solve problems. Frees up your testers to focus on the hard problems
- Test data / fixtures can be managed and reused during automated testing

RECOMMENDED



"More Agile Testing: Learning Journeys for the Whole Team"
By Janet Gregory & Lisa Crispin

"A Reader Lives a Thousand Lives Before He Dies.
The Man Who Never Reads Lives Only One."
-George R. R. Martin

APPENDIX

Additional Resources

IN THE APPENDIX

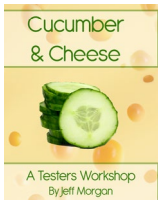
- Recommended Reading
- Tools
- Additional Downloads
- About the Author
- Acknowledgement

RECOMMENDED READING

The LeanDog team has a recommended reading list detailing which books are best for the Business Analyst, Quality Assurance, Management, Developers, Scrum Masters, and Product Owners. Each of these has helped us further our knowledge of practicing Agile in not only our software development, but our company as a whole.

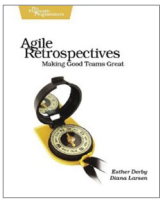
Recommended For:

Business Analyst	BA
Quality Assurance	QA
Management	MGMT
Developer	DEV
Scrum Master	SM
Product Owner	PO
User Experience	UX



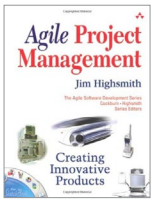
QA
DEV

"Cucumber & Cheese: A Testers Workshop"
By Jeff Morgan



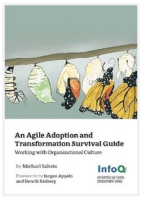
BA
PO

"Agile Retrospectives: Making Good Teams Great"
By Esther Derby & Diana Larsen



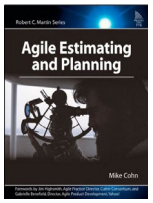
BA
MGMT
PO

"Agile Project Management: Creating Innovative Products"
By Jim Highsmith

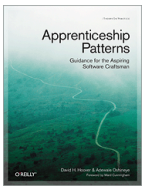


Everyone

"An Agile Adoption and Transformation Survival Guide"
By Michael Sahota



"Agile Estimating and Planning"
By Mike Cohn

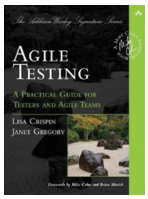


BA
QA
PO

"Apprenticeship Patterns: Guidance for the Aspiring Software Craftsman"
By Dave Hoover & Adewale Oshineye

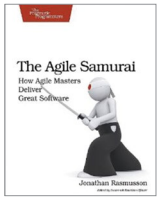
BA
QA
DEV
SM
PO
UX

"The Art of Agile Development"
By James Shore & Chromatic



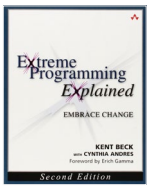
QA
DEV

"Agile Testing: A Practical Guide for Testers and Agile Teams"
By Lisa Crispin & Janet Gregory



QA
DEV
SM
PO
UX

"The Agile Samuai: How Agile Masters Deliver Great Software"
By Jonathan Rasmusson



QA
DEV

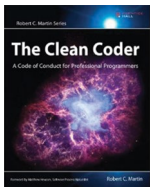
"Extreme Programming Explained: Embrace Change"
By Ken Beck, Cynthia Andres

QA
DEV

"Everyday Scripting with Ruby: for Teams, Testers, and You"
By Brian Marick

DEV

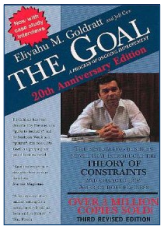
"Clean Code: A Handbook of Agile Software Craftsmanship"
By Robert C Martin



QA
DEV

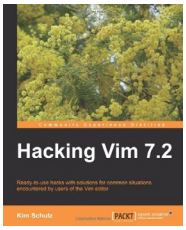
"The Clean Coder: A Code of Conduct for Professional Programmers"
By Robert C. Martin

RECOMMENDED READING



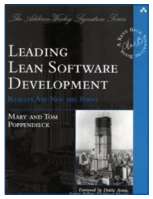
MGMT

"The Goal"
By Eiyahu M. Goldratt



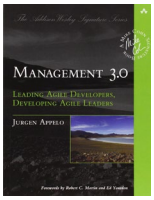
DEV

"Hacking Vim 7.2"
By Kim Schulz



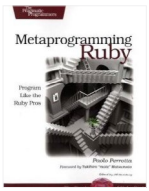
MGMT

"Leading Lean Software Development: Results are Not the Point"
By Mary Poppendieck & Tom Poppendieck



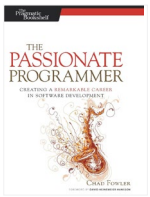
MGMT
PO
SM

"Management 3.0: Leading Agile Developers, Developing Agile Leaders"
By Jurgen Appelo



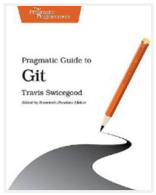
DEV

"Metaprogramming Ruby: Program Like the Ruby Pros"
By Paolo Perotta



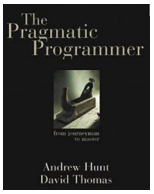
DEV

"The Passionate Programmer: Creating a Remarkable Career in Software Development"
By Chad Fowler



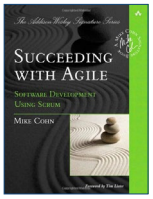
DEV

"Pragmatic Guide to Git"
By Travis Swicegood



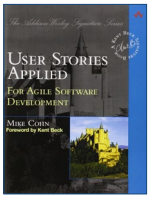
QA
DEV

"The Pragmatic Programmer: From Journeyman to Master"
By Andrew Hunt & David Thomas



BA
MGMT
DEV
SM
PO
UX

"Succeeding with Agile: Software Development using Scrum"

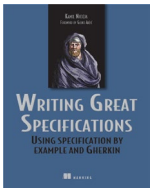


DEV

"User Stories Applied: For Agile Software Development"

BA
PO

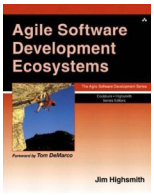
"The Software Project Managers: Bridge to Agility"
By Michele Sliger & Stacia Broderick



"Writing Great Specifications"
By K. Nicieja

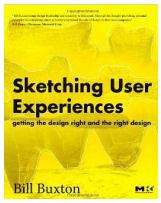
BA
PO
SM
DEV
UX

"Product Release Planning: Methods, Tools and Applications"
By Guenther Ruhe



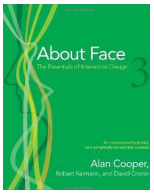
BA
DEV

"Agile Software Development Ecosystems"
By Jim Highsmith



UX

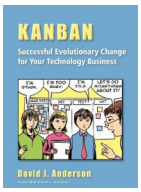
"Sketching User Experiences: Getting the Design Right and the Right Design"
By Bill Buxton



UX

"About Face 3: The Essentials of Interaction Design"
By David Cronin, Robert Relmann, Alan Cooper

RECOMMENDED READING

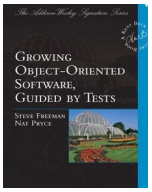


BA
DEV
SM
PO
UX

"Kanban: Successful Evolutionary Change for your Technology Business"
By David J Anderson

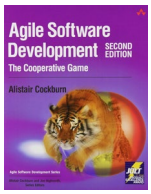
UX

"Agile Experience: A Digital Designer's Guide to Agile, Lean, and Continuous Improvement"*By Lindsay Ratcliffe & Marc McNeill*



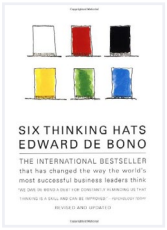
QA
DEV

"Growing Object-Oriented Software guided by tests"
By Steve Freeman



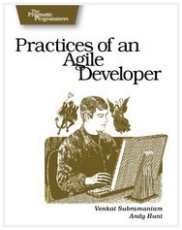
BA
DEV
SM
PO
UX

"Agile Software Development: The Cooperative Game"
By Alistair Cockburn



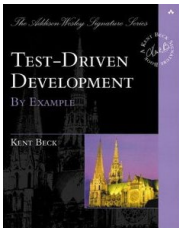
MGMT

"Six Thinking Hats"
By Edward de Bono



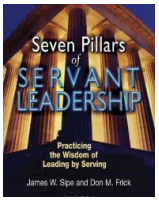
DEV

"Practices of an Agile Developer"
By Venkat Subramaniam and Andy Hunt



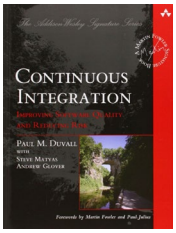
QA
DEV

"Test Driven Development: By Example"
By Kent Beck



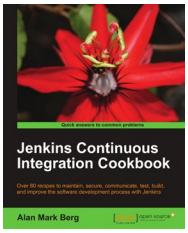
MGMT

"Seven Pillars of Servant Leadership: Practicing the Wisdom of Leading by Serving"
By James W. Sipe & Don M. Frick



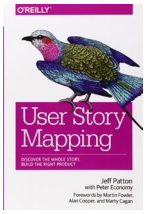
DEV

"Continuous Integration: Improving Software Quality and Reducing Risk"
By Paul M. Duvall



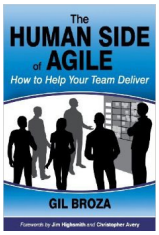
DEV

"Jenkins Continuous Integration Cookbook"
By Alan Berg



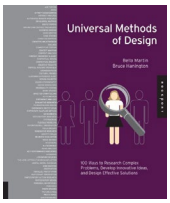
BA
QA
SM
PO
UX

"User Story Mapping: Discover the Whole Story, Build the Right Product"
By Jeff Patton



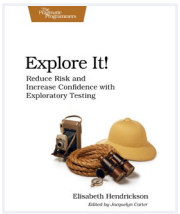
BA
MGMT
SM
PO

"The Human Side of Agile"
By Gil Broza



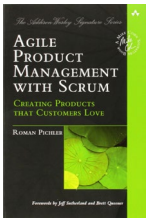
UX

"100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions"



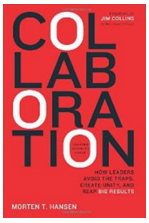
QA

"Explore It!: Reduce Risk and Increase Confidence with Exploratory Testing"
By Elisabeth Hendrickson



BA
SM
PO

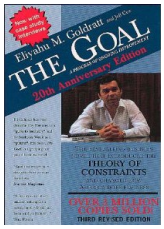
"Agile Product Management with Scrum: Creating Products that Customers Love"
By Roman Pichler



BA
MGMT
PO
UX

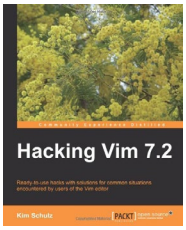
"Collaboration: How Leaders Avoid the Traps, Create Unity, and Reap Big Results"
By Morten Hansen

RECOMMENDED READING



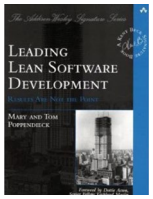
MGMT

"The Goal"
By Eiyahu M. Goldratt



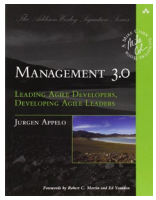
DEV

"Hacking Vim 7.2"
By Kim Schulz



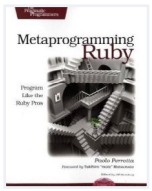
MGMT

"Leading Lean Software Development: Results are Not the Point"
By Mary Poppendieck & Tom Poppendieck



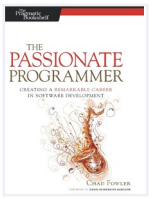
MGMT
PO
SM

"Management 3.0: Leading Agile Developers, Developing Agile Leaders"
By Jurgen Appelo



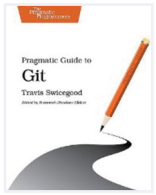
DEV

"Metaprogramming Ruby: Program Like the Ruby Pros"
By Paolo Perotta



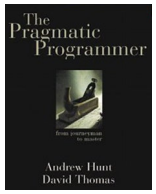
DEV

"The Passionate Programmer: Creating a Remarkable Career in Software Development"
By Chad Fowler



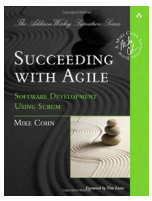
DEV

"Pragmatic Guide to Git"
By Travis Swicegood



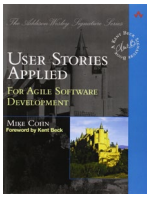
QA
DEV

"The Pragmatic Programmer: From Journeyman to Master"
By Andrew Hunt & David Thomas



BA
MGMT
DEV
SM
PO
UX

"Succeeding with Agile: Software Development using Scrum"
By Mike Cohn



DEV

"User Stories Applied: For Agile Software Development"
By Mike Cohn

BA
PO

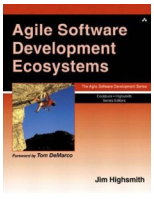
"The Software Project Managers: Bridge to Agility" *By Michele Sliger & Stacia Broderick*

DEV

"Specfication by example"
By Gojko Adzic

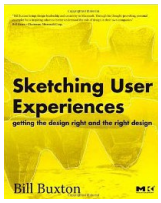
BA
PO
SM
DEV
UX

"Product Release Planning: Methods, Tools and Applications"
By Guenther Ruhe



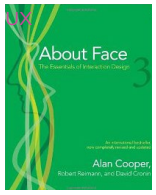
BA
DEV

"Agile Software Development Ecosystems"
By Jim Highsmith



UX

"Sketching User Experiences: Getting the Design Right and the Right Design"
By Bill Buxton



UX

"About Face 3: The Essentials of Interaction Design"
By David Cronin, Robert Relmann, Alan Cooper

RECOMMENDED READING

BA
DEV
SM
PO
UX

"Lean Software Development: An Agile Toolkit" *By Mary Poppendieck, Tom Poppendieck*

UX

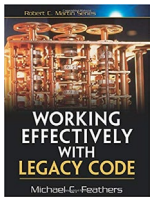
"The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses" *By Eric Ries*

QA
DEV

"Innovation by Design: How AnyOrganization Can Leverage Design Thinking to Produce Change, Drive New Ideas, and Deliver Meaningful Solutions" *By Tom Lockwood*

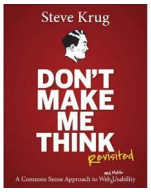
BA
DEV
SM
PO
UX

"Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation" *By Tim Brown*



MGMT

"Working Effectively with Legacy Code" *By Michael Feathers*



EV

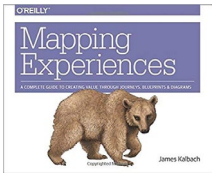
"Don't Make Me Think" *By Steve Krug*

QA
DEV

"Lean UX: Designing Great Products with Agile Teams Hardcover" *By Jeff Gothelf, Josh Seiden*

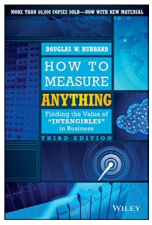
MGMT

"Mental Models: Aligning Design Strategy with Human Behavior" *By Indi Young*



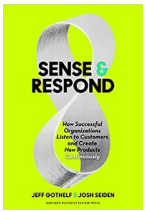
DEV

"Mapping Experiences: A Complete Guide to Creating Value through Journeys, Blueprints, and Diagrams" *By James Kalbach*



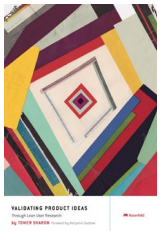
DEV

"How to Measure Anything: Finding the Value of Intangibles in Business" *By Douglas W. Hubbard*



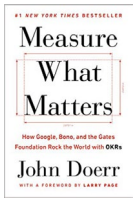
BA
QA
SM
PO
UX

"Sense and Respond: How Successful Organizations Listen to Customers and Create New Products Continuously Hardcover" *By Jeff Gothelf, JoshSeiden*



BA
MGMT
SM
PO

"Validating Product Ideas: Through Lean User Research Paperback" *By Tomer Sharon*

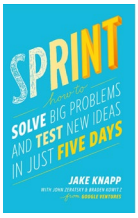


UX

"Measure What Matters: How Google, Bono, and the Gates Foundation Rock the World with OKRs" *By John Doerr*

QA

"The Beginner's Guide to OKR" *By Felipe Castro*



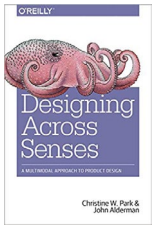
BA
SM
PO

"Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days" *By Jake Knapp, John*

BA
MGMT
PO
UX

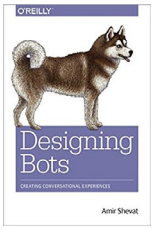
"Service Design: From Insight to Implementation" *By Andy Polaine, Ben Reason & Lavrans*

RECOMMENDED READING



MGMT

"Designing Across Senses: A Multimodal Approach to Product Design" *By Christine Park, John Alderman*

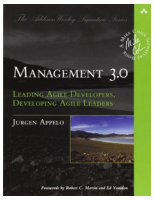


DEV

"Designing Bots: Creating Conversational Experiences" *By Amir Shevat*

MGMT

"Designing Voice User Interfaces: Principles of Conversational Experiences" *By Cathy Pearl*

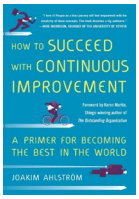


MGMT
PO
SM

"Management 3.0: Leading Agile Developers, Developing Agile Leaders" *By Jurgen Appelo*

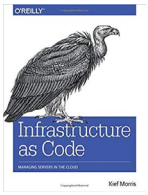
DEV

"Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations" *By Nicole Forsgren PhD, Jez Humble, & Gene Kim*



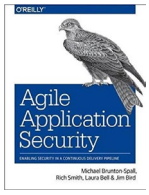
DEV

"How to Succeed with Continuous Improvement: A Primer for Becoming the Best in the World" *By Joakim Ahlstrom*



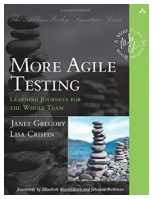
DEV

"Infrastructure as Code" By Kief Morris



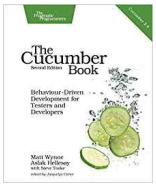
QA
DEV

"Agile Application Security: Enabling Security in a Continuous Delivery Pipeline 1st Edition" *By Laura Bell, Michael Brunton-Spall, Rich Smith, & Jim Bird*

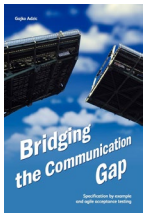


BA
MGMT
DEV
SM
PO
UX

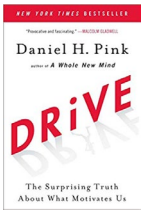
"More Agile Testing: Learning Journeys for the Whole Team" *By Janet Gregory & Lisa Crispin*



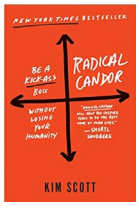
"The Cucumber Book: Behaviour-Driven Development for Testers and Developers" *By M. Wynne and A. Hellesoy*



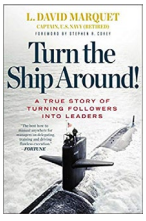
"Bridging the Communication Gap" *By G. Adzic*



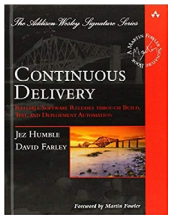
"Drive" *By Daniel H. Pink*



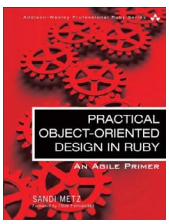
"Thinking In Systems: A Primer" *By Donella H. Meadows*



"Turn the Ship Around" *By L. David Marquet*



"Continuous Delivery: Reliable Software Release through Build, Test and Deployment Automation" *By Jez Humble & David Farley*



"Practical Object- Oriented Design: An Agile Primer Using Ruby" *By Sandi Metz*

TOOLS

PRODUCT DESIGN

- **InvisionApp** - https://www.invisionapp.com/
- **UXPin** - https://www.uxpin.com/
- **Zeplin** -https://zeplin.io/

REMOTE COLLABORATION

- **Slack** - https://slack.com/
- **Sococo** - https://www.sococo.com/
- **Zoom** - https://zoom.us/home?zcid=2478

VISUAL COLLABORATION

- **iObeya** - https://www.iobeya.com/
- **Miro** - https://miro.com/
- **Mural** - https://mural.co/

WORK/PROJECT MANAGEMENT

- **Asana** - https://asana.com/
- **Jira** - https://www.atlassian.com/software/jira
- **Pivotal Tracker** - https://www.pivotaltracker.com/
- **Trello** - https://trello.com/

ADDITIONAL DOWNLOADS

Bring best practices into your world today with these tools you can use to streamline product development and delivery.

POSTERS & SIGNS YOU CAN PRINT:

- Agile Signs & Posters
- Agile Principles Poster
- Agile Manifesto Poster
- Information Radiators & Wall Signs

AVAILABLE AT LEANDOG.COM/DOWNLOADS

DOWNLOAD OUR APP

LEANDOG AGILE TOOLS

Use our mobile collaboration cards to help you estimate, plan, and gain consensus.

CARDS INCLUDED:

- Six Thinking Hats
- Collaboration 8
- Fist to Five
- T-Shirt Sizing

ABOUT THE AUTHOR

Jon Stahl, CEO of LeanDog

Jon Stahl co-founded LeanDog in 2008. As the CEO of LeanDog, Jon maintains an active role coaching his clients on agile methodologies. He shares his knowledge by traveling around the world teaching what he has learned while energizing people and organizations. His passion is helping companies improve collaboration, communication, and adopt lean and lightweight software development practices.


LeanDog

ACKNOWLEDGEMENT

The following folks have helped bring this guide to life by contributing content, design, filming and edits throughout the print book as well as the interactive version.

- Jon Stahl
- Jeff Morgan
- Michael Norton
- Matt Barcomb
- James Grenning
- Eric Hankinson
- Angela Harms
- Shane Hayes
- Joel Helbling
- Susan Gibson
- Dan Parks
- Nicole Capuana
- Michael Lutton
- David Shah
- Matt Fousek
- Steve Jackson
- Joel Byler
- Sahithya Wintrich
- Michael Jebber
- Matt Volosin
- Pat Kelly
- Chris Nurre
- Paul Carvalho
- Keith Chirco
- Kevin Sivic
- Todd Cotton
- Brian Link
- Charles Luzar
- Amy Koran

CONNECT WITH US:

LeanDog.com • 216.236.4705 • info@leandog.com


Thank you

AGILE

Discussion Guide 4.0

WE ARE ALWAYS ADDING TO THE GUIDE

Subscribe to the email list to stay up to date.
Check out what else we have to offer at LeanDog.com/downloads.

READY TO LEARN? WE'RE READY TO TEACH

Learn the fundamentals of Agile in two half-day interactive sessions to start practicing immediately.
Find out more at LeanDog.com/training.

INTERACTIVE VERSION

Get access to tons of videos and watch our experienced coaches explain the practices featured in every chapter.

LeanDog

CONTACT US

LeanDog.com
216.236.5705