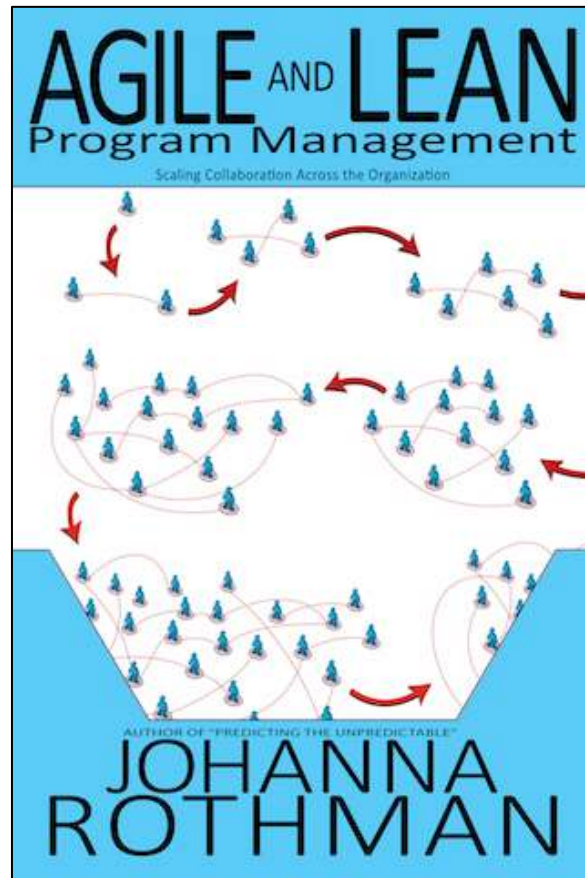


Supplemental Materials for
Agile and Lean Program Management: Scaling Collaboration Across the Organization



All contents copyright Johanna Rothman. Provided for your reading pleasure. If you want to use an image or wording, please contact me, jr at jrothman dot com.

See all my books at <http://www.jrothman.com>.

Supplemental Materials for
Agile and Lean Program Management: Scaling Collaboration Across the Organization

Figure 2.1: Cynefin Framework	3
Figure 2.2: Large Program, One Coherent Product.....	3
Figure 2.3: Inter-Related Product Program.....	3
Figure 2.4: Integrated System Product Program.....	4
Figure 2.5: What Your Core Team Might Look Like	4
Figure 2.6: What Your Software Program Team Might Look Like	5
Figure 3.1: Different Teams	5
Figure 4.1: Agile Program Charter Template	5
Figure 4.2: Program Risk Template.....	6
Figure 4.3: A Potential Agile Roadmap	6
Figure 4.4: Example of an Agile Roadmap for One Quarter.....	6
Figure 5.1: Agile Roadmap for One Quarter	7
Figure 5.2: Ranking with Business Value Points	7
Figure 6.1: Possible Kanban for a Core Team	8
Figure 6.2: Potential for Release Frequency.....	8
Figure 7.1: Possible Picture of Nine Team Small-World Network.....	8
Figure 7.2: Community of Practice	9
Figure 8.1: Possible Kanban Board for a Core Team.....	9
Figure 9.1: Program Team Estimates by Team	9
Figure 10.1: Program Feature Chart	10
Figure 10.2: Product Backlog Burnup	10
Figure 10.3: Voicemail Product Backlog Burnup, After Several Interim Releases.....	11
Figure 10.4: Program Obstacle Report.....	11
Figure 11.1: Comparison of Fixed and Growth Mindset	11
Figure 12.1: Release Frequency and the Cost of Architectural Decisions	12
Figure 14.1: One Quarter Agile Roadmap for a Robot.....	12
Figure 14.2: Possible Mechanical Engineering Kanban.....	12
Figure 14.3: Possible Silicon Kanban	13
Figure 14.4: Possible FPGA Kanban.....	13
Figure 15.1: Staggered Development and Testing	14
Figure 15.2: Implement by Feature.....	14
Figure 15.3: Curlicue Features	14
Figure 17.1: Staged Delivery Life Cycle.....	15
Figure 17.2: Design-to-Schedule Life Cycle.....	15
Figure 17.3: Release Train: Each train releases on the same relative day each quarter ..	16
Glossary.....	17
Annotated Bibliography	19
More from Johanna	22

Figure 2.1: Cynefin Framework

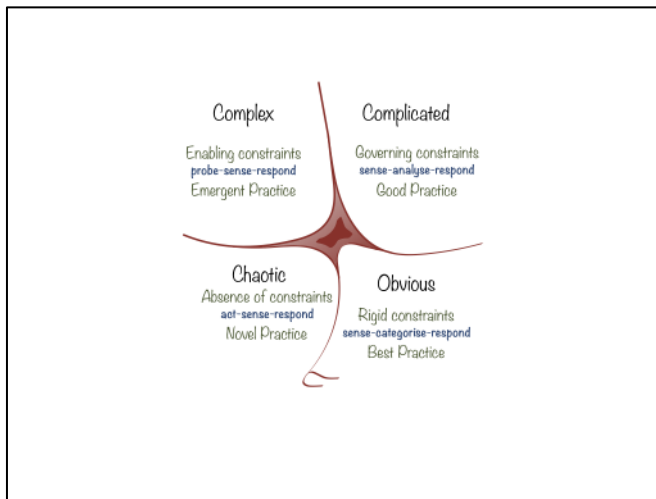


Figure 2.2: Large Program, One Coherent Product

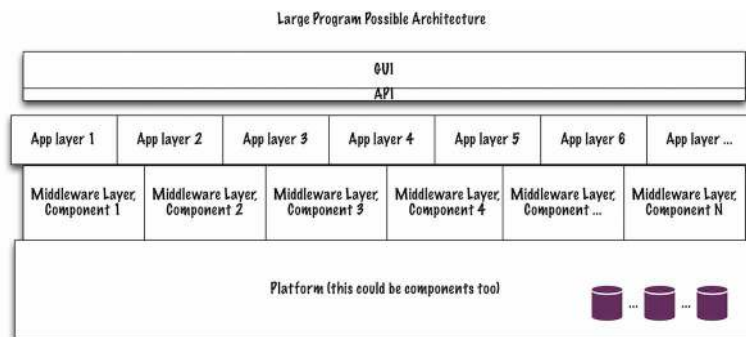


Figure 2.3: Inter-Related Product Program

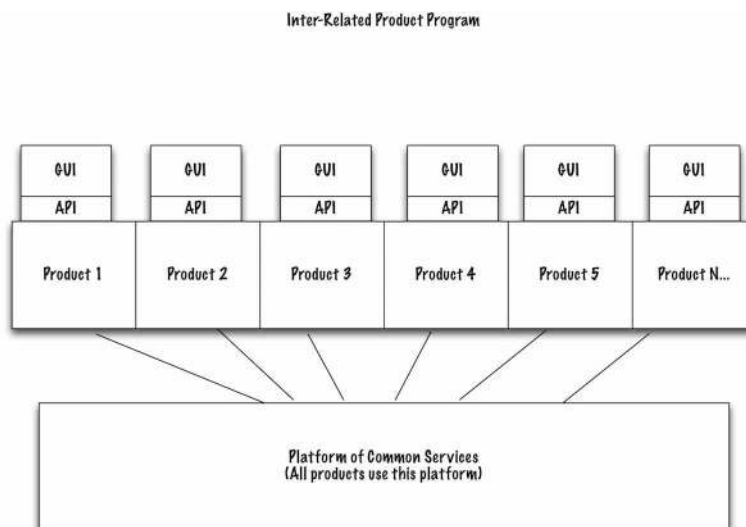


Figure 2.4: Integrated System Product Program

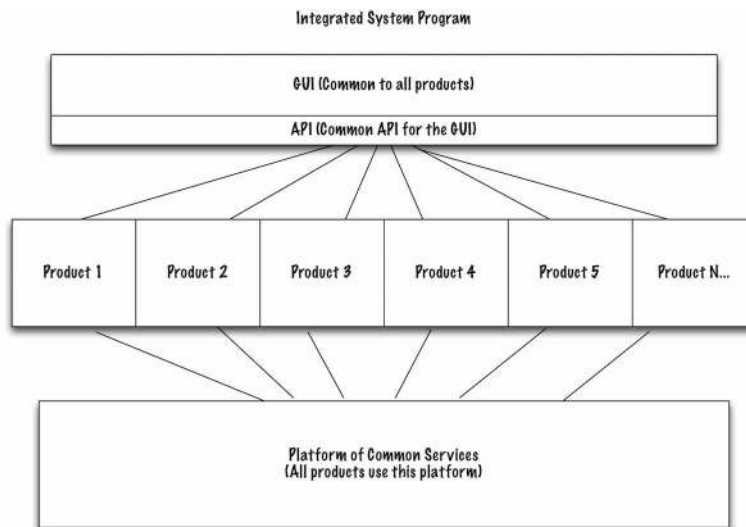


Figure 2.5: What Your Core Team Might Look Like



Figure 2.6: What Your Software Program Team Might Look Like

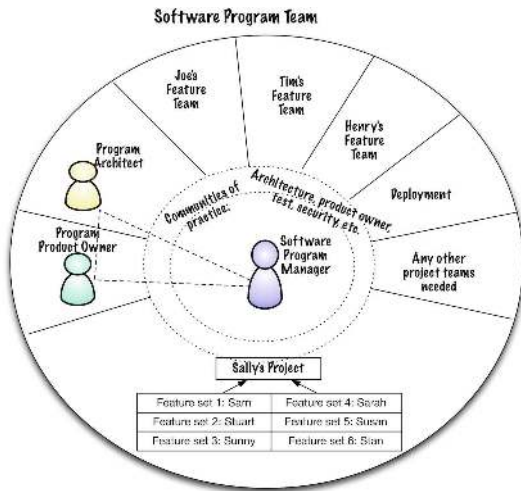


Figure 3.1: Different Teams

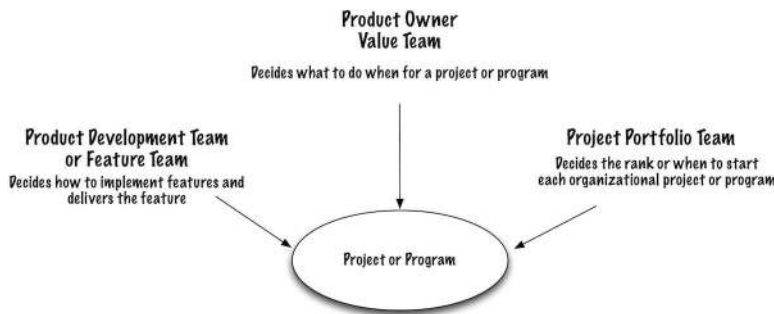


Figure 4.1: Agile Program Charter Template

- Agile Program Charter Template: Start here and make it yours**
1. Product vision: Why does the organization want the product?
 2. Release criteria or acceptance criteria: How you know the product is done.
 3. Major dates with program implications, such as demos or target release date.
 4. Product Roadmap or pointer to it
 5. Pointer to other plans: Deployment plan, Sales plan, Training plan, etc.
 6. You may need a pointer to the program risk list.
- Remember: You can always add more. But, people do not read long documents.

Figure 4.2: Program Risk Template

Numbered Risk	Risk Description	Probability	Severity	Exposure	Trigger Date	Mitigation Plan
Number each risk	Name the risk with a phrase or sentence	Probability the risk will occur	The severity if the risk occurs	Multiply the probability times severity	Date by which you need to act	Plan to deal with the risk
1	The potential problem	High, medium or low	High, medium or low	(M, H)	Make sure this date is not too late to solve the potential problem.	Say as much as you need to so your readers understand your plan.

Figure 4.3: A Potential Agile Roadmap

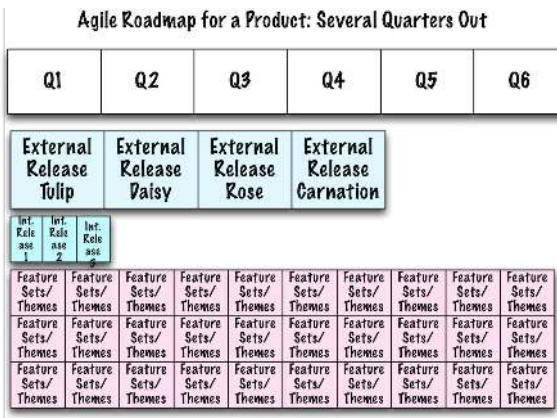


Figure 4.4: Example of an Agile Roadmap for One Quarter

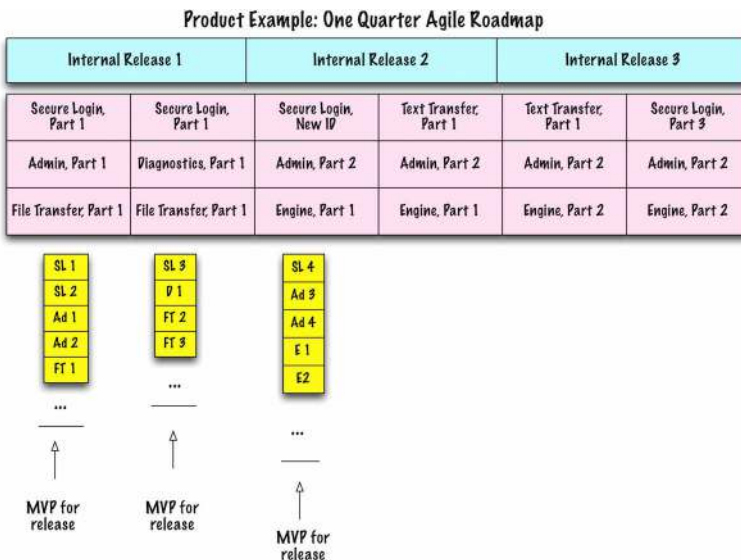


Figure 5.1: Agile Roadmap for One Quarter

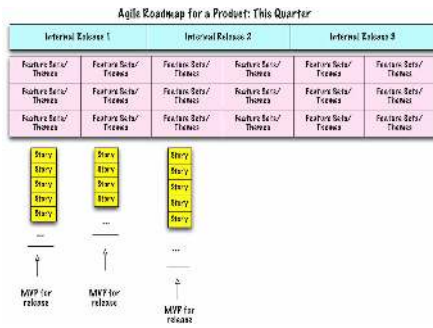


Figure 5.2: Ranking with Business Value Points

Feature	Business Value Points
Feature 1	2500
Feature 2	2000
Feature 3	1950
Feature 4	500
Feature 5	250
Feature 6	249
Feature 7	200
Total	7649

Figure 6.1: Possible Kanban for a Core Team

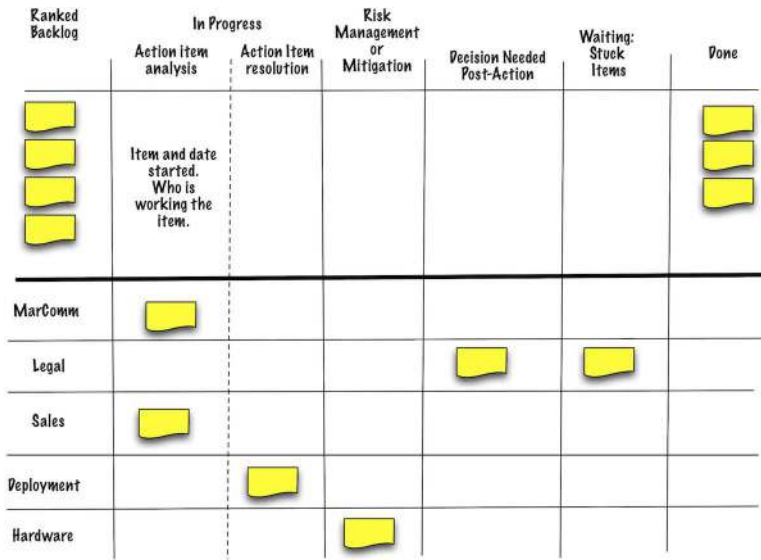


Figure 6.2: Potential for Release Frequency

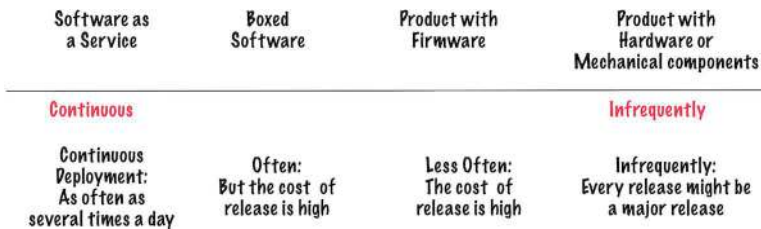


Figure 7.1: Possible Picture of Nine Team Small-World Network

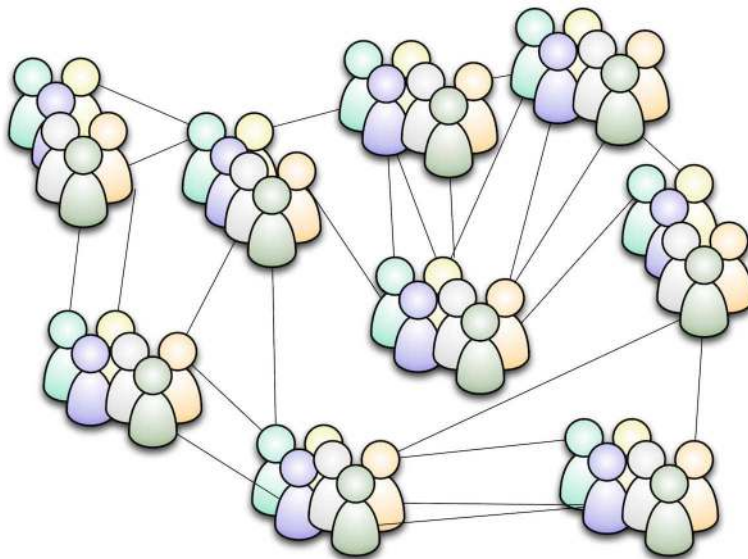


Figure 7.2: Community of Practice

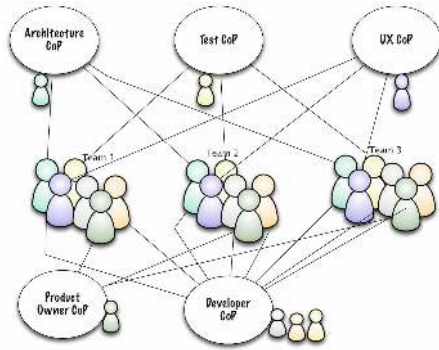


Figure 8.1: Possible Kanban Board for a Core Team

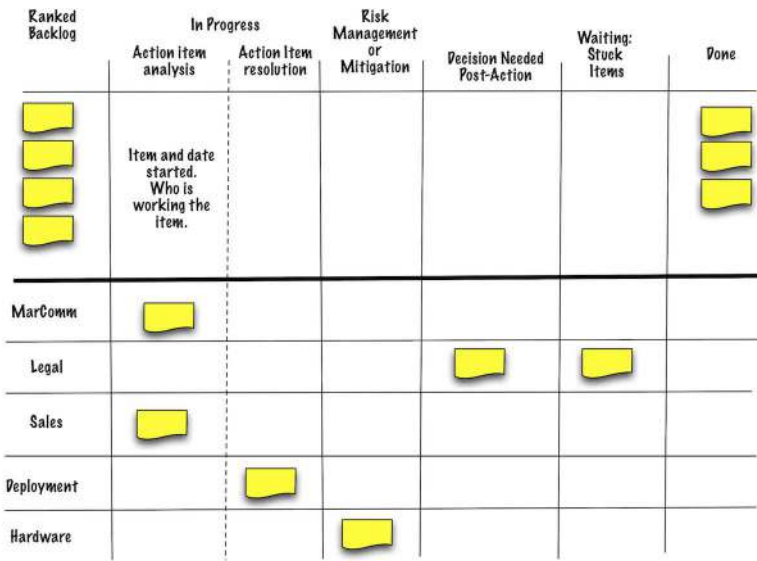


Figure 9.1: Program Team Estimates by Team

Team	Relative Estimate
Team 1	12 weeks duration, 50% confidence
Team 2	16 weeks duration, 25% confidence
Team 3	8 weeks duration, 90% confidence
Team 4	10 weeks duration, 10 % confidence
Team 5	7 weeks duration, 90% confidence

Figure 10.1: Program Feature Chart

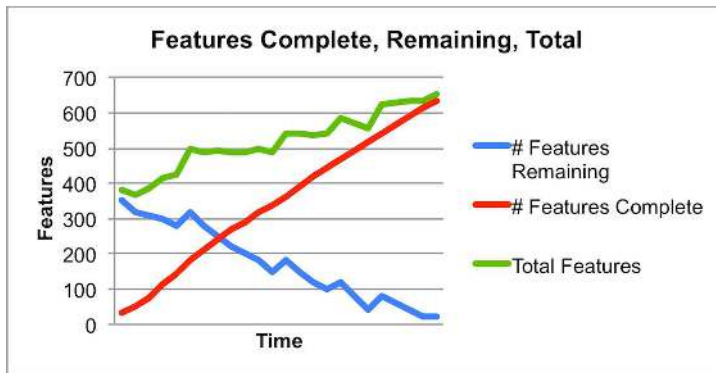


Figure 10.2: Product Backlog Burnup

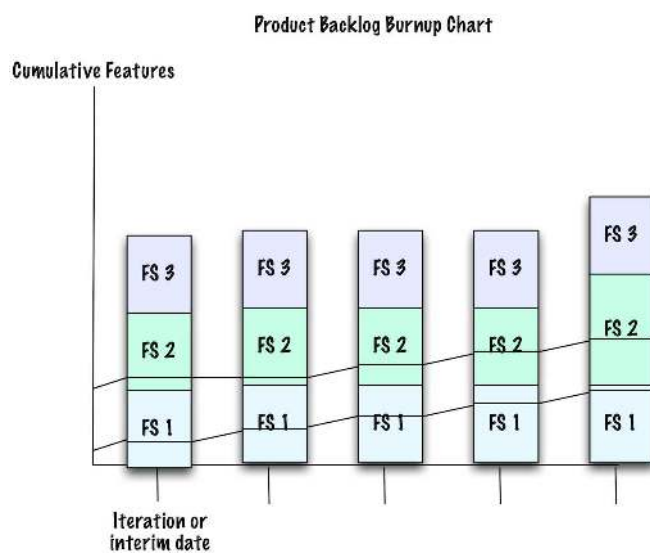


Figure 10.3: Voicemail Product Backlog Burnup, After Several Interim Releases

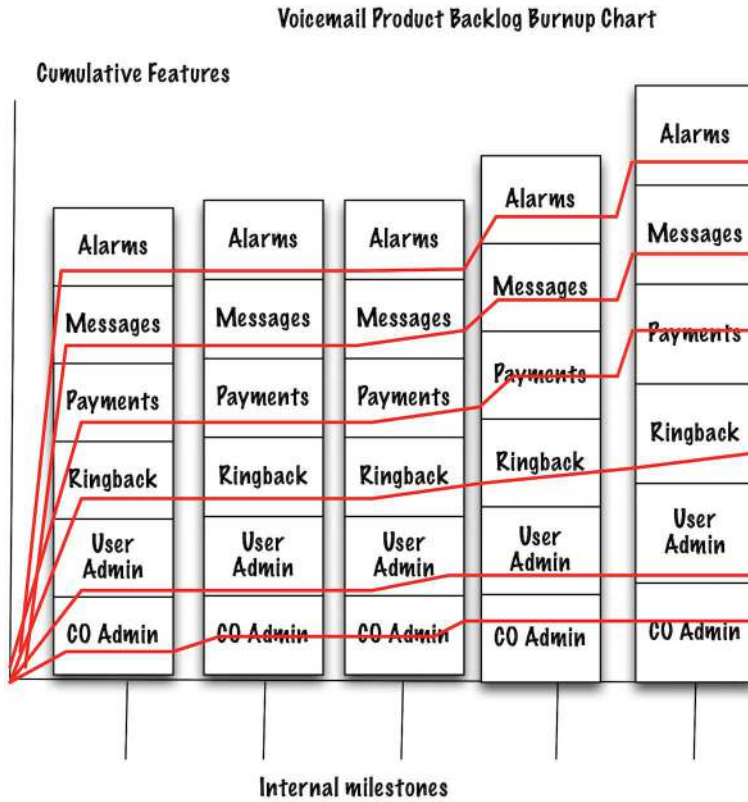


Figure 10.4: Program Obstacle Report

Obstacle Report

Rank	Obstacle	Request Date	Days Since Request Date
1	Chair for Jim	Feb 1	15
2	Need tester full time	Jan 1	44
3	Standup meeting area with whiteboard	Jan 7	38

Figure 11.1: Comparison of Fixed and Growth Mindset

Fixed Mindset	Growth Mindset
You are born with fixed skills or talents.	Skills arise from hard work. You can improve.
Avoid challenges. In the face of challenge, give up easily.	Challenges are an opportunity. Persist until you get it right.
Coast by, don't bother with effort.	Effort is essential to mastery.
Get defensive with feedback.	Learn from feedback.
With setbacks, blame others. Get discouraged.	Setbacks are something you use to try harder the next time.
Feel threatened by others' success.	Find inspiration in others' success.

Supplemental Materials for
Agile and Lean Program Management: Scaling Collaboration Across the Organization

Figure 12.1: Release Frequency and the Cost of Architectural Decisions

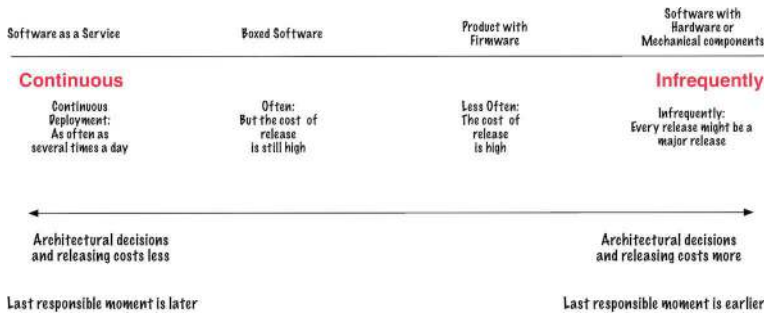


Figure 14.1: One Quarter Agile Roadmap for a Robot

Possible One Quarter Agile Roadmap for a Robot

Internal Release 1: Component Demo only		Internal Release 2: Component Demo Only		Internal Release 3: Joint Demos	
Mechanical: Joint exploration/R&D	Mechanical: Joint exploration/R&D	Joint Design, features 1-4 in emulator	Joint Design, features 5-6 in emulator	Create physical prototype in lab	Simulate OS input to current joint design
Silicon: First phase layout to check for heat and performance	Silicon: First phase layout to check for heat and performance	Silicon: Simulate Driver input, features 1-3	Silicon: Simulate Driver input, features 4-8	Silicon: Simulate Driver input, features 9-12	Silicon: Verify current features against all other work to date
OS: Interrupt and driver interactions, features 1-3	OS: Interrupt and driver interactions, features 1-3	Interrupt handler, features 1-5	Interrupt handler, features 6-10	Diagnostics basic features, 1-3	Diagnostic and Alarm integration
FPGA: Boot functions, 1-3	FPGA: Stop functions, 1-2	FPGA: Stop functions, 3-5	FPGA: Alarm, features 1-4	FPGA: Alarm, features 5-8	Integrate Alarm with Diagnostics
Program Concerns: Develop landing zones	Program Concerns: Iterate on landing zones	Program Concerns: Design review (OS, FPGA) to date	Program Concerns: Design review (Silicon & FPGA)	Program Concerns: Functional review to date (All)	Joint demo: FPGA & OS

Figure 14.2: Possible Mechanical Engineering Kanban

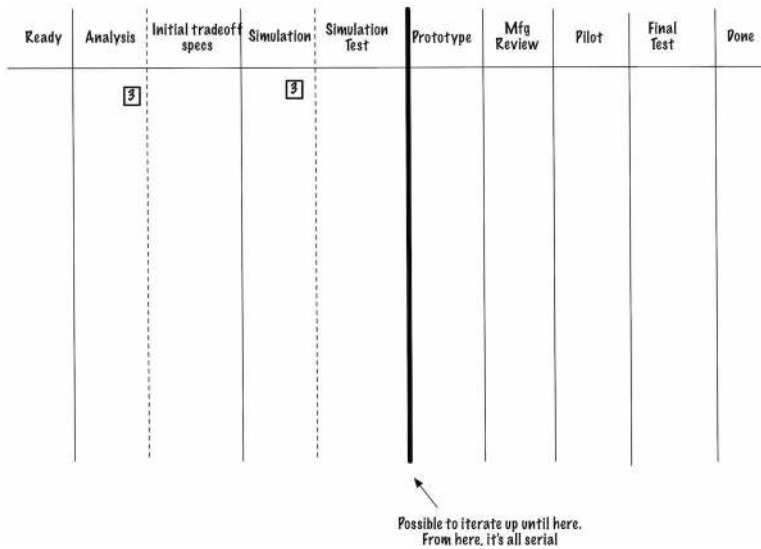


Figure 14.3: Possible Silicon Kanban

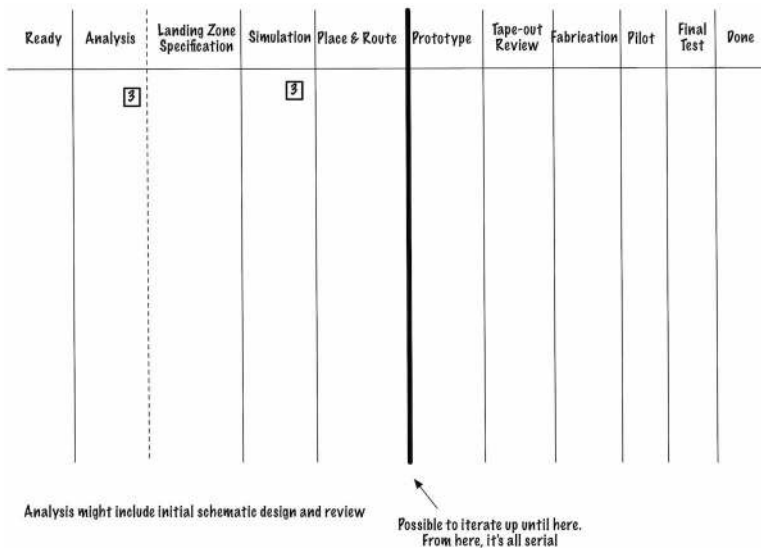


Figure 14.4: Possible FPGA Kanban

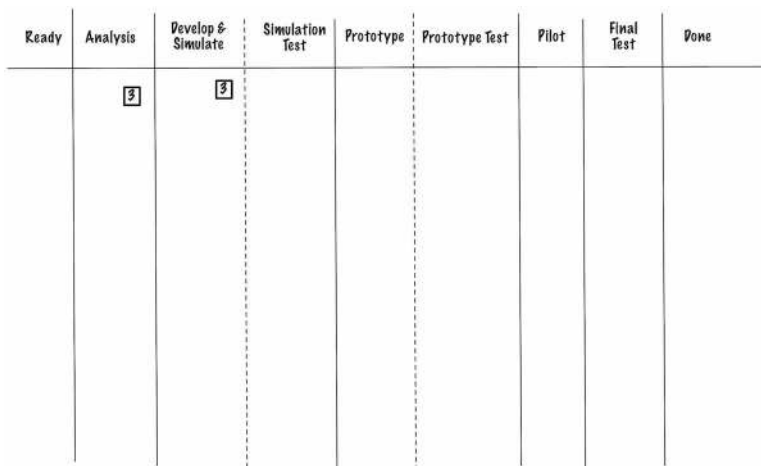


Figure 15.1: Staggered Development and Testing

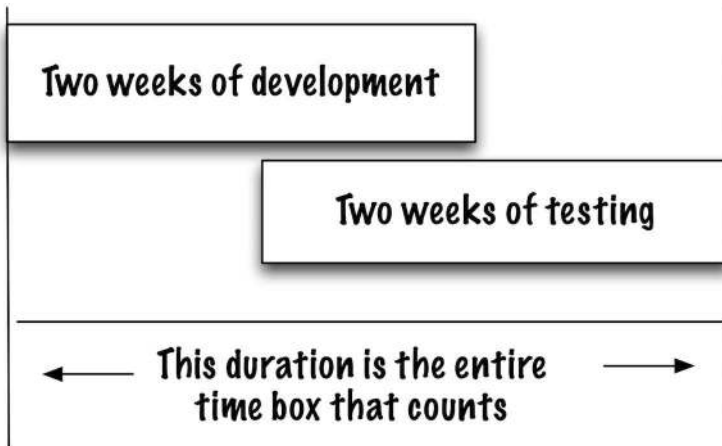


Figure 15.2: Implement by Feature

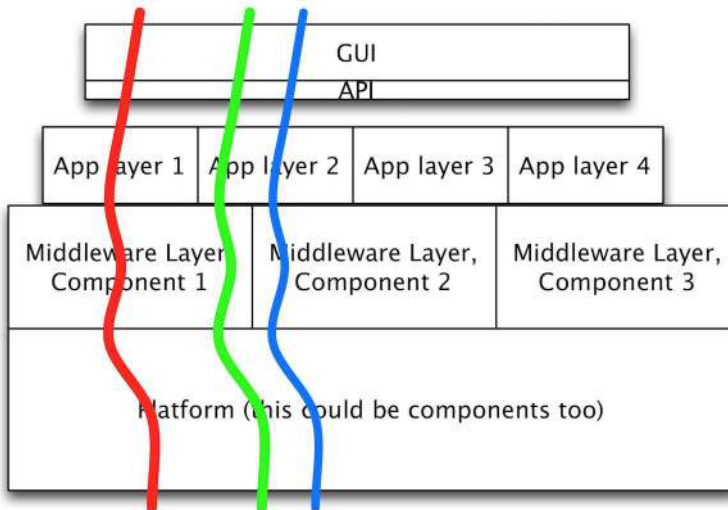


Figure 15.3: Curlicue Features

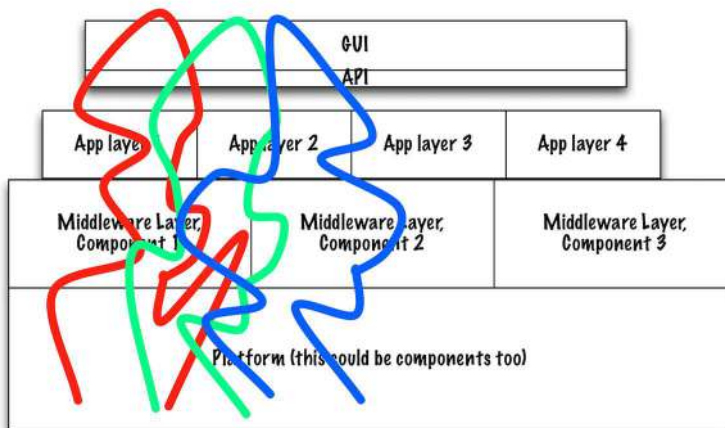
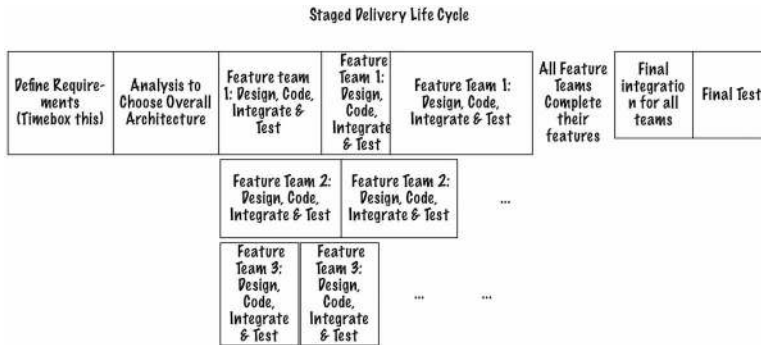
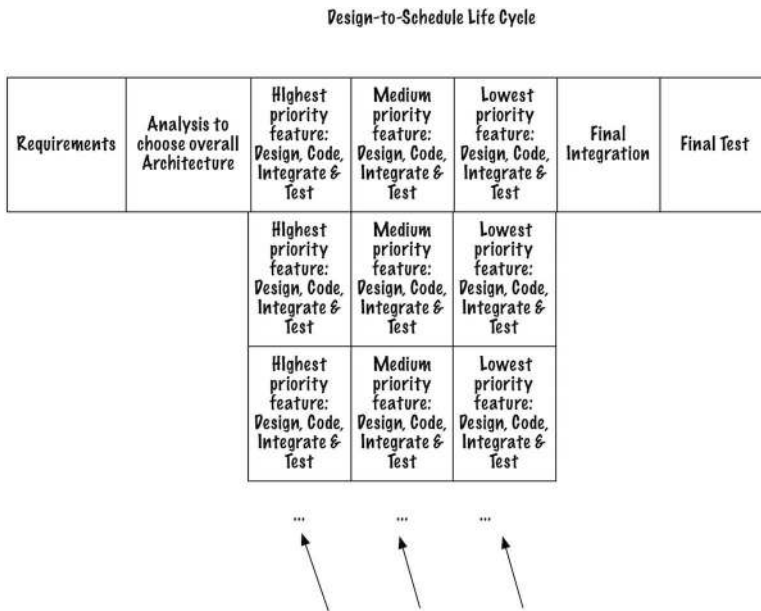


Figure 17.1: Staged Delivery Life Cycle



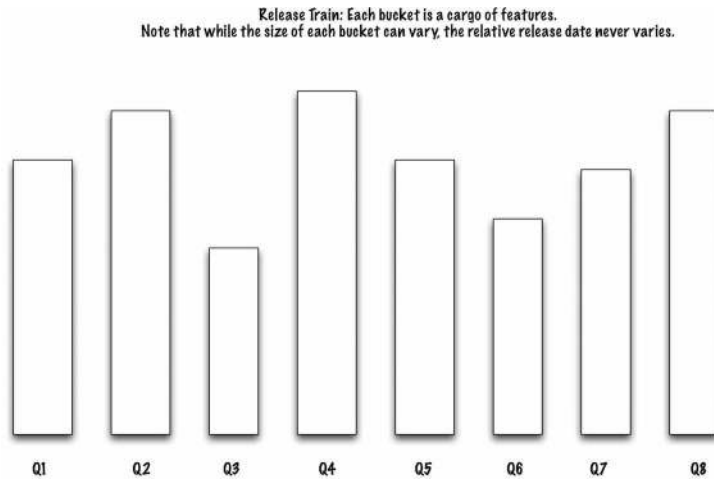
Feature teams develop their feature in each Design, Code, Integrate & Test box. The features are not all the same size. The teams manage scope, not timeboxes.

Figure 17.2: Design-to-Schedule Life Cycle



Feature teams develop their feature in each Design, Code, Integrate & Test box. The project stops when the project team runs out of time (or money).

Figure 17.3: Release Train: Each train releases on the same relative day each quarter



Glossary

If you are not familiar with the terms I've used, here are the definitions.

Adaptive: Any approach that allows you to adjust your practices or behavior to the current reality.

Agile: You work in small chunks, finishing work that is valuable to the customer in the order the customer specifies. The value of working in an agile way is that you have the ability to change quickly, because you complete work.

Backlog: Ranked list of items that need to be completed for the product.

Cost of Delay: The revenue impact you incur when you delay a project. Aside from "missing" a desired release date, you can incur Cost of Delay with multitasking, or waiting for experts, or from one team waiting for another in the program. All of these problems—and more—lead to delay of your product release.

Community of Practice: A way to share knowledge among people who belong to different teams, and share the same interests or function. For example, in a program, you might have an architecture community of practice that helps any developer learn how to evolve the design of the product. A test community of practice would provide a forum for testers to discuss what and how to test.

Flow: The team takes a limited number of items to complete, and uses the WIP limit instead of a timebox as a way to control how much work the team takes.

Generalizing Specialist: Someone who has one skill in depth, and is flexible enough to be able to work across the team to help move a feature to done.

Hardening Sprint: If a team does not complete all the work they need for a release, they may need a hardening sprint to complete all the testing for a release. This is an indication the teams are not really getting to done each iteration. They have work in progress past the end of the iteration.

Inch-pebble: Inch-pebbles are one-to-two day tasks that are either done or not done.

Iteration: A specific timebox. For agile projects, that time is normally one to four weeks. In programs, I like even smaller iterations because you want feedback more often and want to build momentum.

Kanban: Literally the Japanese word for "signboard." A scheduling system for limiting the amount of work in progress at any one time.

Lean: A pull approach to managing work that looks for waste in the system.

MVP: Minimum viable product. What is the minimum you can do, to create an acceptable product? This is not barely good enough quality. This is shippable product. However, this is minimal in terms of features.

Pairing: When two people work together on one task.

Parking Lot: This is a place to put issues you don't want to lose but don't necessarily want to address at this time.

All contents copyright Johanna Rothman, <http://www.jrothman.com>. Provided for your reading pleasure. If you want to use an image or wording, please contact me, jr at jrothman dot com.

Supplemental Materials for

Agile and Lean Program Management: Scaling Collaboration Across the Organization

Spike: If you cannot estimate a story, timebox some amount of work (preferably with the entire team) to learn about it. Then you will be able to know what to do after the day or two timebox.

Servant Leadership: An approach to managing and leading where the leader creates an environment in which people can do their best work. The leader doesn't control the work; the team does. The leader trusts the team to provide the desired results.

Sprint: An iteration in Scrum.

Swarming: When the team works together to move a feature to done, all together.

Technical Debt: Shortcuts a team takes to meet a deliverable. Teams might incur technical debt on purpose, as a tactical decision. Technical teams can have architectural, design, coding, and/or testing debt. Program teams might have risk or decision debt---the insufficiency of work for managing risks or making decisions.

Timebox: A specific amount of time in which the person will attempt to accomplish a specific task.

WIP or Work in Progress: Any work that is not complete. When you think in lean terms, it is waste in the system. Note that you do not get credit for partially completed work in agile.

Annotated Bibliography

[ADZ12] Adzic, Gojko. *Impact Mapping: Making a big impact with software products and projects*. Provoking Thoughts, 2012. Understand what you want to build.

[ADZ14] Adzic, Gojko and David Evans. *Fifty Quick Ideas to Improve Your User Stories*. Neuri Consulting LLP, 2014. Many teams struggle with user stories. This little book can help you improve what you plan to deliver.

[AMA11] Amabile, Teresa and Steven Kramer. *The Progress Principle: Using Small Wins to Ignite Joy, Engagement, and Creativity at Work*. Harvard Business Review Press, Boston, 2011. They have completed the research that says we like to finish work in small chunks so we can make progress.

[BEL06] Belshee, Arlo. *Promiscuous Pairing and Beginner's Mind: Embrace Inexperience*. IEEE Computer Society, 2005. We often think pairing has to be between similar experienced people. Not true.

[BRO14] Brodzinski, Pawel. *Minimal Indispensable Feature Set* at <<http://brodzinski.com/2014/12/minimal-indispensable-feature-set.html>>, 2014. What do you really need to build? How little can that be?

[BRO95] Brooks, Frederick P. *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition)*. Addison-Wesley, Boston, 1995. Learn from a master.

[DER06] Derby, Esther and Diana Larsen. *Agile Retrospectives: Making Good Teams Great*. Pragmatic Bookshelf, Dallas, TX and Raleigh, NC, 2006. The classic work about retrospectives.

[DWE07] Dweck, Carol. *Mindset: The New Psychology of Success*. Ballantine Books, New York, 2007. This book discusses the fixed mindset and the growth mindset. If you have the fixed mindset, you believe you can only do what you were born with. If you have the growth mindset, you believe you can acquire new skills and learn. The growth mindset allows you to improve, a little at a time.

[EDM12] Edmondson, Amy C. *Teaming: How Organizations Learn, Innovate, and Compete in the Knowledge Economy*. Jossey-Bass, San Francisco, 2012. How self-organized teams really work, and what we need to make them work in different cultures.

[FOW03] Fowler, Martin. *Who Needs an Architect?*, IEEE Software July-August 2003, pp 2-4, also at <<http://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf>>. Read this if you want to understand what architects can do for you, and what they should not do. Wonderful article about the value of architecture.

[GLI15] Gonçalves, Luis and Ben Linders. *Getting Value out of Agile Retrospectives: A Toolbox of Retrospective Exercises*. Leanpub, 2015. More retrospective exercises for your consideration.

[GRE02] Greenleaf, Robert K. *Servant Leadership: A Journey into the Nature of Legitimate Power and Greatness*, 25th Anniversary Edition. Paulist Press, New York,

All contents copyright Johanna Rothman, <http://www.jrothman.com>. Provided for your reading pleasure. If you want to use an image or wording, please contact me, jr at jrothman dot com.

Supplemental Materials for

Agile and Lean Program Management: Scaling Collaboration Across the Organization

2002. The original and definitive text on servant leadership. The forewords and afterwords provide significant value to understanding how servant leaders work.

[HAC02] Hackman, J. Richard. *Leading Teams: Setting the Stage for Great Performances*. Harvard Business Review Press, Boston, 2002. The classic work about what a team is, including what a self-managing or self-organizing team is.

[HAM14] Hammarberg, Marcus and Joakim Sundén. *Kanban in Action*. Manning, Shelter Island, NY, 2014. Terrific introduction to kanban.

[KEI08] Keith, Kent M. *The Case for Servant Leadership*. Greenleaf Center for Servant Leadership, Westfield, IN, 2008. Useful because it's short, sweet, and specific.

[MOA13] Modig, Niklas and Pär Åhlström. *This is Lean: Resolving the Efficiency Paradox*. Rheologica Publishing, 2013. Possibly the best book about how managers should consider agile and lean. A wonderful discussion of resource efficiency vs. flow efficiency.

[PAT14] Patton, Jeff. *User Story Mapping: Discover the Whole Story, Build the Right Product*. O'Reilly, Sebastopol, CA, 2014. A terrific way to explain your stories to yourself. This book will help you move from epics and themes to stories your feature teams can build.

[POP03] Poppendieck, Mary and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley, Boston, 2003. The first book to provide a lean approach to software.

[REI09] Reinertsen, Donald G. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, Redondo Beach, CA, 2009. A classic for understanding batch size and weighted shortest job first.

[BCD05] Rothman, Johanna and Esther Derby. *Behind Closed Doors: Secrets of Great Management*. Pragmatic Bookshelf, Dallas, TX and Raleigh, NC, 2005. We describe the Rule of Three and many other management approaches and techniques in here.

[ROT07] Rothman, Johanna. *Manage It! Your Guide to Modern, Pragmatic Project Management*. Pragmatic Bookshelf, Dallas, TX and Raleigh, NC, 2007. If you want to know more about how to estimate task size, establish a project rhythm, or see a project dashboard, this is the book for you. I have references about why multitasking is crazy in here.

[ROT09] Rothman, Johanna. *Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects*. Pragmatic Bookshelf, Dallas, TX and Raleigh, NC, 2009. Sometimes, program managers encounter project portfolio decisions with the feature set, or the request for people to multitask. This book helps you manage all the work in your project portfolio. I also have more references about why multitasking is crazy in here.

[RE14] Rothman, Johanna and Jutta Eckstein. *Diving for Hidden Treasures: Uncovering the Cost of Delay in Your Project Portfolio*. Practical Ink, 2014. A book about Cost of Delay and how to see how those costs affect your project portfolio.

All contents copyright Johanna Rothman, <http://www.jrothman.com>. Provided for your reading pleasure. If you want to use an image or wording, please contact me, jr at jrothman dot com.

Supplemental Materials for

Agile and Lean Program Management: Scaling Collaboration Across the Organization

[ROT15] Rothman, Johanna. *Predicting the Unpredictable: Pragmatic Approaches to Estimating Project Schedule or Cost*. Practical Ink, 2015. What you need to know about estimation and what to do when your estimate is wrong.

[SHI08] Shirky, Clay. *Here Comes Everybody: The Power of Organizing with Organizations*. Penguin Books, New York, 2008. Why collaboration works, even when people don't know each other. It's fascinating. Where I first learned the term "small-world networks."

[SIN09] Singer, David J., PhD., Captain Norbert Doerry, PhD., and Michael E. Buckley. *What is Set-Based Design?* at <<http://www.doerry.org/norbert/papers/SBDFinal.pdf>>, 2009. A readable paper about what set-based design is and how to use it.

[SNB07] Snowden, David J. and Mary E. Boone. *A Leader's Framework for Decision Making* in Harvard Business Review, November 2007. The introductory article about the Cynefin Framework.

[SH06] Subramaniam, Venkat and Andy Hunt. *Practices of an Agile Developer: Working in the Real World*. Pragmatic Bookshelf, Dallas, TX and Raleigh, NC, 2006. I first learned about the term "PowerPoint architects" from Venkat and Andy. I'd seen those kinds of architects, of course. If you want to become an agile developer, or an agile architect, start here.

[TIK14] Tikka, Ari. *Coordination Chaos* at <<http://www.slideshare.net/aritikka/coordination-chaos-41883070>>, 2014. Learn how experts can make life much more complicated.

[WIR11] Wirfs-Brock, Rebecca. *Starting with Landing Zones* at <<http://wirfs-brock.com/blog/2011/07/20/introducing-landing-zones/>>, 2011. How to trade off architectural qualities on the way to done.

[SHU14] Unknown. *Secret to Shutterstock Tech Teams* at <<http://bits.shutterstock.com/2014/05/08/the-secret-to-shutterstock-tech-teams/>> What a real manager does with real teams.

Product Manager vs. Product Owner is at <http://svpg.com/product-manager-vs-product-owner/>

Agile principles are at <http://www.agilemanifesto.org/principles.html>

Supplemental Materials for
Agile and Lean Program Management: Scaling Collaboration Across the Organization

More from Johanna

I consult, speak, and train about all aspects of managing product development. I provide frank advice for your tough problems. I'm more interested in helping you become more effective than I am in sticking with some specific approach. There's a reason my newsletter is called the "Pragmatic Manager"--that's because I am!

If you liked this book, you might also like the other books I've written:

Diving for Hidden Treasures: Uncovering the Cost of Delay in Your Project Portfolio,
<http://www.jrothman.com/books/diving-for-hidden-treasures/>

Predicting the Unpredictable: Pragmatic Approaches to Estimating Project Schedule or Cost, <http://www.jrothman.com/books/predicting-the-unpredictable-pragmatic-approaches-to-estimating-cost-or-schedule/>

Project Portfolio Tips: Twelve Ideas for Focusing on the Work You Need to Start & Finish

Manage Your Job Search, <http://www.jrothman.com/books/manage-your-job-search/>

Hiring Geeks That Fit, <http://www.jrothman.com/books/hiring-geeks-that-fit/>

Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects,
<http://www.jrothman.com/books/manage-your-project-portfolio-increase-your-capacity-and-finish-more-projects/>

Manage It!: Your Guide to Modern, Pragmatic Project Management,
<http://www.jrothman.com/books/manage-it-your-guide-to-modern-pragmatic-project-management/>

Behind Closed Doors: Secrets of Great Management,
<http://www.jrothman.com/books/behind-closed-doors-secrets-of-great-management/>

In addition, I have essays in:

Readings for Problem-Solving Leadership, <https://leanpub.com/pslreader>

Center Enter Turn Sustain: Essays on Change Artistry,
<https://leanpub.com/changeartistry>

I'd like to stay in touch with you. If you don't already subscribe, please sign up for my email newsletter, the Pragmatic Manager, <http://www.jrothman.com/pragmaticmanager>, on my website. Please do invite me to connect with you on LinkedIn, <http://www.linkedin.com/in/johannarothman>, or follow me on Twitter, @johannarothman.

I would love to know what you think of this book. If you write a review of it somewhere, please let me know. Thanks!

Johanna

All contents copyright Johanna Rothman, <http://www.jrothman.com>. Provided for your reading pleasure. If you want to use an image or wording, please contact me, jr at jrothman dot com.