

# Managing Agile Open-Source Software Projects with Microsoft Visual Studio Online

Professional



Brian Blackman, Gordon Beeming,  
Michael Fourie, and Willy-Peter Schaub

Visit us today at

[microsoftpressstore.com](http://microsoftpressstore.com)

- **Hundreds of titles available** – Books, eBooks, and online resources from industry experts
- **Free U.S. shipping**
- **eBooks in multiple formats** – Read on your computer, tablet, mobile device, or e-reader
- **Print & eBook Best Value Packs**
- **eBook Deal of the Week** – Save up to 60% on featured titles
- **Newsletter and special offers** – Be the first to hear about new releases, specials, and more
- **Register your book** – Get additional benefits



# Hear about it first.

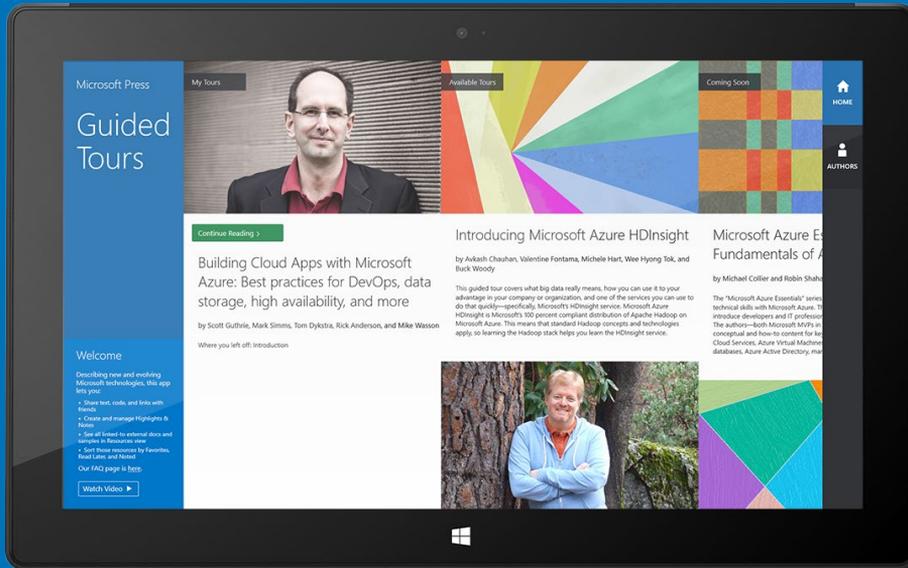


Get the latest news from Microsoft Press sent to your inbox.

- New and upcoming books
- Special offers
- Free eBooks
- How-to articles

Sign up today at [MicrosoftPressStore.com/Newsletters](https://MicrosoftPressStore.com/Newsletters)

# Wait, there's more...



## Find more great content and resources in the Microsoft Press Guided Tours app.



The [Microsoft Press Guided Tours](#) app provides insightful tours by Microsoft Press authors of new and evolving Microsoft technologies.

- Share text, code, illustrations, videos, and links with peers and friends
- Create and manage highlights and notes
- View resources and download code samples
- Tag resources as favorites or to read later
- Watch explanatory videos
- Copy complete code listings and scripts



PUBLISHED BY  
Microsoft Press  
A Division of Microsoft Corporation  
One Microsoft Way  
Redmond, Washington 98052-6399

Copyright © 2015 Microsoft Corporation

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

ISBN: 978-1-5093-0064-8

Microsoft Press books are available through booksellers and distributors worldwide. Please tell us what you think of this book at <http://aka.ms/tellpress>.

This document is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet website references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user.

This book expresses the author's views and opinions. The information contained in this book is provided without any express, statutory, or implied warranties. Neither the authors, Microsoft Corporation, nor its resellers, or distributors will be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft and the trademarks listed at <http://www.microsoft.com/about/legal/en/us/IntellectualProperty/Trademarks/EN-US.aspx> are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Acquisitions and Project Editor:** Devon Musgrave

**Editorial production:** John Pierce, Flying Squirrel Press

**Cover:** Twist Creative • Seattle

# Table of contents

Foreword.....	7
Preface.....	8
Introduction.....	9
Who should read this book.....	9
Assumptions .....	9
This book might not be for you if . . . ..	9
Organization of this book.....	9
System requirements.....	10
Downloads: Toolbox samples .....	10
We need your candid feedback.....	11
Conventions and features in this book.....	11
Errata, updates, & book support.....	11
Free ebooks from Microsoft Press.....	12
We want to hear from you .....	12
Stay in touch .....	12
About us .....	13
Authors.....	13
Brian Blackman .....	13
Gordon Beeming.....	13
Michael Fourie.....	13
Willy-Peter Schaub .....	13
Coauthors and editors.....	14
Bijan Javidi .....	14
Jeff Beehler .....	14
Patricia Wagner .....	14
Acknowledgments .....	14
Chapter 1: Triage of ideas.....	15
Flights of ideas .....	15
Roles, responsibilities, and ownership.....	15
Idea management.....	16
Capturing ideas.....	16
Triageing ideas to meet priorities, strategies, and return on investment (ROI).....	19
Identify passionate owners .....	23
Planning the kickoff to enable innovative teams .....	25
Motivation .....	25
Vision .....	26
Categorize solution.....	26
Objectives .....	27

---

Features .....	28
Roadmap .....	29
What about the orphaned ideas? .....	30
Scaling flights . . . how many are too many? .....	31
Visibility from start to finish .....	33
Dogfooding case study: Venturing into the cloud .....	37
Background information .....	38
Requirements and ownership triage .....	38
Key learnings .....	40
Chapter 2: Getting ready .....	41
Training-research-plan (TRP) .....	41
It all starts with the kickoff! .....	42
Planning the meeting .....	42
Hosting the meeting .....	43
Organizing the team .....	43
Objectives .....	44
Team structure .....	44
Portfolio "ideas" level .....	45
Solution "flights" level .....	45
Team "feature" level .....	45
Team infrastructure .....	45
Training . . . learning new things from the SMEs .....	50
Research . . . investigate and model requirements .....	51
Planning .....	51
Estimating and prioritization fundamentals .....	52
Release planning: Offline preparations .....	57
Release planning: virtualFace-to-virtualFace (vFace-to-vFace) .....	61
Schedule the infamous worldwide scrums .....	63
Summary of our process and requirement rudiments .....	63
Glimpse of tomorrow . . . tracking with an informative board .....	65
Dogfooding case study: Where is the fire? .....	66
Evidence from the field .....	66
Training-research-planning (TRP) as a remedy? .....	68
Key learnings .....	68
Chapter 3: Building the working solutions .....	69
Development (DEV) sprints .....	69
Team realizes features with stories .....	70
Running through the sprint .....	71
Pushing or pulling stories to team members? .....	71
Fixed or variable cadence? .....	72

---

Repetition! .....	72
Sprint objectives rule the deliverable .....	74
(Bi-)weekly scrum.....	74
Regular scrum of scrums .....	76
Coping to work in isolation .....	77
Metrics are another key.....	83
Key deliverables.....	84
Planning.....	85
Deliver on demand with silent preview releases .....	89
Dealing with bugs.....	90
Self-describing bugs.....	90
Triage.....	90
Resolve.....	91
Dealing with impediments.....	91
Dealing with scope creep.....	92
Dealing with critical chickens .....	92
Dogfooding case study: Triage quadrant.....	93
Background information.....	93
The quadrant triage experience .....	93
Key learnings.....	95
Chapter 4: Raising the quality bar .....	96
Quality and planning (QP) .....	96
What it is not.....	96
Hardening .....	96
Stabilize deliverables.....	97
Quality essentials .....	99
Copyediting/Reviews.....	100
Shippable release.....	101
Innovate for next time.....	105
Planning what is next.....	105
Product Owner sign off.....	105
Announcement and noisy release .....	106
Why do we ship on CodePlex and not VSO? .....	107
Dogfooding case study: vsarVersionControl and vsarTreasureMap ship ringing victory bells differently.....	107
Background information.....	107
Different strategies, same objective ... "land" .....	107
Innovate elsewhere .....	109
Keep everyone busy and informed.....	109
Make noise .....	109
Key learnings.....	110

---

Appendix A: Supporting toolbox .....	111
Triage of ideas .....	111
Gems and checklists .....	111
Getting ready .....	114
Gems and checklists .....	114
Templates: Email .....	115
Templates: Survey.....	119
General templates.....	121
Acceptance criteria (Context: Customer == Product Owner) .....	121
Definition of done (DoD) .....	121
Ship-it checklist .....	122
Work items .....	123
Building the working solutions.....	124
Gems and checklists .....	124
Raising the quality bar .....	125
Gems and checklists .....	125
Eating your own dogfood is key .....	126
Gems and checklists .....	126
Tooling .....	128
Controlled Vocabulary: Walkthrough .....	130
Appendix B: Eating your own dogfood is key.....	132
Agility is key .....	132
ALM Rangers manifesto .....	132
Microsoft Solutions Framework .....	133
Agile Manifesto.....	133
Lean.....	134
Scrum.....	134
Kanban .....	135
SAFe .....	136
Using technology wisely.....	136
Team Foundation Server.....	136
Visual Studio Online .....	136
CodePlex.....	136
Collaboration .....	137
Adapting to reality.....	138
One maintainable environment . . . simplicity rules.....	138
Personas are the glue.....	142

---

Case study: ALM Readiness Treasure Map . . . walking the plank.....	145
Background information.....	145
Team and their use of technology.....	146
Key learnings.....	146
Afterword: We are definitely not dysfunctional!.....	147
Context: "Our final sprint is blocked and potentially delayed again." .....	147
Influenced by giants.....	148
Ruck == Loose Scrum == Scrum != Dysfunctional.....	149
What's Next? .....	150
It's up to you now . . . ..	150
Continuous innovation.....	150
Further information.....	150
References.....	151

# Foreword

The ALM Rangers are a special group for several reasons. Not only are they innovative and focused on the real world, providing value-added solutions for the Visual Studio developer community, but they live and work in all four corners of the globe. The ALM Rangers are a *volunteer* organization. Talk about dedication! When we were offered the opportunity to write a foreword for this book, we knew we'd be part of something special.

The ALM Rangers don't pontificate that they've found the one true way. This is practical advice and examples for producing great software by those who've done it and—most importantly—are *still* innovating and coding. Readers will find that they have virtual coworkers who share their experiences with honesty and humor, revealing learnings and what has worked for them. This doesn't mean that this book lacks prescriptive guidance. The Rangers have embraced Visual Studio Online as their one and only home. They are evolving with the product, embracing open source software in GitHub to learn how successful OSS projects are run there and what the community values most. They've created an ecosystem that identifies the "low hanging fruit" and tracks it from idea to solution, and they never fail to recognize the Rangers and the ALM VPs who dedicate their personal time and passion to their OSS projects.

The extensive guidance shared here is not an end-to-end plan for everyone, although it could be used as a definitive guide for some teams. One of the many assets of this book is its organization into practical walkthroughs of typical ALM Ranger projects from idea to solution, presented as an easy to consume reference. Other bonuses are an appendix to quick-start your own project and reference checklists to keep you on track.

Among the authors, this book was called the "v1 dawn edition." True to their core value of "learn from and share all experiences," the ALM Rangers are always mindful that producing great software means continuous refinements from new learnings and feedback and that there will be more versions of this book. But first we invite you to immerse yourself in *Managing Agile Open-Source Software Projects with Microsoft Visual Studio Online*.

In the true spirit of Agile, ongoing innovation,

*Sam Guckenheimer*

*Clemri Steyn*

# Preface

In January 2006, I moved from Microsoft Consulting Services in Munich to Redmond and joined the Visual Studio product group. A few months later, we founded the Visual Studio ALM Rangers with the vision to fill feature and guidance gaps with community-built solutions. “Let’s solve our problems instead of complaining” was our motto, which resonated well with ALM practitioners. Today, we have shipped dozens of solutions with large numbers of downloads and top ratings.

If there is only one reason for this book, it is sharing best practices. What we have learned and shared over the past busy years has revolutionized the art of open source software for the community development.

This book is a library of best practices that we created and matured through dozens of iterations. To make it easier to digest, we have arranged these best practices in chapters. Each chapter represents the main operations of the community. These best practices cover our business end to end, from ideas to shipped products.

Every good solution starts with a good idea. Just like every other operation, we collect all ideas in a systematic and structured way. I used to call this “mathematical democracy,” where the community decides, through votes, what we work on and what is the most important thing to do next.

We are hands-on practitioners. We use our tools to develop our solutions and recommend them only if they work well for us. Typically, we extend the concept of *dogfooding* by using and testing what we create in real customer projects.

We are also about readiness. We have mastered this area to perfection by merging readiness and problem solving as one continuous and overlapping challenge. Every project starts with a readiness event. Our motto is, “If you don’t understand the space well, you can’t solve the problem.” Readiness is a prerequisite for efficient production.

When it comes to development, our famous “laser focus” meets pragmatism. No scope creep and no gold plating. We prefer to put all our energy into the quality of our solutions.

We wrote this book to share our knowledge with the community of ALM practitioners in the spirit of open source, hoping to enable others to repeat our success.

*Bijan Javidi, Principal Program Manager*

The practice of “eating your own dogfood” has a long history at Microsoft and within the Visual Studio team in particular. The notion that by using your own product to build your product and adopting the latest versions long before your customers is a powerful and compelling notion. The positive effect of the practice is so strong that it is clear to many which components have broad adoption within the team as a result of their polish, utility, and reliability. Even though the initial impact on the productivity of the dogfood “eaters” is considerable, the return on investment to both internal and external customers is significant, making the practice worthwhile.

The ALM Rangers have been longtime users of Visual Studio Online (VSO). In fact, they started using VSO before it was even called VSO. Starting in April 2011, two and a half years before the general availability of VSO, the ALM Rangers began using VSO in earnest.

While the journey has not always been smooth, the benefits of our collaboration with the ALM Rangers have been significant. Given the globally distributed nature of their team, it seems as though someone from their team is always using the service, so we regularly hear from them first when something goes wrong with the service. They have a direct connection to the engineering team that runs VSO and are not shy about providing us feedback. They have even been on the cutting edge of some service upgrades that did not go as well as planned. Their quick feedback has prevented others from experiencing outages or other errors in the service. We thank them for their dedication and contribution to the success of VSO. It is clear from the high-quality service that VSO has become that their efforts are paying off.

*Jeff Beehler*

# Introduction

With this book, we share our adventures as we hone our skills in managing solution requirements in an environment where transparency, simplicity, and trust prevail. We are hardly a typical group but instead a worst-case scenario.

We are a geographically distributed, part-time, virtual, and volunteer-based team, which implies hard-to-get commitments. We build solutions that range from small to complex and use a variety of technologies that are often at the bleeding edge. For us, being unconventional is an asset because with it comes a variety of skills, traditions, cultures, and experiences.

## Who should read this book

This book targets Agile development teams and their Scrum Masters who want to explore and learn from our “dogfooding” experiences and our continuous adaptation of software requirements management. Product Owners and other stakeholders will also find value in this book in learning how they can support their Agile development teams and by gaining an understanding of the constraints of open source community projects.

## Assumptions

This book assumes that you have at least a minimal understanding of Agile, Lean, and Scrum development concepts and are familiar with Team Foundation Server (TFS) and Visual Studio Online (VSO). To go beyond this book and expand your knowledge of Agile practices or Visual Studio technologies, [MSDN](#)<sup>1</sup> and other Microsoft Press books offer both complete introductions and comprehensive information.

## This book might not be for you if . . .

This book might not be for you if you are looking for an in-depth discussion focused on the process, development, or architecture of software requirements, tooling, or practices.

Similarly, if you are looking for source code or guidance on ALM, DevOps, or proven and official frameworks such as Agile, Scrum and Kanban, this book will not be fully relevant, and we recommend that you consider these publications instead:

- [Agile Software Engineering with Visual Studio, by Sam Guckenheimer and Neno Loje](#)<sup>2</sup>
- [Professional Scrum Development with Microsoft Visual Studio 2012, by Richard Hundhausen](#)<sup>3</sup>

## Organization of this book

We divided this book into four chapters and two appendixes, intended as easy reading to get you started and later as an invaluable reference as you, your team, and your process mature in a complex environment.

The appendixes focus on technologies, checklists, and supporting guidance. Chapters 1 through 4 take you through a practical and simple lifecycle that starts with triaging ideas and moves to getting ready, building the solution, and raising the quality bar, with a focus on requirements management, not tooling.

---

<sup>1</sup> <https://msdn.microsoft.com/en-ca/default.aspx>

<sup>2</sup> <http://amzn.com/0321685857>

<sup>3</sup> <http://amzn.com/073565798X>

**Chapter 1, “Triage of ideas”** We describe our use of Visual Studio Online (VSO) to drive the requirements, prioritization, ownership, and management concepts. Every working solution starts with a great and neatly pruned idea!

**Chapter 2, “Getting ready”** As part of training-research-planning (TRP), we gather as a team, understand the why and the what, and agree on the how. Looking back at a number of projects tackled by our geographically distributed, volunteer, and part-time teams, it is evident that this phase is pivotal.

**Chapter 3, “Building the working solutions”** We switch from planning and enter the development phase, where the rubber meets the road—executing our plan, implementing features and their associated value, and shipping working solutions quickly and often.

**Chapter 4, “Raising the quality bar”** We complete an adventure by raising the quality bar and planning the future. This is an opportunity for the team to innovate, plan, and reflect on the project and the overall process to support continuous improvement. It provides the time to complete activities often forgotten or postponed during critical development and testing phases, especially with geographically dispersed and part-time project teams.

**Appendix A, “Supporting toolbox”** A collection of templates, checklists, and gadgets that can be used for quick reference and for enabling teams to get started in a consistent way.

**Appendix B, “Eating your own dogfood is key”** A collection of proven technologies and techniques we have used 24/7 on our projects. We share our cherry-picked and tailored framework and learnings, allowing everyone to reflect, discover possible improvements, and continuously improve the way they work.

**WARNING****Technologies/TRP/QP**

- Technologies or concepts covered herein, such as training-research-planning (TRP) and quality and planning (QP), are practices we evolved to support our geographically distributed, virtual, and part-time development teams.
- Whether you adopt these technologies or concepts, make them part of the usual development sprints, or avoid them altogether is your prerogative.
- We are not mandating these concepts but simply sharing what has proven valuable to us.

**Continuous reflection, adaption, and improvements**

- We will continue to evolve the concepts covered herein. By the time you read this, we have probably made changes based on continuous process improvements and adaptations.
- We publish what we know at the time of writing, gather feedback, innovate tomorrow, and will update this content at the next possible opportunity.

## System requirements

We have no real system requirements other than an Application Lifecycle Management (ALM) system to track and manage your requirements and backlogs. While the concepts covered herein are technology agnostic, we recommend that you evaluate [Visual Studio Online](http://aka.ms/ALMRangers)<sup>4</sup> as your ALM solution.

## Downloads: Toolbox samples

You can download the templates and tools mentioned in Appendix A, “Supporting Toolbox” here: <http://aka.ms/ALMRangers/files>.

<sup>4</sup> <http://www.visualstudio.com/>

## We need your candid feedback

We appreciate and need your candid feedback to improve our content. If you notice any incorrect assertions within this book, please let us know. You can contact the authors of this book [here](#).<sup>5</sup>

## Conventions and features in this book

We use the following informational elements to highlight important information:

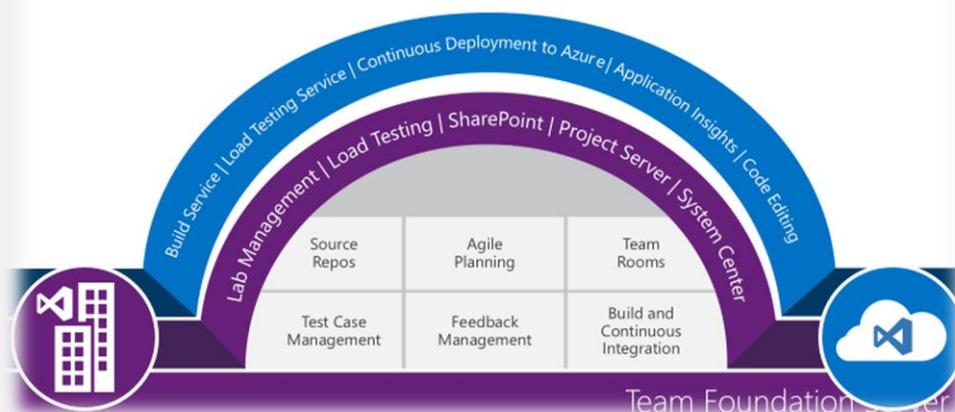
**NOTE** A general note.

**WARNING** A warning!

**GEM** A gem or nugget of information that highlights a lesson from the real world or from dogfooding experiences.

**TOOLING** A pointer to tooling documentation, mapping the theme of this book to the Team Foundation Server services and associated guidance.

### ALM your way



## Errata, updates, & book support

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

<http://aka.ms/ALMRangers/errata>

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at [mspinput@microsoft.com](mailto:mspinput@microsoft.com).

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to <http://support.microsoft.com>.

<sup>5</sup> vsarAsm4Dvpt@visualstudio.onmicrosoft.com

---

## Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at: <http://aka.ms/mspressfree>.

Check back often to see what is new!

## We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

<http://aka.ms/tellpress>

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

## Stay in touch

Let's keep the conversation going! We're on Twitter: <http://twitter.com/MicrosoftPress>

# About us

## Authors

### Brian Blackman

Brian Blackman is a principal consultant with [Microsoft Premier Developer](#), focusing on affecting the success of ISV partners and enterprises in engineering and the marketplace. He has an MBA, is a CSM, CSP, MCSD (C++), SAFe Agilist, MCTS, and a Visual Studio ALM Ranger. During his tenure at Microsoft, he worked with Windows for Automotive, Windows Mobile, Access, SQL Server, Windows Embedded, Visual Studio, and Team Foundation Server. When he is not Scrum Mastering and contributing to ALM Ranger projects, he spends his time writing code, creating and delivering workshops, and consulting in various concentrations, especially helping organizations in their quest for business agility. Brian's first authoring of technical content appeared in the first release of the MSDN library, where the article remained relevant for more than 10 years and led to him contributing to Que and Sams Publishing as a technical editor and coauthor. Brian has coauthored [MSDN Magazine](#) articles and contributed to numerous ALM Ranger publications and solutions over the last seven years.

### Gordon Beeming

Gordon Beeming is a software developer at Derivco in the sunny city of Durban, South Africa. He spends most his time hacking away at the keyboard in Visual Studio or with his family relaxing. He is a Visual Studio ALM Ranger, Visual Studio ALM MVP, and a member of the Friends of Redgate. He is also the author of *Team Foundation Server 2013 Customization*, which can be found on [packtpub.com](#). His blog is at [binary-stuff.com](#), and you can follow him on Twitter at [twitter.com/gordonbeeming](#).

### Michael Fourie

Mike started his studies in electronic engineering, where he was introduced to the wonders of computing (mostly in Smalltalk!) and has spent the last 16 years rebooting. He is an independent consultant with extensive software development experience, currently specializing in build and deployment automation to support continuous delivery. He is a technical author and has written for *MSDN Magazine* and contributed to various books and publications. He is a multiawarded Microsoft ALM MVP and Distinguished ALM Ranger. Reach him via his blog at [freetodev.com](#). You can also follow him on Twitter at [twitter.com/mikefourie](#).

### Willy-Peter Schaub

Willy started his IT career in the early 1980s during his electrical engineering studies, focusing on the BTOS/CTOS operating systems until he moved over primarily to Microsoft technologies in the early '90s. Since then, his passion has been to investigate, research, and evangelize technology and best practices, striving for simplicity and maintainability in software engineering. Apart from writing books such as *.NET Enterprise Solutions ... Best Practices*, *.NET Enterprise Solutions ... Interoperability for the Connoisseur*, and *Software Engineers on their way to Pluto*, his varied and extreme interests include scuba diving, cycling, science fiction, astronomy, and most importantly his family. He dedicates this latest adventure to [Alexander](#)<sup>6</sup> and his two brothers, Jacques and Thorsten. Keep on smiling and remember, *it is up to you*. Never, ever give up on your dreams!

---

<sup>6</sup> <http://aka.ms/alexchamp2>

## Coauthors and editors

### Bijan Javidi

Bijan is a Principal Program Manager Lead at Visual Studio. He recently moved to the Microsoft Garage team where he works with hack communities inside and outside Microsoft and Microsoft partners. He works with senior leaders across Microsoft to define hack challenges.

### Jeff Beehler

Jeff is the Program Management Director of the Visual Studio Sparc team. In this role, he coordinates many aspects of the Visual Studio effort from planning to execution to release to collecting customer feedback. As a result, he is involved with each of the teams shipping Visual Studio and "dogfoods" many parts of the system to do his job. He was on the original Visual C++ 1.0 team and finds it gratifying to be part of the effort to expand Visual Studio beyond "just" writing code and taking Visual Studio to the cloud with Visual Studio Online.

### Patricia Wagner

Patricia has been editing developer documentation at Microsoft since the days of Visual Basic 5.0. You can contact her at [p.wagner.editor@live.com](mailto:p.wagner.editor@live.com).

## Acknowledgments

To put together a book such as this has been a humbling team effort. Passionate authors, contributors, reviewers, and those who contributed intellectual property, experience, guidance, knowledge, and candid feedback inspired us and made it all possible.

Anisha Pindoria helped us align the guidance with our own process and vice versa, helped us define the key learnings of each chapter, and supported us with positive motivation.

A huge THANK YOU and COWABUNGA to Patricia Wagner for her professional, tenacious, and tireless efforts to turn our binary brain dumps into human-readable content. It has been a privilege to collaborate and walk the talk with you all on this book!

A special thanks to Devon Musgrave and John Pierce, who patiently guided us through the Microsoft Press process, doing a final polish and allowing us to ship this book.

Last, but not least, thank you to all the reviewers who patiently and pedantically worked with us to improve the content: Beth Massi, Caroline Williams, [David Starr](#),<sup>7</sup> Pete Fuenfhausen, Rob Maher, and [Soung Bae](#).<sup>8</sup>

Last but not least, a thank you to Clemri Steyn for sponsoring this book and to all of our ALM Rangers for their phenomenal passion, commitment, and investment of precious family time for the ALM community.

You all "rock"!

---

<sup>7</sup> <http://aka.ms/U071nc>

<sup>8</sup> <http://aka.ms/b6h2vm>

# Chapter 1: Triage of ideas

*It is wise to persuade people to do things and make them think it was their own idea.*

—Nelson Mandela

With more than 200 subject matter experts collaborating on at least five projects simultaneously, we could regard ourselves as a small enterprise in need of portfolio management. However, two of our core values motivate us: to *favor simplicity and low tech over complexity* and take on *regular dogfooding of Visual Studio Application Lifecycle Management tools*. We base the requirements, ownership, and management concepts introduced in this book on Visual Studio Online (VSO) features and the releases driven by stakeholder ideas that we all triage (Figure 1). Remember, simplicity rules!

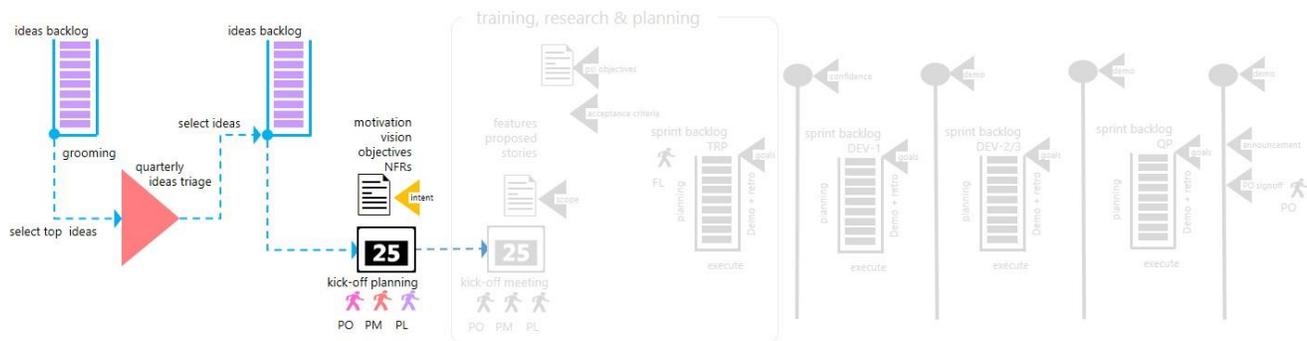


Figure 1. Lifecycle: Requirements and ownership triage.

## Flights of ideas

We represent an idea in Visual Studio Online by an Epic work-item type (WIT) decorated with one or more tags to align with Idea Source (where we found the idea), Themes, and Initiatives.

### TOOLING

[Agile Portfolio Management: Using TFS to support backlogs across multiple teams](#)<sup>9</sup> (Boer & Ferrell, 2013).

### GEM

#### What is a great idea?

Just as there are no bad questions, there are no bad ideas. Ideas that might seem wild to some are to others an indicator of thinking “out of the box” and innovation. An idea should be bold, out of this world, and should offer differentiating innovation.

From a high-altitude view, we visualize our projects as flights, which is a well-understood metaphor. Each flight is time boxed, has one or more teams working together when the project is in flight, and delivers a solution when it lands.

## Roles, responsibilities, and ownership

We introduce common personas and their responsibilities in Appendix B, “Eating your own dogfood is key,” in the section “[Personas are the glue.](#)” The titles and responsibilities of some personas may overlap with other personas. This varies from organization to organization.

<sup>9</sup> <http://msdn.microsoft.com/en-us/library/dn798712.aspx>

We need a number of personas to drive the overall vision, own and maintain the backlogs, and coordinate with all stakeholders during the requirements and ownership triage.

Table 1 lists two personas and summarizes their roles, responsibilities, and ownership during the triage phase:

Persona	Responsibility	Ownership
Program Manager	Works with the teams, triage group, and leadership to elaborate ideas, participate in planning reviews, coordinate, and drive the ideas triage.  Maintains the operational environment and the ideas backlog, produces a report of ideas for the triage group, and helps launch new flights.	<ul style="list-style-type: none"> <li>• Triage group</li> <li>• Triage</li> <li>• VSO environment</li> <li>• Ideas gathering</li> <li>• Pre-triage</li> </ul>
Scrum Master	Mentors the teams, enforces the framework guidelines, eliminates impediments raised during the triage phase, and actively leads the continuous improvement and training efforts.	<ul style="list-style-type: none"> <li>• Process</li> <li>• Framework</li> <li>• Scrum Masters</li> </ul>

Table 1. Roles, responsibilities, and ownership during the triage phase.

One or both of these personas will work with other stakeholders in a triage group, which has the ultimate responsibility for selecting and coordinating the next wave of initiatives.

## Idea management

We work with ideas, triage them, and turn them into solutions. The ALM community and stakeholders submit great ideas (commonly referred to as initiatives, investment themes, or product themes).

It often helps to visualize solutions as the beginning of something new (dawn), in full operation (daylight), and during the period in which the value of a solution has faded (dusk) to highlight the risk and value. Ideas need to be harvested, groomed, and prioritized to allow the triage group to understand their value proposition and decide how to invest in new (dawn), future, existing (daylight), or retiring (dusk) solutions.

For example, investing in a solution that is in the dusk phase is potentially a high-risk and short-lived investment. Remember the saying “the darkest hour is just before the dawn” (Thomas Fuller, 1650) when you’re faced with the challenges of embarking on a new project.

## Capturing ideas

Unlike farmers, we have no specific season for harvesting ideas. We collect ideas—no matter how small, big, or out-of-this-world—from a number of sources throughout the year (see Figure 2). Our primary collection channels are [Visual Studio UserVoice](http://visualstudio.uservoice.com/forums/121579-visual-studio),<sup>10</sup> events, forums, email, and face-to-face discussions with the product group, ALM Rangers, and the ALM community.

<sup>10</sup> <http://visualstudio.uservoice.com/forums/121579-visual-studio>

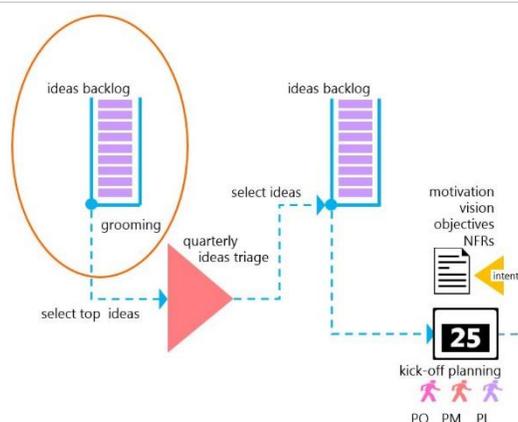


Figure 2. Collecting new ideas.

**TOOLING** [Agile Software Engineering with Visual Studio, by Sam Guckenheimer and Neno Loje<sup>11</sup>](#)

We capture all ideas that are in harmony with our mission statement in an ideas backlog.

**NOTE** **Our mission:**  
The Visual Studio ALM Rangers provide professional guidance, practical experience, and gap-filling solutions to the ALM community.

We capture each idea using the TFS work item title and use supporting information as the description. If sourced from UserVoice we use the number of votes to drive the business value. We use priority to define perceived importance compared with the known product and marketing strategies.

**GEM** **Avoid associating business value with monetary value!**  
Using the Fibonacci sequence for priority avoids the attempt to associate business value with monetary value. Alternatively, you can use any numerical value with a higher value to note greater business value or impact. See [Measuring the Business Value of a PBI<sup>12</sup>](#) for a good starting point if you are unsure what to put in the business value field.

The information captured at this point is “raw” because, apart from minor editing, we capture the originators’ idea and context as is. We do, however, encourage idea submissions to define at least the what, why, and when to minimize misunderstanding and misinterpretation.

We identify ideas with two tags. The *type* tag categorizes the idea as a business, architecture, or process idea (see Table 2). The *source* tag identifies where the idea originated.

**NOTE** At the time of this writing, we were dogfooding the use of tags and a custom Requirements Type work-item field. The recommended way is to use a custom field, but if you are unable to customize your process, template tags are an alternative.

Category	Values	Description
Type	Architecture	Ideas focused on architectural or infrastructure changes that typically are reusable and evolve.
	Business	Ideas focused on business functionality or user experience. This is our default type.

<sup>11</sup> <http://amzn.com/0321685857>

<sup>12</sup> <http://blog.accentient.com/measuring-the-business-value-of-a-pbi/>

Category	Values	Description
	Process	Ideas focused on our Scrum-based process.
Source	Community idea	Ideas sourced from communities, typically during events.
	Research Idea	Ideas sourced from stakeholders to research or future-proof technologies.
	Strategic idea	Ideas sourced from leadership, marketing, or product groups.
	UserVoice idea	Ideas sourced from the UserVoice Forum.

Table 2. Idea type tags.

## TOOLING

[Scaled Agile Framework: Using TFS to support epics, release trains, and multiple backlogs](#)<sup>13</sup>

As shown in Figure 3, the new idea joins other new ideas or ideas in progress on the backlog. If we plan to consider the idea in the upcoming idea triage, the iteration path is set to *team project name*\Triage.

Enhance ALM Readiness Treasure Map to address v2 backlog and deliver a v3

Iteration: VisualStudio.ALM\Triage

**STATUS**

Assigned To: Willy-Peter Schaub  
 State: New  
 Reason: Moved to the backlog

**DETAILS**

Priority: 1  
 Business Value: 4  
 Target Date: 2014-06-30 12:00:00 AM  
 Area: VisualStudio.ALM\Strategy Team\Research Idea

**DESCRIPTION** IMPLEMENTATION (6)

This feature is intended as a container for all v2.1 bugs and future feature enhancements which will result in a v3 update. This continues as an ALM

**ACCEPTANCE CRITERIA** HISTORY LINKS (8) ATTACHMENTS

Ideas 40 work items (23 top level)

ID	Priority	Busin...	Work Item...	Area Path	Title	Assigned To	State	I... Tags
8941	1	999	Feature	VisualStudio.ALM\vsarSAFE	TFS on Azure (IAAS) Planning and Ops Guide	Mario Rod...	In Progress	Business Strategic Idea
9359	1	999	Feature	VisualStudio.ALM\vsarSAFE	Project Flight Plan (Status) Board	Willy-Peter...	In Progress	Business Research Idea
9543	1	997	Feature	VisualStudio.ALM\vsarSAFE	SAFE Awareness and Readiness	Gregg Boer	In Progress	Business Research Idea
9469	1	995	Feature	VisualStudio.ALM\vsarDevOps\Guidance	ALM Rangers DevOps / Building a Release Pipeline with TFS 2013	Roopesh...	In Progress	Business Strategic Idea
9467	1	888	Feature	VisualStudio.ALM\vsarVersionControl	Version Control Guidance Upgrade v3 (Branching, TFVC and NuGet)	Matthew...	In Progress	Business Strategic Idea
9468	1	888	Feature	VisualStudio.ALM\vsarVmFactory	On-Time VM Factory Service	Bijan Javid	In Progress	Architecture Research Idea
9649	1	888	Feature	VisualStudio.ALM\vsarVersionControl	Version Control Guidance Upgrade v3 (Git for TFVC User)	Matthew...	In Progress	Business Strategic Idea
9115	1	28	Feature	VisualStudio.ALM\vsarSecurity	Extract effective permission from TFS for audit purposes	Mario Rod...	In Progress	Business UserVoice Idea
8425	1	4	Feature	VisualStudio.ALM	Enhance ALM Readiness Treasure Map to address v2 backlog and delive...	Willy-Pet...	New	Business Research Idea

Figure 3. An idea captured in the ideas backlog.

## NOTE

At the time of this writing, we were dogfooding the use of tags and the Feature work-item type to represent an Epic.

We evolve the idea, represented by an Epic, and the associated work-item hierarchy as we progress through the triage and subsequent milestones. Base information includes a meaningful title, a concise description, priority, business value (as described earlier), and an iteration set to Triage by default (Figure 4).

<sup>13</sup> <https://msdn.microsoft.com/en-us/library/dn798712.aspx>



Figure 4. Idea represented by an Epic before the triage.

## Triaging ideas to meet priorities, strategies, and return on investment (ROI)

We actively manage the ideas backlog to keep it from stagnating. Our default triage cadence is quarterly, aligned with the organizational financial year and introducing a predictable planning rhythm. As shown in Figure 5, we initiate a triage in January, April, July, and October. If a new priority or critical idea emerges, we trigger an ad hoc triage.



Figure 5. Quarterly triage cadence.

### GEM

#### Quarterly [ + Rolling ]

In addition to the quarterly triage rhythm, you can perform rolling triages when an in-flight project enters its last sprint. This does not mean that we start a new project every time we finish a project, but enables pro-active planning for the next project to start.

## Pre-triage . . . warming up

In the pre-triage phase, we review ideas that have landed (released) or are in flight (under development). For new ideas, we produce a list of service-mode candidates, an ordered list of new candidates, and a report that we submit to the triage group for consideration (Figure 6).

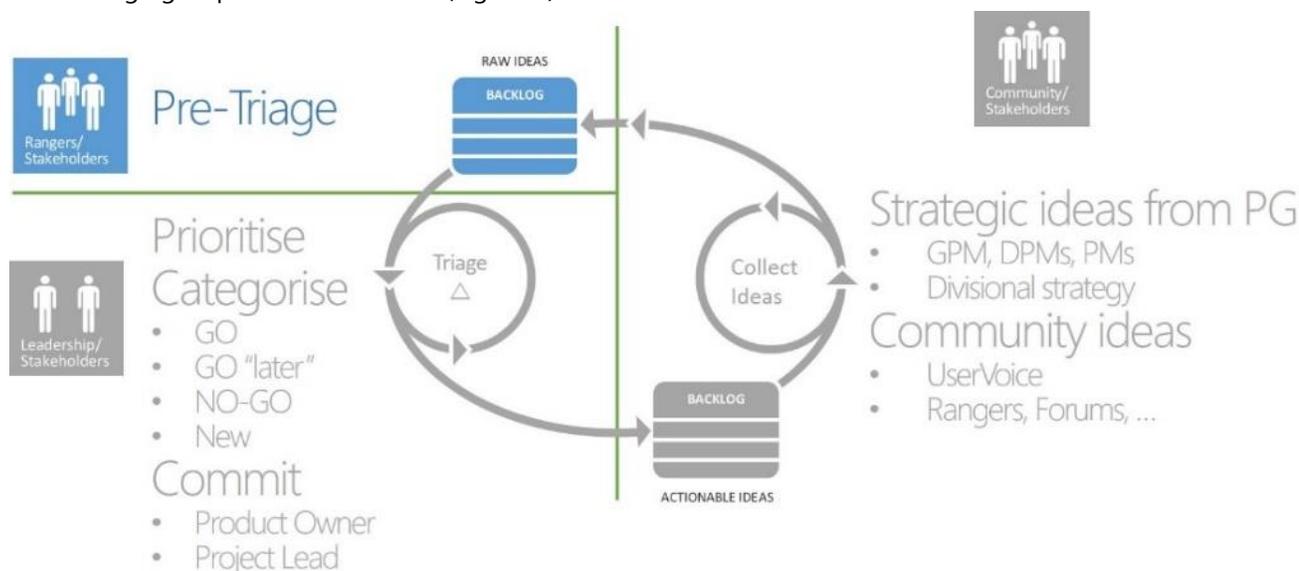


Figure 6. Pre-triage process.

## Service-mode candidates

Before committing to new ideas, we evaluate our previously shipped assets to decide which are ready for service mode and which should be retired. We do this because our bandwidth is limited and severely impacted by long-term maintenance.

### GEM

#### Top service-mode candidates

Good service-mode candidates are those solutions with declining business value, that require maintenance investment, and are misaligned from the main product strategy. Do not invest invaluable resources in solutions that no longer add little or no value.

When a solution enters service mode, we continue to monitor forums and discussions to maintain a vibrant community. However, we consider only critical bug-fix maintenance and no new features. This ensures that we are able to invest our limited and precious bandwidth in new initiatives.

## New candidates

Based on priority and business value, new ideas already appear in a rough order of importance. To display them in a visible ranking diagram, we use these values to calculate a *ranking score*.

### NOTE

The calculation of the ranking score is going to vary from environment to environment. We usually use the following:

$$\text{Ranking score} = ( ( 5 - \text{Priority} * 1000 ) + \text{business value} ) / \text{size}$$

Priority is defined in TFS process templates as 1 through 4 by default, with 1 representing the highest priority.

We calculate the business value based on community votes (interest) and the actual number of votes received for the idea by the community; we use a constant (900 to 999) for strategic ideas. We prefer to use [Fibonacci numbers](#)<sup>14</sup> (Wikipedia, Fibonacci number, 2015), assigning 1 for <10 votes, 2 for <20, 3 for <30, 5 for <40, 8 for <50, and 13 for 50+.

The template defines *size* with the values 1 through 5, representing an eXtra Small, Small, Medium, Large, and eXtra Large problem domain to resolve. Size corresponds to the length of time or number of sprints in the development queue.

### GEMS

#### (Fast) value delivery

Value decays while dissatisfaction grows over time. Ideas that are quickly developed or can be broken down into smaller shippable increments should be favored over ideas that will decay over time.

For more information, see [shortest job next \(SJN\) and shortest job first \(SJF\)](#)<sup>15</sup> (Wikipedia, Fibonacci number, 2015).

## Prepare a triage crisp presentation

We distribute the triage report together with the triage invitation a week before the online teleconference meeting to give everyone an opportunity to review and contemplate the ideas. It is highly likely that stakeholders who are physically colocated will dominate the meeting. We recommend that you include the following information in a triage slide deck (Figure 7 shows an example):

- **Project highlights**

A summary of recent project activities and highlights to provide context for past and recent success stories as well as failures that are relevant to the triage.

- **Recommendations—Service Mode**

A summary of the service-mode analysis that shows the ranking and recommended candidates

<sup>14</sup> [http://en.wikipedia.org/wiki/Fibonacci\\_number](http://en.wikipedia.org/wiki/Fibonacci_number)

<sup>15</sup> [http://en.wikipedia.org/wiki/Shortest\\_Job\\_First](http://en.wikipedia.org/wiki/Shortest_Job_First)

**Recommendations—Next Wave**

A summary of the recommended ideas to provoke discussion and prioritization of the top ideas

**Supporting information—Ideas**

For each recommended idea, include a slide that summarizes the motivation, deliverables, origin, proposed leads, and known constraints.

Figure 7. Triage: Idea context.

Triage calibration

While we conduct most of our meetings and workshops as teleconferences due to our geographically distributed and part-time volunteer resources, the triage is a meeting that we must do face-to-face, with as many triage group members as possible. This ensures that there are no misunderstandings and no missed emotional facial reactions, and it minimizes the impact on senior leadership. We need a strong facilitator who understands the context and the process and is also trusted and respected by the stakeholders. This event is outlined in Figure 8 and Figure 9.

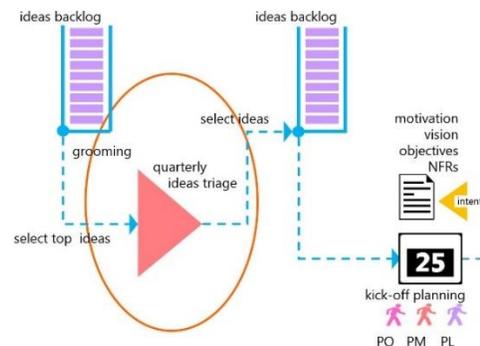


Figure 8. Triage: Quarterly ideas triage.

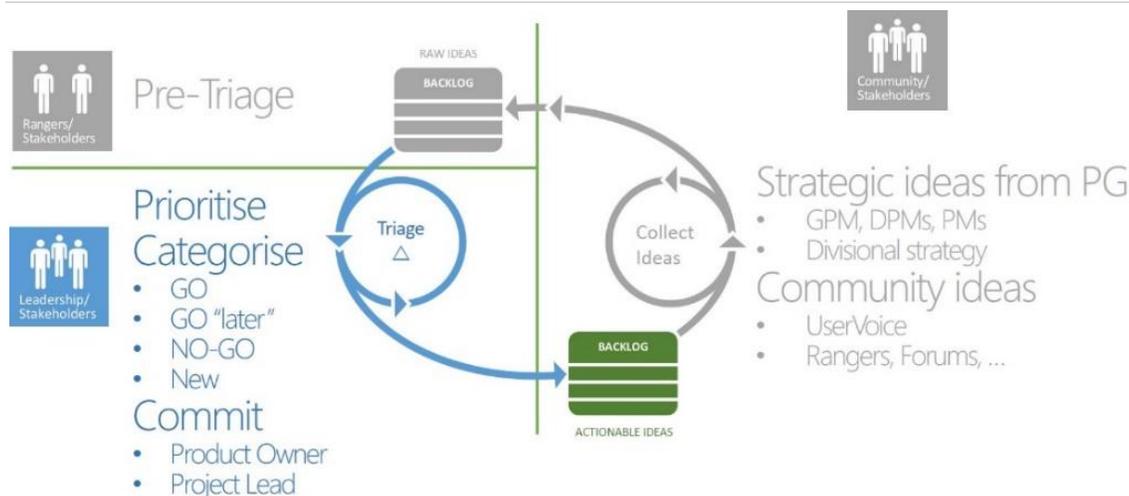


Figure 9. Triage process.

### GEM Top ideas

The best candidates are those with the highest business value, that can be delivered most quickly, and that impose the least risk to the main product line and mission statement.

When we brainstorm, it is important to consider variables such as value, viability, quality, cost, risk, and mission to be able to select and prioritize the right ideas.

- **Mission** An idea needs to be in harmony with our mission statement or we will disqualify it. We rejected the idea "add UserVoice integration service" because it had a low community vote and was not aligned with our mission statement.
- **Value** An idea needs to deliver additional value, quantified in terms of enabling more adoption of the main product, return on investment (ROI), and, most importantly, customer satisfaction. Delivering a gap-filling solution that unblocks Visual Studio unit testing has a higher value than a readiness solution that promotes future technologies. On the other hand, delivering a solution that has less value today probably has a greater impact than delivering a solution that may have astronomical value in 10 years.
- **Viability** An idea needs to be viable. The idea of sending a manned mission to Pluto may be a nice dream, and it may offer great value as our sun continues to expand. However, this idea is not yet technically viable.
- **Quality** An idea must be doable within predefined quality and within a time box (a roadmap).
- **Cost** An idea must be affordable and deliver a return on investment. Developing a stealth fighter that exceeds budgets or costs a fortune to maintain for a nation in desperate need of reconnaissance capabilities to patrol vast oceans and iced land mass may not deliver a return on investment.
- **Risk** An idea may be phenomenal, but it may be risky to develop or may be a risk to the main product line. Developing a utility that determines the effective permissions a user has across all services may be a strategic idea, but if it negatively affects the performance and stability of an operational product, it may be too risky.
- **Bandwidth** An idea needs to be realized, which requires bandwidth for a variety of dedicated resources. The best idea can quickly turn into a nightmare if the required resources, subject matter experts (SMEs), or processing bandwidth is not available.
- **In-flight limit** An idea requires focused attention. The meals served by a cook working on a handful of dishes are highly likely to be more successful (tasty) than meals served by a cook trying to coordinate dozens of dishes. As shown below, there may be a number of in-flight solutions at the time of the triage. This reduces the number of new projects that we can consider. Figure 10 depicts are triage cadence and overlap through the year.

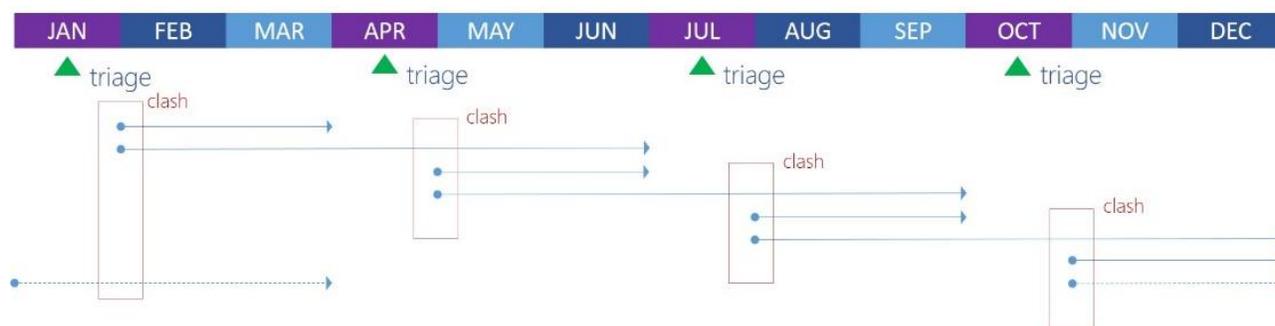


Figure 10. Triage cadence and project overlap.

#### GEM **A maximum of five concurrent flights (for us)**

- Complexity, duration, and personal involvement influence the in-flight limits, also known as work-in-process (WIP) limits.
- More than five (5) projects that are in flight in parallel place undue strain on our two (2) Program Managers (PMs), require increased coordination and context switching, and drain the community passion.
- You will need to experiment and refine this guidance to suit your PMs.

When the triage is finished, you should have an ordered list of potential ideas. There should be no doubt which solution ideas are critical, which are important, and which would be “nice” to consider.

At this point, we know what we should be considering for our next project adventures. The next step is to determine who will be the driving force behind the projects.

#### GEM **To be successful we need passionate stakeholders**

Experience has shown that distributed, part-time, and virtual teams thrive with committed and passionate stakeholders (for example, the Product Owner, Program Manager, and Scrum Master) who deliver guidance, direction, and a “pulse” that keeps the team spirit alive. Without these, the team eventually transforms into a black hole and dies, or it continues to trundle through space without any noticeable progress. The team will reflect the stakeholders’ passion or lack thereof.

## Identify passionate owners

With distributed, virtual, and part-time teams, three personas are pivotal to the success of the project. Scrum typically defines a team that consists of a Product Owner (PO), a Scrum Master, and the rest of the team. We recommend two additional owners, namely a Project Lead (PL) and a Program Manager (PM).

How you raise the visibility of the idea to identify these champions depends on you and your ecosystem. An approach we use is shown in Figure 11.

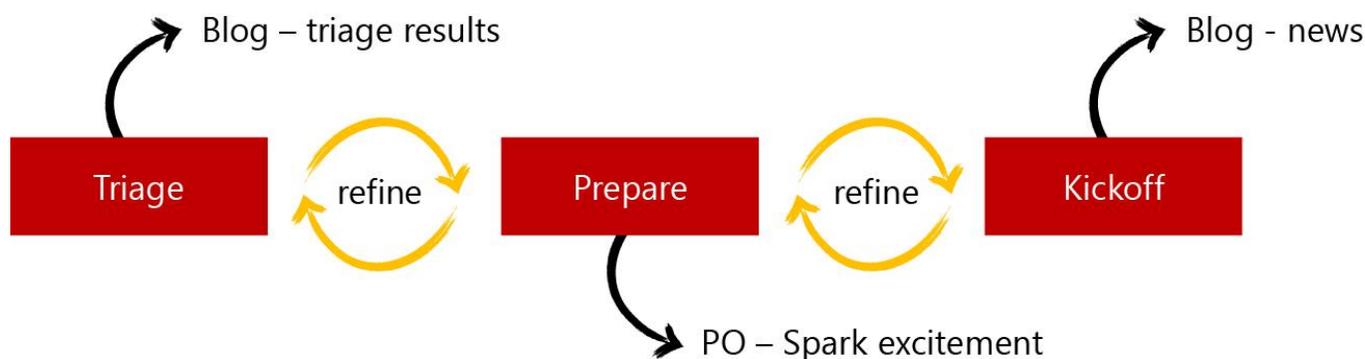


Figure 11. Raise the visibility of adventures “early.”

We share the results of the triage on an internal or external blog as soon as the triage is complete. Similarly, we share news on the project after we kick it off. This raises the visibility and overall transparency of the project idea and the project and nurtures interest and excitement for the solution.

While preparing for the kickoff (meeting), the Product Owner or representative delivers a 15–30 minute awareness session to explain why the project idea is important, what is needed, when it is needed by, and who the prospective subject matter experts are. This allows the community to see a context, mull over the adventure, and volunteer as Project Lead or as a contributor.

- The *Product Owner* is the face of the product, owns the product, makes the tough decisions on scope, and engages other stakeholders as needed.
- The *Project Lead*, also known as Dev Lead, is the team enabler and focuses on fine-tuning the team and representing the team on public forums. The Project Lead typically owns and maintains the project long after it has shipped and the rest of the team has moved on to other adventures.
- The *Program Manager* works with stakeholders and customers, focusing on team orchestration and keeping the team and backlog from stagnating. The [Program Management—Thinking about PM != PM and Visual Studio ALM Rangers \(Part 2\)](#)<sup>16</sup> and [Program Management—Are some of the ALM Rangers Symbiotic PM's?](#)<sup>17</sup> blog posts explore the definition of an ALM Ranger PM, the role's focus on orchestration, and other responsibilities.

Why three owners? Depending on the environment and availability of resources, we could combine these three personas into one Product Owner persona.

In our experience the Product Owner and Project Lead personas are, like most other ALM Rangers, available to projects on a part-time basis and, in the case of the Product Owner, might occasionally even change through the course of the project. The only permanent persona in our ecosystem is the Program Manager, who typically coordinates a handful of concurrent new projects, a number of service-mode projects, and a community of 200+, with the associated infrastructure and quality processes. To make matters more interesting, the Program Manager typically also acts as a tester/reviewer and occasionally works as a developer/contributor on multiple projects.

What becomes apparent is that none of these owner personas is truly committed to “the” project and therefore load balancing among the Product Owner, Project Lead, and Program Manager combines skills, knowledge, and passion, but more importantly creates crucial backup.

Once we have identified all the owners and everyone is committed to the project, we proceed with the final preparations and planning for the kickoff meeting (Figure 12).

<sup>16</sup> <http://aka.ms/vt9z43>

<sup>17</sup> <http://aka.ms/ypqmzz>

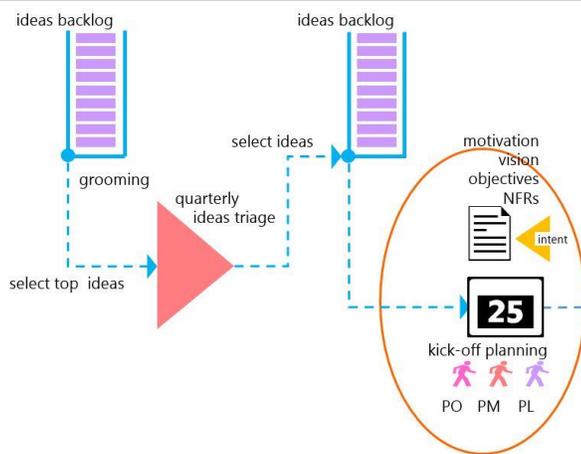


Figure 12. Lifecycle: Requirements and ownership triage final preparations.

### GEM

#### Every lead should have a backup

Everyone needs an opportunity to relax, focus on family > job > rangers, or focus on a complex and crucial task. To ensure that there is no single point of failure, we recommend that the Program Managers and Project Leads each have backups.

## Planning the kickoff to enable innovative teams

### GEM

#### KISS . . . Keep it simple and ship soon!

- Do less, better and quicker! We can always improve and do better next time!
- Share solutions sooner, even in alpha or beta state, to get early candid feedback.
- Empower the community to influence and take over solutions that are entering maintenance-only mode.

The owners now collaborate on the strategic intent for the idea, align with the overall vision, and start the planning activities for the kickoff event. In an ideal world, the most effective way would be to assemble the team in a face-to-face workshop, allow the team to organize itself, and allow the requirements to emerge as outlined in the [Agile Manifesto](#).<sup>18</sup> As mentioned before, we work as geographically distributed teams, with part-time volunteers, making a face-to-face and self-organizing ecosystem very challenging. The objective of the owners is to define the motivation, vision, objectives, and a few features. The owners prepare a realistic roadmap, deferring innovation, emerging designs, and self-organization to the team.

### NOTE

It is important to highlight that an idea is turned into a project only when the stakeholders, owners, and the team give the thumbs up. At this point, we have only the stakeholder and owners committed to the idea. In [Robert Bernstein](#)'s<sup>19</sup> words, we have a proposed treasure map and the captain of the ship but no crew (yet).

## Motivation

We derive motivation from the original description of the idea, captured on UserVoice, in an email, or during a discussion around the coffee machine. The motivation describes the problem the idea is supposed to solve, the features and benefits it may provide to the beneficiaries, and why we should care about the idea.

<sup>18</sup> [www.agilemanifesto.org](http://www.agilemanifesto.org)

<sup>19</sup> <http://aka.ms/nso70j> or <http://aka.ms/vsarindex>

An example borrowed from the ALM Readiness Treasure Map team defines the motivation as follows:

- *Provide a master catalogue (treasure map) of the available ALM Readiness content to guide users through the process of becoming proficient in ALM practices. Users include ALM Rangers and the ALM Community, looking to become more proficient in ALM, searching for specific guidance or wanting to track their progress through all the guidance.*

## Vision

The vision defines *what* we intend to solve. The vision outlines architectures, technologies, and core deliverables and highlights the key benefits. It is the first nugget we share with a colleague or senior manager who asks us about our project at the coffee machine. The ALM Readiness Treasure Map team has the following on its quick-reference chart:

- *Enhance the existing functionality in the ALM Readiness Treasure Map application to provide the user with an improved experience.*
- *The v1/2 vision provides a master catalog (treasure map) of the available ALM Readiness content to guide users through the process of becoming proficient in ALM practices. Enable a team to research and provide an innovative Windows 8 user experience and architecture reference sample solution, to be demonstrated at TechReady 16 in support of the Windows Store campaign.*

## Categorize solution

It is important to categorize the type of solution and the expected quality in the form of objectives, acceptance criteria, and release governance (Table 3). Always keep an eye on the overall mission and ensure alignment with the mission (Figure 13).



Figure 13. Our mission statement.

Category	Description
Quick-response sample	Quickly provide information, samples, and prototypes in response to feature gaps to supplement the product and knowledge base information. In some cases, the community adopts, enhances, and maintains these samples. See <a href="#">TOC—Quick Response Solutions</a> <sup>20</sup> for examples.
Guidance	Real-world, insightful, and practical guidance made up of topics such as crisp technology overviews, patterns, planning, usage, walkthroughs, and hands-on labs.

<sup>20</sup> [http://blogs.msdn.com/b/willy-peter\\_schaub/archive/2012/08/10/toc-quick-response-solutions.aspx](http://blogs.msdn.com/b/willy-peter_schaub/archive/2012/08/10/toc-quick-response-solutions.aspx)

Category	Description
	<a href="#">TFS Planning and DR Avoidance Guide</a> <sup>21</sup> and <a href="#">Version Control (previously named Branching and Merging) Guide</a> <sup>22</sup> are two popular examples. For more, see our <a href="#">Solutions Catalog</a> <sup>23</sup> .
Tooling	Solutions and tools for features missing in the out-of-box products or prototypes for proposed features. Ship a combination of source code, binaries, and documentation. <a href="#">WCF Load Test</a> <sup>24</sup> and <a href="#">Unit Test Generator</a> <sup>25</sup> are two popular examples, the former representing a solution for a missing feature and the latter a prototype. For more see our <a href="#">Solutions Catalog</a> .
Hybrid	A combination of two or more of the other solution categories.

Table 3. Solution categories that we use.

The expected quality ranges from minimum quality bars for quick-response solutions to high quality bars for work on a solution that impacts the main product (for example, Visual Studio). See Appendix A, “[Supporting toolbox](#),” for more information on process, quality, and associated checklists. Figure 14 shows the range of our quality bars.



Figure 14. Our quality bar.

## Objectives

There are many definitions for [SMART](#)<sup>26</sup> (Wikipedia, SMART Criteria, 2015) objectives. We use the Specific, Measurable, Achievable, Realistic, and Time-based version of SMART for defining the objectives for a release (for example, version 3 of the ALM Readiness Treasure Map solution). Table 4 describes these qualities.

Property	Description
Specific	Objectives need to be precise and unambiguous. Every team member should be able to explain each objective. There should be no disagreements among team members.
Measurable	Objectives must be measurable and aligned with expectations and acceptance criteria. The team and Product Owner must be able to decide when they have achieved the objectives.
Achievable	Objectives need to be achievable in terms of timeline and the capabilities of the team.
Realistic	Objectives must be aligned with the vision and relevant to the overall idea. For example, it is unrealistic to expect the ALM Readiness Treasure Map team to deliver version control guidance or release-management tooling. Such expectations will result in frustration and failure.

<sup>21</sup> <http://vsarplanningguide.codeplex.com>

<sup>22</sup> <http://vsarbranchingguide.codeplex.com>

<sup>23</sup> <http://aka.ms/vsarsolutions>

<sup>24</sup> <http://wcfloadtest.codeplex.com>

<sup>25</sup> <http://aka.ms/treasure50>

<sup>26</sup> [http://en.wikipedia.org/wiki/SMART\\_criteria](http://en.wikipedia.org/wiki/SMART_criteria)

Time-based	We must set the completion date for our objectives within a specified time box and align these dates with the overall roadmap. We naturally prioritize objectives that we must deliver by a specified date (when) over those we can be deliver any time (whenever).
------------	---

Table 4. SMART objectives.

Objectives are important for contextualizing the team's goals, creating a solid foundation and baseline for feature and story planning and tracking, and evolving the acceptance criteria for the release and its features.

As shown in the example below, we use [Fibonacci numbers](#)<sup>27</sup> (Wikipedia, Fibonacci number, 2015) to rank each objective by assigning a business value. Higher-ranking numbers indicate which objectives are the most important and have greater business value.

We can prefix an objective with *Exceeded* to indicate that it is an exceeded objective for the release. Once again, let us peek at the Readiness Treasure Map team's quick-reference chart:

- *Real-time update of treasure map metadata, without the need to publish and install a new release.*  
BV:8 (BV-Business Value)
- *Release by June 2014 and host a TechReady 19 Chalk Talk.*  
BV:5
- *Optional metrics of user navigation and popularity of islands, using Application Insights*  
BV:3
- *(Exceeded) Practical guidance/blog on the use of Application Insights to capture metrics in a Windows Store application.*  
BV:2
- *(Exceeded) Improved navigation in terms of UX and contents of treasure islands.*  
BV:1

## Features

From the core idea, the owners identify a few features for the team to invest in during the next and optionally future releases of the solution that realize the idea. The [INVEST](#)<sup>28</sup> (Wikipedia, INVEST (mnemonic), 2015) acronym, explained in Table 5, is useful for features in this context and stories and test cases in the future.

Property	Description
Independent	Self-contained, loosely coupled, but with tight cohesion
Negotiable	Until a feature is committed, we can renegotiate and revise it.
Valuable	Feature must add value.
Estimable	Feature must align with your estimation strategy.
Small	The size of the feature must be realistic and fit into the iteration cadence and release milestone.
Testable	Feature must have clear acceptance criteria that we use to create test cases and enable sign off of the feature as Done.

Table 5. INVEST acronym.

An idea, represented by an Epic, can have one or more associated features, whereas a feature is associated with zero or one Epics. Base information for an Epic includes a meaningful title, a crisp description, priority, business value, area path, and iteration (Figure 15).

<sup>27</sup> [http://en.wikipedia.org/wiki/Fibonacci\\_number](http://en.wikipedia.org/wiki/Fibonacci_number)

<sup>28</sup> [http://en.wikipedia.org/wiki/INVEST\\_\(mnemonic\)](http://en.wikipedia.org/wiki/INVEST_(mnemonic))

- The area path identifies the solution (flight)—for example, `\vsarDevOps` defines our DevOps hybrid solution, which contains a guidance, tooling, and patterns team.
- The business value uses [Fibonacci numbers](#)<sup>29</sup> (Wikipedia, Fibonacci number, 2015) to estimate the business value of the feature in relation to the objectives and other features.
- The iteration identifies the point in time when the team should start the feature—for example, `\FY14` indicates that the team should implement the feature during the 2014 financial year.

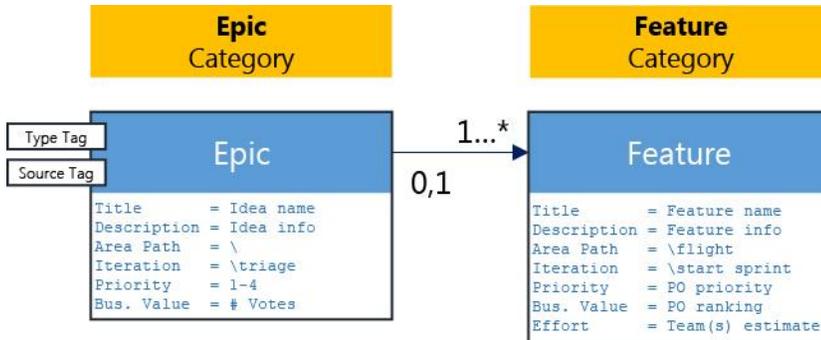


Figure 15. An idea is broken down into features.

Looking at the example hierarchy of an Epic (idea), six associated features, and one parked feature from the ALM Readiness Treasure Map team for the version 3 release, it is evident that both the features and their business value (BV) relate to the release objectives (Figure 16).

Project: VisualStudio.ALM Server: .com\DefaultCollection Query: Product Backlog List type: Tree					
ID	Title 1	Title 2	State	BV	Iteration Path
8425	Enhance ALM Readiness Treasure Map to address v2 backlog and deliver a v3		New	4	\
9495	As Alex, the technology consultant, I would like known previous issues resolved		New	8	\Projects\FY14
9496	As Alex, the technology consultant, I need to be able to update the treasure map meta data		New	8	\Projects\FY14
9497	As Robert, the captain, I want to host a TechReady 19 ChalkTalk		New	5	\Projects\FY14
9498	As Alex, the technology consultant, I need a more granular split of the treasure islands to improve navigation		New	3	\Projects\FY14
9499	As Mike, the program manager of the treasure map, I need metrics on which islands are most popular		New	2	\Projects\FY14
9500	As Alex, the technology consultant, I need practical guidance on using AppInsights to capture metrics		New	1	\Projects\FY14
9501	As Alex, the technology consultant, I would like known v3 issues resolved		New		\Parked

Figure 16. Example readiness treasure map features for v3.

## GEM

### One to three features per flight

The number of features defined per Epic depends on the granularity of each feature. We define one to three “must have” features and optionally one to two “exceeded” features. Defining more features disperses the team’s focus, complicates coordination, and introduces the potential of scope creep.

## Roadmap

Now that we have the proposed motivation, vision, category, objectives, and features, we can talk about the roadmap that defines when one or more teams will realize the idea. As discussed in [“One maintainable environment,”](#) we use shared sprints (iterations) for all our teams. The gray-haired users will remember the MSF tradeoff triangle, which states that if we fix one side of the triangle (the schedule, for example), changing the second side of the triangle (features) will affect the third side (resources). Therefore, when we fix the schedule or roadmap, any increase in scope implies that we will need more resources. A reduction of resources means we have to reduce scope. Figure 17 illustrates the triangle.

## NOTE

The second scenario (a reduction of resources) is common in our part-time, volunteer-based community and project teams.

<sup>29</sup> [http://en.wikipedia.org/wiki/Fibonacci\\_number](http://en.wikipedia.org/wiki/Fibonacci_number)



Figure 17. Trade-off triangle.

Our project teams typically work in two to five sprint time boxes, which vary based on the project category and set of features (Table 6). We will divide any project release that exceeds these time boxes into multiple, smaller, shippable releases to ensure that we have a consistent cadence, which becomes the *heartbeat* at a portfolio level.

Project type	Getting ready	Development	Quality	Time box
Quick-response sample	0–1	1–2	1	2–4 sprints
Quick-response prototype	0–1	1–3	1	2–5 sprints
Guidance	1	2–3	1	4–5 sprints
Tooling	1	2–3	1	4–5 sprints

Table 6. Number of sprints per project category.

Once the owners have agreed to the roadmap, they can define the release number, proposed team home, release milestone, and proposed solution home (Figure 18).



Figure 18. Example readiness treasure map categorization for v3.

**GEM****Time box flights from two to five sprints**

Working part-time, we have found that projects that exceed a four-sprint duration result in rapid loss of passion, energy, and feature value.

Do less and release sooner—we can always build on and do better next time!

At this point, we are ready to proceed to the next chapter and the next step in our project lifecycle. However, what about the ideas that are not selected and orphaned?

## What about the orphaned ideas?

As shown in Figure 19, some ideas may not make it onto the ideas backlog, failing to get a thumbs up from the triage group. Others may make it onto the ideas backlog but fail to attract the prerequisite passionate owners or are not the targets of any actions before their business value or relevance decays.

**NOTE** We refer to these ideas as orphaned or “left team-less.” Instead of simply rejecting the ideas, we submit them to the community for a final triage.

The community then decides whether to tackle the idea as a *community solution*, optionally supported by the ALM Rangers.

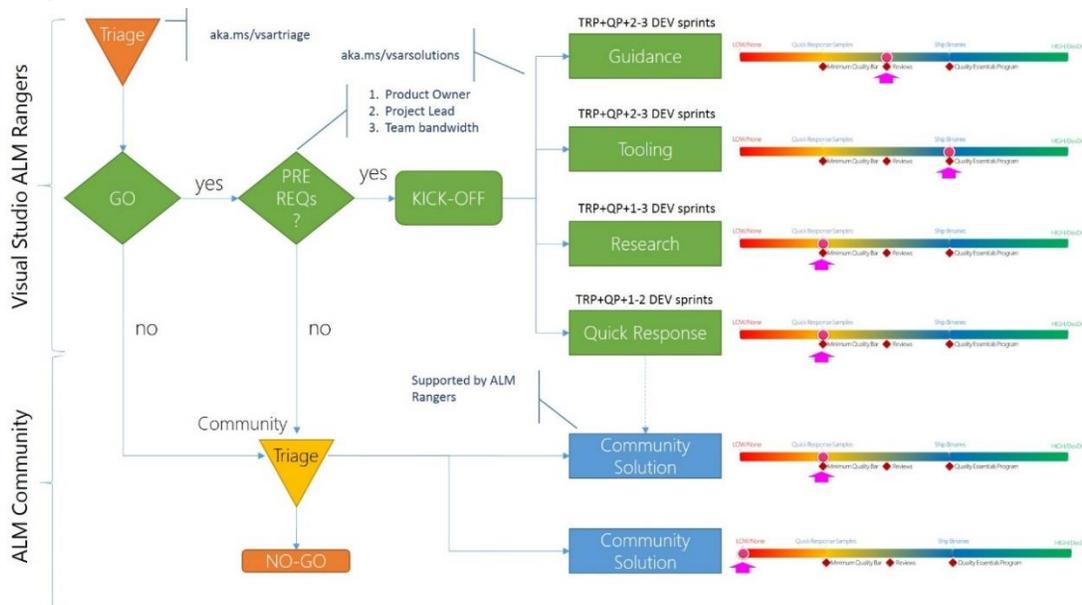


Figure 19. Process for selecting and rejecting project ideas.

Examples of quick-response solutions that evolved to community solutions include [TFS Community Teams Command Line Utility](#)<sup>30</sup> and [TFS Community Branch Tool](#).<sup>31</sup> An example of an idea that we developed as a community solution, with collaboration and support from the ALM Rangers, is the [Flight Plan](#).<sup>32</sup>

## Scaling flights . . . how many are too many?

A common business strategy is to do more, quicker and with less. It makes business sense and it supports the Agile principles such as Agile Manifesto’s principle #1: “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software” (Agile Manifesto, 2001b).

In the world of geographically dispersed teams working part-time on one or more community projects, the nature of leadership, collaboration, and scalability differs from how these qualities are experienced and managed in traditional small to enterprise-size organizations. We focused this book on the nontraditional but growing ecosystem of teams that are scattered around the globe, team members working on multiple projects and primarily on part-time community project releases, also referred to as flights.

Based on a few years of observations, using crude monitoring and analysis concepts, our ecosystem has revealed a few interesting patterns, shown in Figure 20.

<sup>30</sup> <http://aka.ms/qx2i43>

<sup>31</sup> <http://aka.ms/yv1770>

<sup>32</sup> <https://vsarflightplan.codeplex.com/>

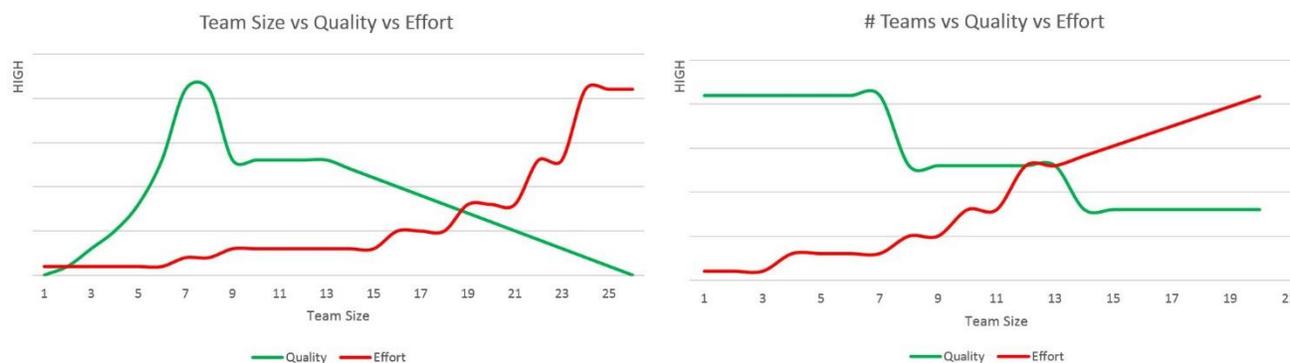


Figure 20. Perceived value and effort of projects as team size and number of projects grew.

What is evident is that our teams and overall program cannot scale indefinitely. The more that geographically distributed teams and team members need to be coordinated, the greater the amount of effort and distribution of focus is required from the Project Lead and Program Manager, from a project and overall ecosystem.

It is futile to argue with management—who are used to full-time enterprise or even full-time geographically dispersed project teams—that distribution, part-time availability, and lack of resource commitment are challenging.

- **Pillar #1—Respect for People** (Lean Enterprise Institute, 2009) can be a challenge for us. The Project Lead and Program Manager constantly need to be cognizant of infringing on precious family time, on the volunteer's real job, and the variation of traditions and cultures across time zones and around the globe. It is a truly interesting and rewarding environment, but definitely not an easy one to coordinate and balance with the demand for business value and roadmap timelines, both of which are usually intangible.
- **Pillar #2—Continuous Improvement** (Lean Enterprise Institute, 2009) is another potential challenge. The technology, process, and overall ecosystem constantly evolve as technology changes. Team retrospectives share learnings and dogfooding adapts accordingly.

Essentially, each Project Lead and each Program Manager have their own instance of an active queue of things they are tracking, working on, and deferring. Queue management is crucial, as the longer the queue, the longer the response cycle times and the greater the delay and impact on teams, with a rapid decline in overall quality and motivation.

#### GEM

#### MAX = 2–3 projects per Program Manager

For geographically distributed teams we recommend that  $n \leq 3$  per Program Manager (PM), allowing the PM to focus on the teams and associated flights.

We recommend the following guidelines (illustrated in Figure 21):

- A team consists of one or two feature teams, each with up to seven team members.
- A Project Lead owns one in-flight project release.
- A Program Manager is responsible for one to three in-flight project releases.
- A Scrum Master is responsible for one to three in-flight project releases until the team becomes self-organized and self-sufficient.

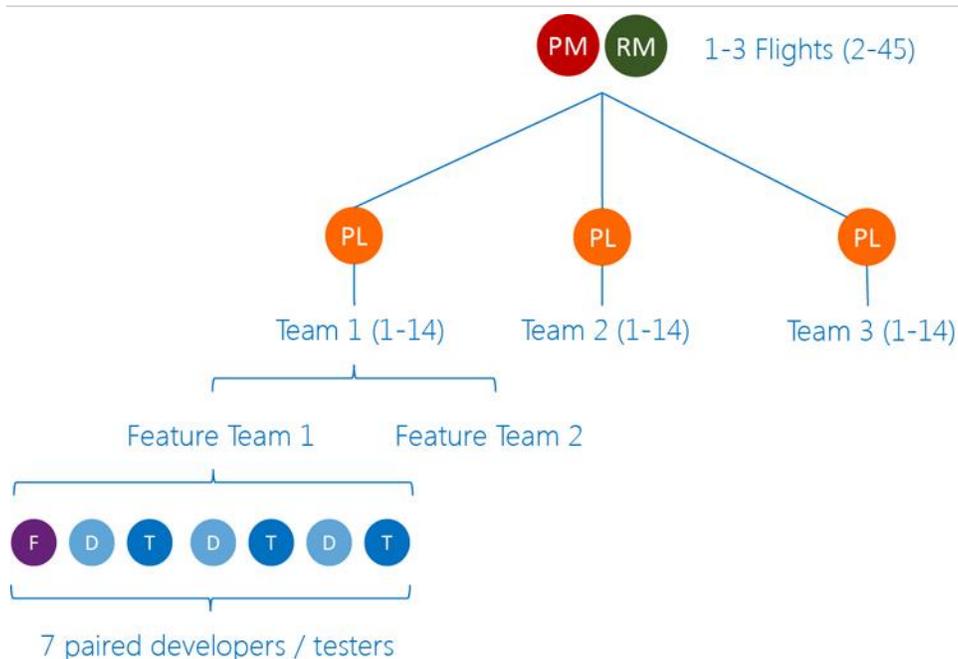


Figure 21. Team and flight scalability.

There is no quantitative or mathematical proof that we can apply for our geographically distributed, part-time, and volunteer-based community. However, it is important to remember that the volunteers are driven by passion and not business value or financial gain important for the program stakeholders.

We start small, remain focused, continuously reflect and adapt, and evolve our own ecosystem, which balances community passion and economic pressure. We protect and nurture our community because without it, we have nothing.

Knowing our limits is an important metric when triaging ideas and selecting our next adventures.

## Visibility from start to finish

A few years ago, project dashboards were in vogue and promised successful project management. Unfortunately, the overuse and misuse of widgets, gadgets, colors, and information created overwhelming dashboards that made it difficult to spot the “eye of the storm” and easy to ignore information until it was too late.

We rely on two types of dashboards to visualize status and trends over time.

The first is a visualization that should seem natural to anyone who has traveled by bus, train, ship, or plane and tracked their departure, arrival, or connecting journey on the status board. Using the [Flight Plan](https://vsarflightplan.codeplex.com/)<sup>33</sup> community solution, we visualize our recently landed, in flight, and imminent projects on our flight plan status board, pulling this information in real-time from our Visual Studio Online account. The use of a common user experience makes the information easy to consume, and the subtle use of colors makes us aware of potential issues.

- **Green** Once the Product Owner signs off on the flight (project), it has landed successfully, and therefore meets expectations.
- **Blue** The flight (project) is in flight (under construction) and is on track.
- **Red** The flight (project) is in flight (under construction) but is overdue and requires attention.
- **Gray** The flight (project) is one of the next ideas to prepare for takeoff.

<sup>33</sup> <https://vsarflightplan.codeplex.com/>

The flight plan status board intentionally does not overwhelm the team with detail, which can be explored by selecting (clicking on) flights. The focus is on showing a snapshot and status of the flights (projects), as shown in Figure 22, and presenting a crisp portfolio view.

Status	Title	Scheduled	Ver	Product Owner	Project Lead	Ruck Master
👍	Lab Management Guide Upgrade for TFS 2013	2014-02-28 2/4 aka.ms/vsar/news2	3.0	Vijay Machiraju	Mathias Olausson	Willy-Peter Schaub
🛑	On-Time VM Factory Service	2014-02-28 Research/MDT	4.0	Bijan Javidi	Rui Melo	Brian Blackman
🛑	Project Flight Plan (Status) Board	2014-03-15 Research/Community	1.0	Willy-Peter Schaub	Robert MacLean	N/A (Community Project)
✈️	Version Control Guidance Upgrade v3 (Branching, TFVC and NuGet)	2014-03-31 Guidance Upgrade	3.0	Matthew Mitrik	Michael Learned	Willy-Peter Schaub
✈️	TFS on Azure (IAAS) Planning and Ops Guide	2014-04-01 Strategic	1.4	Mario Rodriguez	Willy-Peter Schaub, James Szubryt	Jahangeer Mohammed
✈️	SAFe Awareness and Readiness	2014-04-30 Research/Training	1.0	Gregg Boer	Brian Blackman	Brian Blackman
✈️	Extract effective permission from TFS for audit purposes	2014-06-30 Research/Prototype	1.0	Mario Rodriguez	Jon Guerin	Willy-Peter Schaub
✈️	Enhance ALM Readiness Treasure Map to address v2 backlog and deliver a v3	2014-06-30 Research/Readiness	3.0	Willy-Peter Schaub	Robert Bernstein	Robert Bernstein
✈️	ALM Rangers DevOps / Building a Release Pipeline with TFS 2013	2014-06-30 Fast-track prep book	2.0	Roopesh Nair	Casey O'Mara	Marcus Fernandez
✈️	Version Control Guidance Upgrade v3 (Git for TFVC User)	2014-04-30 Guidance Upgrade	3.0	Matthew Mitrik	Michael Learned	Willy-Peter Schaub
💡	Desired State Configuration (DSC) Guidance and Scripts Library to support D...	2014-06-30 Guidance and Community	1.0		James Szubryt, Michael Fourie	
💡	Migration guidance of on premises TFS to VSO	2014-06-30 Whitepaper	1.0			

Figure 22. Innovative flight status board giving us a crisp portfolio view.

The flight plan status board presents a familiar view, as outlined in [Visibility and transparency is key to effective teams ... taking a closer look at the flight plan status board](#),<sup>34</sup> but we also use the Kanban board in Visual Studio Online (Figure 23) to visualize the same information.

Both solutions rely on work-item queries to extract real-time data from Team Foundation Server, but they present the data differently. Select the visualization that is most intuitive to you and your stakeholders!

Column	Count	Items
New		+ New Item, TFS Power Tools for 2015 and beyond, BDD ALM guidance to drive higher utilization of Unit Testing, Collaborate with p&ip Azure DevOps Patterns
Proposed	2/5	TFS Upgrade Guidance Revision, VS Architecture Tooling Guidance Revision
On Deck	2/5	Innovate guidance publication and review channel, TFS Image on Azure "Mooncake"
Committed	2/5	Technical Debt Guidance, Update the existing Test Planning and Management
In Progress	3/5	Create quality PowerShell DSC Resources for DevOps and ALM, Create quality PowerShell DSC Scenarios for DevOps and ALM, Evaluate new VS features by upgrading the VSARVSO
Product Owner Signoff	1/5	Version Control Guidance Upgrade v3 (Git for TFVC User)
Completed		

Figure 23. Innovative view of the Backlog using the VSO Kanban board.

Another visualization, which uses the Visual Studio Online web interface, focuses on detailed information and trends over time for all projects (Figure 24). We use a number of graphs to visualize progress and the allocation of work and tiles to show build results and detailed information, such as orphaned tasks and stories that require immediate corrective action.

**TOOLING** Visual Studio 2013, [Application Lifecycle Management](#)<sup>35</sup>

<sup>34</sup> [http://blogs.msdn.com/b/willy-peter\\_schaub/archive/2015/03/20/10508895.aspx](http://blogs.msdn.com/b/willy-peter_schaub/archive/2015/03/20/10508895.aspx)

<sup>35</sup> <https://msdn.microsoft.com/en-us/library/fda2bad5.aspx>

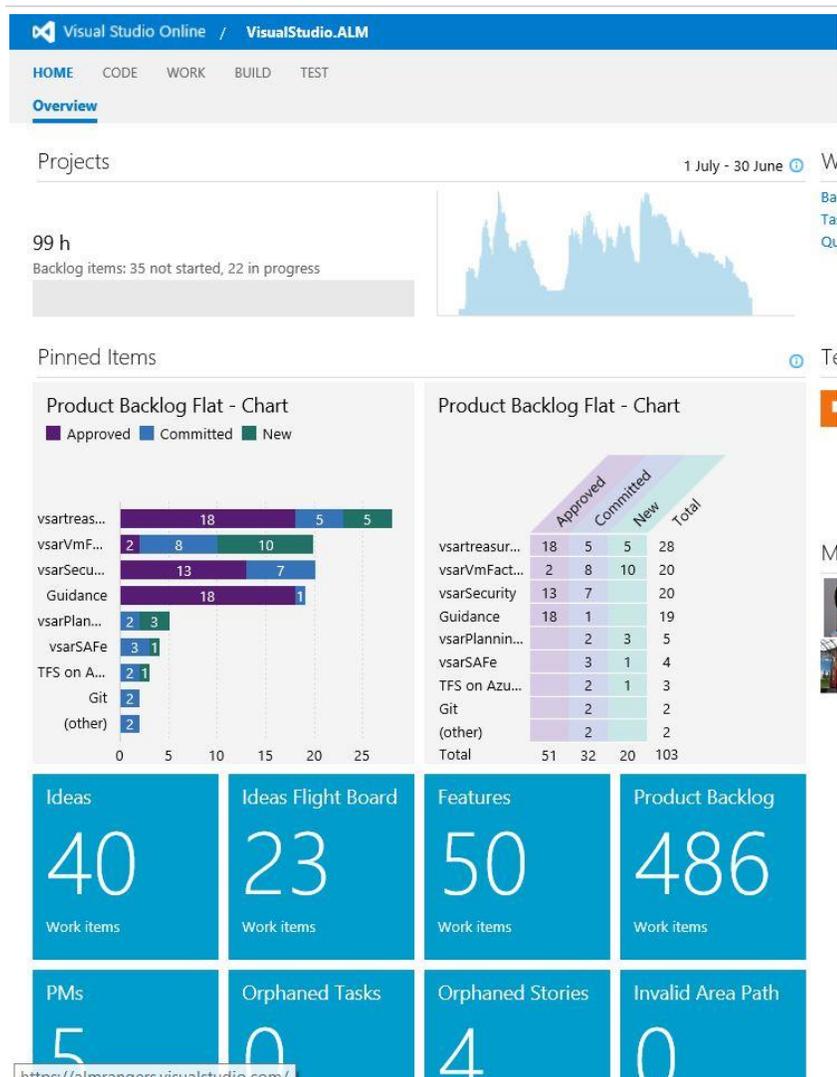


Figure 24. VSO overall dashboard.

Each project team has its own dashboard and can decide which information is important to provide to the team on the team's home page. These dashboards are typically less overwhelming and are customized to give the team a predictive (crystal ball) and reflective (are we in trouble?) state of their nation.

For example, the following team uses a simple dashboard to visualize its burndown, product backlog, and list of things to do.

We should all frown when looking at the "burndown" chart, which at a glance is not ideal. If we look at the details shown in Figure 25, for example 12h of work remaining with 115h of bandwidth, we realize that it should trigger a "ping" to the task owners, but it is not (yet) in a state that requires immediate corrective action.

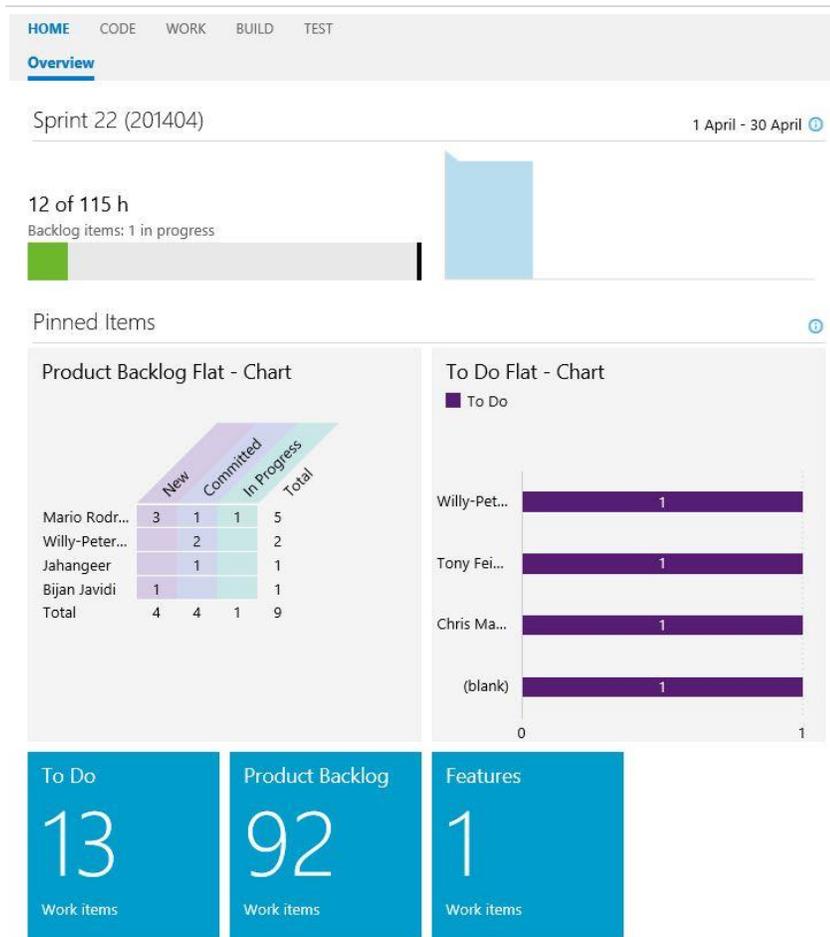


Figure 25. VSO team X dashboard.

Another example, Figure 26, shows a team that is tracking its gated build, which has a few issues and has returned to a green (A-OK) state. What should seem unusual is the lack of information shown for capacity and the burn-down of tasks in the two top graphs. This team uses a Kanban board to track stories, and their home page will likely evolve as future releases of Visual Studio Online allow us to customize the home page at a more granular level.

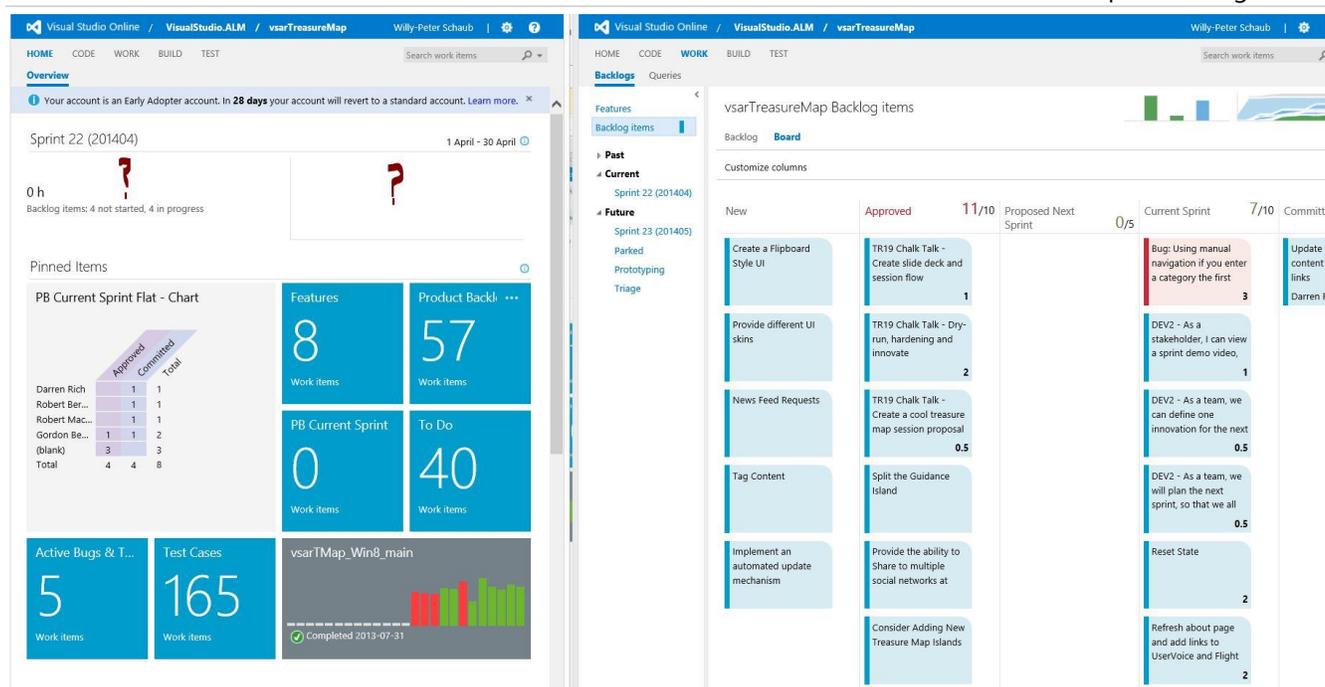


Figure 26. VSO team Y dashboard.

A key point is that a visualization can share the most important status and information about trends over time at a glance. They should not be an afterthought but rather instilled in the team culture before launching a new project.

## Dogfooding case study: Venturing into the cloud

Aligning the business, strategic, and community stars is an art that requires a number of passionate and experienced subject matter experts who are able to define and prioritize motivations, intent and visions and demand to define a clear roadmap.

We introduce the TFS on Azure IaaS guidance team because its members have managed to embrace a strategic solution with a skilled and passionate team scattered across Canada, Europe, India, South Africa, and the United States (Figure 28). A lot of credit needs to go to the team's Product Owner, Mario, who has in-depth technical knowledge and experience. Mario, shown in Figure 27, also has the ability to remain focused and nudge the team in the right direction.



Figure 27. Product Owner Mario (right), with Bijan Javidi at TechReady 14.

## Background information

The team's original vision was to deliver practical guidance and easy-to-consume readiness material to enable field resources to educate themselves on how to plan, implement, and maintain an effective TFS on Azure IaaS environment for product evaluation, testing, training, and production in the cloud. Find more information, see [TFS Planning and DR Avoidance Guide](#).<sup>36</sup>



Figure 28. Distributed TFS on Azure IaaS guidance team.

## Requirements and ownership triage

Under the mentoring eyes of the Scrum Master, the Product Owner, Project Lead, and Program Manager agreed to the project motivation, vision, features, and release objectives. They created a PowerPoint slide deck to deliver a crisp, concise, and motivating kickoff meeting. The clearly defined features and objectives allowed the team to effectively prioritize the backlog, react to impediments, and embrace additional and exceeded ideas.

Throughout the project, the team engaged the Product Owner, who assisted the team with tough decisions and evangelized the project with the stakeholders and the product group, creating more “wind in the sails” for the strategic project.

### TOOLING

[TFS Planning and Disaster Avoidance and Recovery, and TFS on Azure IaaS Guide](#)<sup>37</sup>

The team delivered practical guidance and a reference proof-of-concept (POC) environment, helping the community embrace and adopt TFS on Azure IaaS, as shown in the proof-of-concept diagram in Figure 29:

<sup>36</sup> <http://aka.ms/treasure5>

<sup>37</sup> <http://vsarplanningguide.codeplex.com/>

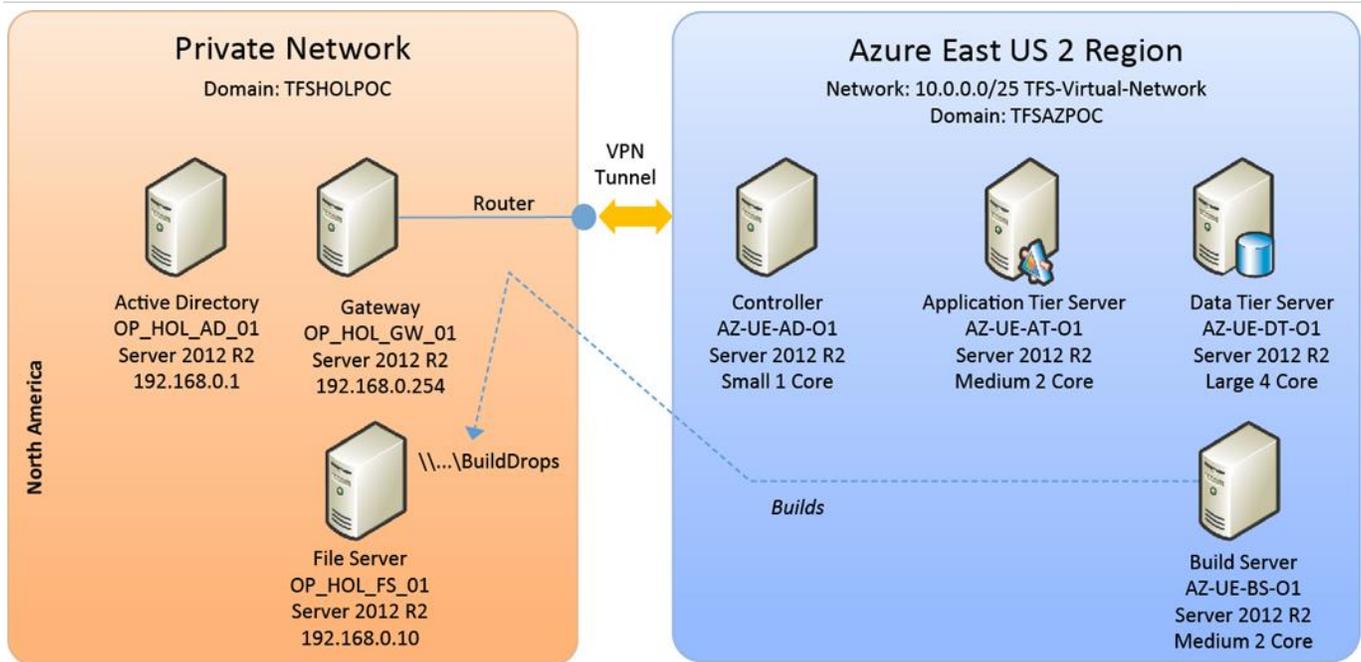


Figure 29. TFS on Azure IaaS POC, phase 1.

An interesting observation was that a clear mapping existed between team members, tasks, and responsibilities. All team members knew precisely what everyone expected from them. This included two subject matter experts who knew when to jump in to unblock the team and when to appeal a decision.

## Key learnings

- Transparency and communication ensure that we are all on the same path, which helps keep the team motivated and strong.
- The best ideas encompass the highest business value and can be delivered most quickly and with the least amount of risk while adhering to the mission statement.
- An active, passionate, and experienced:
  - Product Owner ensures we remain focused and aligned with all stakeholders.
  - Project Leads motivate the team and keeps the ship on course.
  - Scrum Master mentors and guides the team, encouraging consistent Agile practices.
- Value decays while dissatisfaction grows over time.
- The core steps are shown in Figure 30 and listed below:

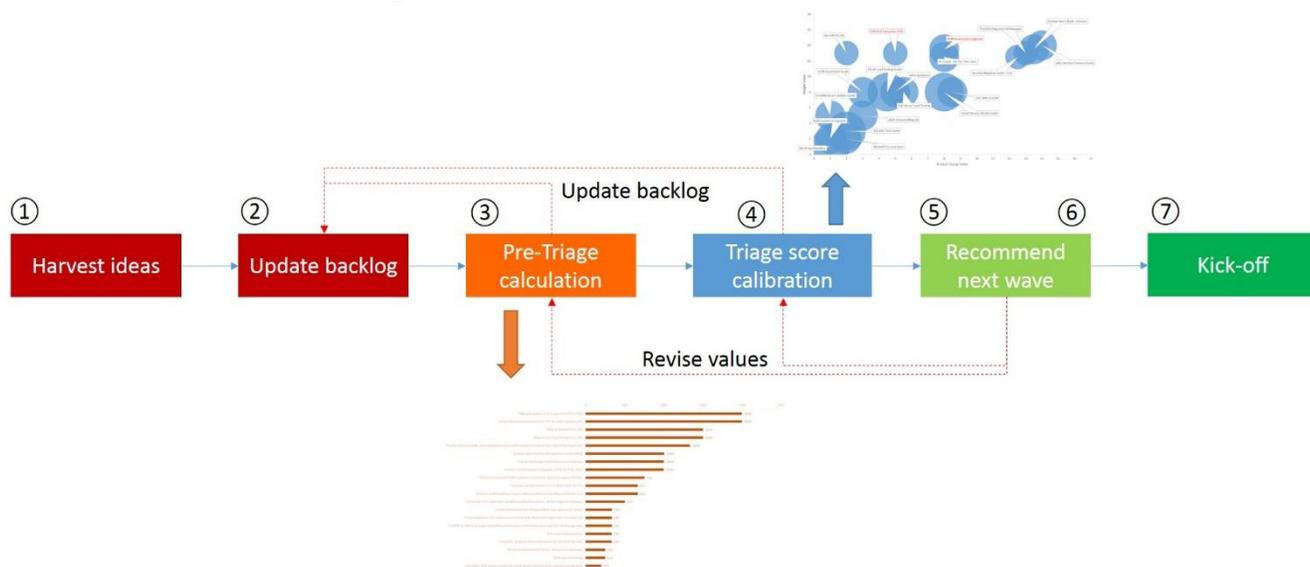


Figure 30. Core steps of ideas triage.

1. Harvest ideas
2. Update the backlog
3. Perform a pre-triage to get a feel of what is hot
4. Perform a triage calibration to balance ideas with our values and the values of the stakeholders
5. Recommend to stakeholders a concise list of ideas to consider and agree on value and priority
6. Recommend the prioritized list of ideas to the community
7. Launch an innovative team and pick the next idea

# Chapter 2: Getting ready

*This is no time for ease and comfort. It is time to dare and endure.*

—Winston Churchill

We continue with simplicity and add clarity! It is time to gather as a team, understand the why and the what, and agree on the how. Looking back at a number of projects tackled by geographically distributed, volunteer, part-time teams, it is evident that the time and effort invested at this stage of a solution is invaluable. Teams that “parachute” into a project without a clear vision, objectives, or requirements quickly fizzle out as resources vanish, or worse, they drag into never-ending nightmares with frustrating value drain. This phase is outlined in Figure 31.

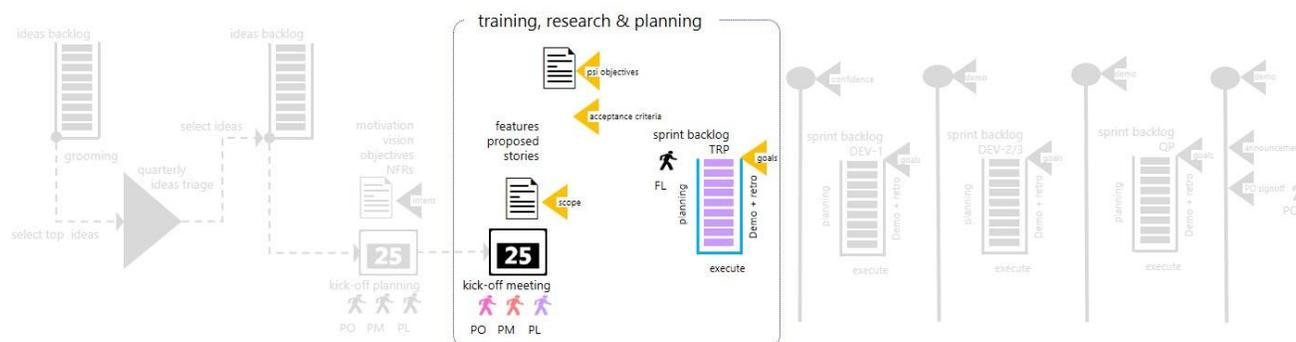


Figure 31. Lifecycle: Getting ready.

## Training-research-plan (TRP)

### GEM

#### Know your destiny!

As a scuba diver, you must plan your dive, limiting your maximum depth and time, and agreeing on the objectives. Unless you are on a military or search-and-recovery dive, the actual excursion is very agile. Yet everyone knows the plan. Everyone is in a position to mitigate changing environments and assist one's friend in times of need.

Each new project or subsequent release starts with training-research-planning and optionally ends with a focus on raising the quality bar and additional planning, as discussed in the section [“Raising the quality bar.”](#)

The introduction of training-research-planning and a quality-research focus will create a perception of a predictive waterfall-like process. With geographically distributed and part-time teams, many of whose members will never meet each other in person, it is imperative that we invest time to understand the idea and feature needs, formulate and agree on the requirements, and train the team to be in a position to optimize the delivery time and reduce the value drain.

The focus on training-research-planning can occur over either an entire or a partial sprint. With complex or new technology projects, a complete sprint is typical, whereas subsequent releases of the same solution usually require only a partial sprint.

The training-research-planning focus allows the team to understand and optionally refine the project:

- Motivation (why)
- Vision (what)
- [SMART](#)<sup>38</sup> objectives (Wikipedia, SMART Criteria, 2015)
- Features

<sup>38</sup> [http://en.wikipedia.org/wiki/SMART\\_criteria](http://en.wikipedia.org/wiki/SMART_criteria)

- Acceptance criteria (corresponds to customer expectations)
- Definition of done (corresponds to what needs to be completed prior to handing off a feature)

The sprint starts with a kickoff meeting, followed by research and training, and finally a sprint-planning meeting for the next and best-case estimations for subsequent development sprints.

## It all starts with the kickoff!

It is important to sell the concepts of the project idea to all stakeholders, which in our case includes every development team member. In addition, it is important to identify anyone who could contribute to or whose needs must be satisfied to ensure that the solution is successful and valuable.

The kickoff meeting delivers an overview, drives consensus, and allows the team to reach an agreement and plan the next steps in a very short time.

The meeting typically lasts 30–60 minutes and is facilitated by the Program Manager and Project Lead.

### GEM

#### **Provide context ... but not too early!**

It is valuable to enable the attendees to prepare themselves by sending them context for the meeting a few hours beforehand. Typically, we share the kickoff meeting slides in PDF format for preview and in the event that the teleconference infrastructure acts up during the meeting. Do not send out the information too early. Otherwise, you may influence the attendees and their prioritization accordingly.

A structured kickoff meeting allows everyone to engage and raise any technical, business, or political concerns. It also ensures that stakeholders are fully committed and, most importantly, that we align everyone's understanding of the idea, the project, and the solution.

Here's a ample kickoff meeting agenda:

- **Why** Motivation and beneficiaries
- **What** Vision, milestones, team leadership, and features
- **Plan** Deliverables, delivery channels, and process
- **Team** Skills needed and contact details

We have used a consistent, formal kickoff meeting template for years but have recently started investigating *A3 Problem Solving—Applying Lean Thinking* (Flinchbaugh, 2014). We need to select a strategy that suits us. There is no one size fits all.

## Planning the meeting

We start by filling in the template for the kickoff meeting agenda. Work with the Product Owner, Project Lead, Program Manager, and the designated Scrum Master to flesh out the details, include out-of-the-box thinking and ideas where appropriate, and agree on a date and time to host the kickoff meeting.

### GEM

#### **Respect family time!**

Avoid scheduling meetings on a Friday or weekend across time zones because you will encroach on precious Friday evening family time. Where possible, schedule a morning and an afternoon repeat meeting, allowing your community to pick the most convenient meeting for their time zone.

When working with part-time volunteers, give at least a week advance warning. Be sure that the invitation clearly defines the intent and appears highly professional to emphasize the importance of both the kickoff and the solution. We get only one chance to entice volunteers to join. The energy and passion levels of a team directly relate to the stakeholders' energy when hosting the event.

## Hosting the meeting

The facilitator(s) of the event are responsible for ensuring that the kickoff meeting is a roaring success. Within our context, this is typically the only time that we are able to gather all stakeholders in a face-to-face meeting, teleconference, or combination thereof.

If you are a facilitator, remember the following:

- Keep it simple!
- Record the meeting so that attendees can view the recording offline.
- Start and stop the meeting on time. We must respect everyone's time.
- Define the rules of engagement during the meeting—for example, when and how to ask questions.
- Keep the meeting on track, facilitate and defer discussions, and keep notes of all decisions and ideas raised.
- Keep everyone engaged but control dominating stakeholders. Remember that English is not everyone's native language and that some cultures will not interrupt (ever) out of respect.

The last step of the kickoff meeting should be a simple go/no-go decision and a simple confidence poll. Using Microsoft Lync, we can consider the two standard polls shown in Figure 32:



Figure 32. Kickoff meeting go/no-go and confidence polls.

Before we end the kickoff meeting, we make sure that everyone is happy with the outcome of the meeting and knows what the next steps are.

## Organizing the team

After the kickoff, we consolidate the feedback and continue refining the Epic and the features for the project, while we assemble the team and decide whether we need one or two feature teams. To minimize degradation of value over time, it is important to ship value as often and as soon as possible. It is important to realize that we can always raise an idea and plan a subsequent release, we can always do better next time, but we have only one chance to get each release shipped.

### GEM

$$\text{MAX} = (3-9)n + 2$$

The recommended team size is  $6^{+3}$  development team members per feature area (n), plus the Scrum Master and Product Owner, spanning across all feature areas of the one release identified by the Epic. For geographically distributed teams, we recommend that  $n \leq 2$ .

We eventually have to commit the team, which often raises the dilemma of having too many volunteers. See [I volunteered for a project, but am booked on another voyage ... why](#)<sup>39</sup> for more information.

One of the pillars of Lean thinking is respect for people. Be sure to keep the community and volunteers informed, never go dark, and promote the following concepts:

- Do not despair! There is more opportunity than bandwidth!
- Look for other project opportunities!
- Always listen for a call for help from teams by email, at the coffee machine, or during discussions! All the part-time resources are uncommitted by default, therefore team changes are inevitable.
- Team members never leave.
- Team members can be spread across other projects.

We send an email to the team members selected for the project. The message is similar to the example of the [welcome email from the vsarSecurity team](#).

It is important to remember to notify those not selected for the team in a professional and supportive manner, similar to the [team-limit-reached example from the vsarDevOps team](#).

## Objectives

The kickoff meeting not only allows us to share the story but also to take a step back and ensure that we align all the stars perfectly. After the kickoff, everyone knows why we are doing the project, what he or she needs to do, who will be responsible for what, and how confident the team is about succeeding. Most importantly, they know when they will start and when they will need to complete the first potentially shippable release.

## Team structure

The team structure depends on a number of variables, such as the organizational structure, process, framework, and even the experience and preferences of individuals within the organization. (See Figure 33.) We are continuously analyzing, improving, and adapting the process and team structure in line with [kaizen](#)<sup>40</sup> (continuous improvement and reflection). Let's explore an overview of our ecosystem and team structure.

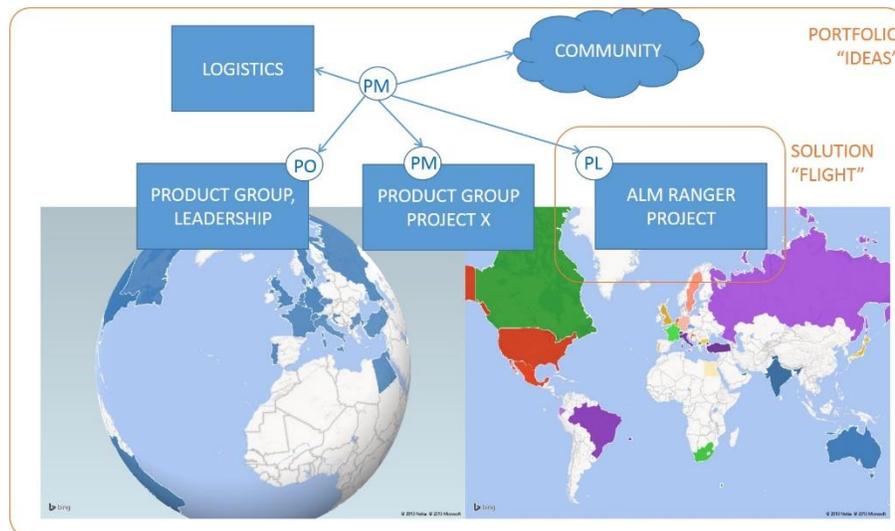


Figure 33. Our geographically distributed team ecosystem.

<sup>39</sup> <http://aka.ms/fc6d28>

<sup>40</sup> <http://en.wikipedia.org/wiki/Kaizen>

Figure 33 emphasizes that we operate in an atypical ecosystem. The portfolio level, where ideas and themes emerge, is worldwide. The solution level, where projects (flights) are instantiated, is also worldwide, and its members are geographically dispersed virtual teams based on the passion, knowledge, and time investments by part-time volunteers. Continuous evolution allows us to adapt and grow in response to an increase in solution demands and a reduction in delivery cadence, both of which stress the ecosystem and team members.

#### TOOLING

[Visual Studio Online provides you the tools you need to run your agile team](https://www.visualstudio.com/en-us/explore/agiletools-vs)<sup>41</sup>

[SharePoint—the new way to work together](http://www.sharepoint.com/)<sup>42</sup>

[Project Server—A flexible solution for project portfolio management \(PPM\) and everyday work](https://products.office.com/en-us/project/enterprise-project-server)<sup>43</sup>

## Portfolio “ideas” level

The stakeholders, consisting of community PMs, product group PMs, and product group leadership, triage the ideas and ensure that they align with the projects (flights) and features as well as with business and technology strategies. At this level, we work with the bigger picture and strive for a continuous delivery of value. The Product Owner (PO) is the interface to the product group and leadership and represents the solution (flight). The operational Program Manager (PM) coordinates the collaboration among the PO, the community, other projects, and the Project Lead (PL) of the solution (flight).

## Solution “flights” level

At the solution (flight) level, the interface is the Project Lead (PL), who represents one or two feature teams within the solution team, operating within the defined roadmap. The Product Owner (PO) is responsible for outlining, prioritizing, and approving features and the overall solution release. The Program Manager (PM) works with the Scrum Master (SM) to facilitate communication, mentor team members about the process, and help all parties identify and resolve impediments.

## Team “feature” level

Each of our feature teams consist of team members who pair up for development/contribution and testing/review responsibilities. Work is complete only when the team completes development *and* testing activities and meets the definition of done (DoD). This does not encourage or reintroduce a waterfall sequence because the team can create unit tests and test cases and work on these in parallel to development efforts.

#### GEM

#### Transparency rules!

Transparency and visibility into requirements, plans, and status are required down to the team level. If you leave a solution team or feature team in the dark for too long, you will typically return to an empty team room because your rock stars have moved on to other adventures.

## Team infrastructure

During the early sprint(s), we focus on an infrastructure as well as three main pillars, as shown in Figure 34. The infrastructure and pillars are part of the overall nonfunctional requirements.

<sup>41</sup> <https://www.visualstudio.com/en-us/explore/agiletools-vs>

<sup>42</sup> <http://www.sharepoint.com/>

<sup>43</sup> <https://products.office.com/en-us/project/enterprise-project-server>

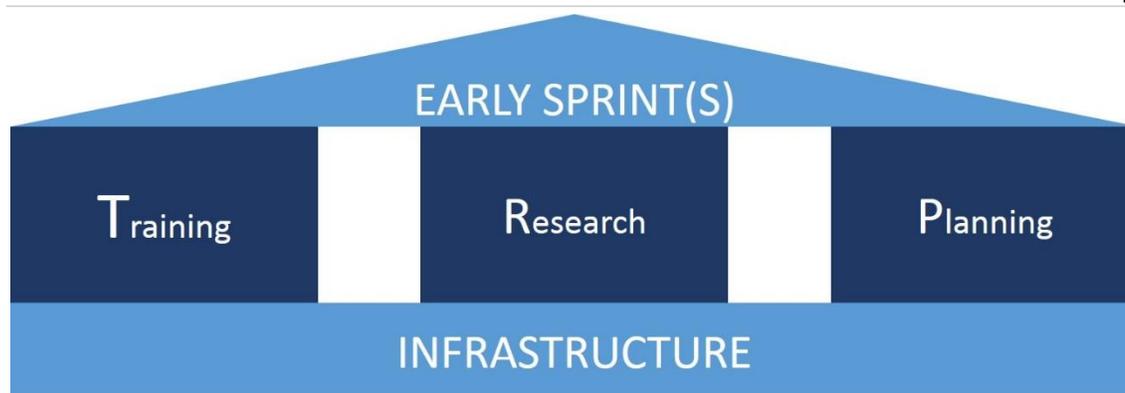


Figure 34. Sprint(s) focused on the training-research-planning foundation and pillars.

The infrastructure delivers a functional and consistent environment to all team members. Consistency is paramount to reduce ramp-up time. Consistency allows resource sharing and productivity, especially when dealing with geographically distributed, virtual, and part-time teams.

**NOTE** This book and our dogfooding has been based on the Visual Studio Online Application Lifecycle Management (ALM) Infrastructure, which provides a flexible and agile environment that adapts to your team's needs, removes barriers between roles, and streamlines processes so that you can focus on delivering high-quality software faster and more efficiently. You can apply the concepts we are covering with the ALM solution of your choice; however, we encourage you to explore [Visual Studio Online](#).<sup>44</sup>

Using Visual Studio Online (VSO), we typically create the following environment for each solution team:

**GEM** **Plan!** Use the [TFS Planning and DR Avoidance Guide](#)<sup>45</sup> to plan your team environment and the [Version Control \(ex Branching and Merging\) Guide](#)<sup>46</sup> to explore the branching strategies, version control gems, and related guidance.

- New project team (see Figure 35), contained within a team project, adding both the Project Lead and Scrum Master as team administrators. The team project provides a repository for source code and work items for one or more teams.

<sup>44</sup> <http://www.visualstudio.com/products/visual-studio-online-overview-vs>

<sup>45</sup> <http://vsarplanningguide.codeplex.com/>

<sup>46</sup> <http://vsarbranchingguide.codeplex.com/>

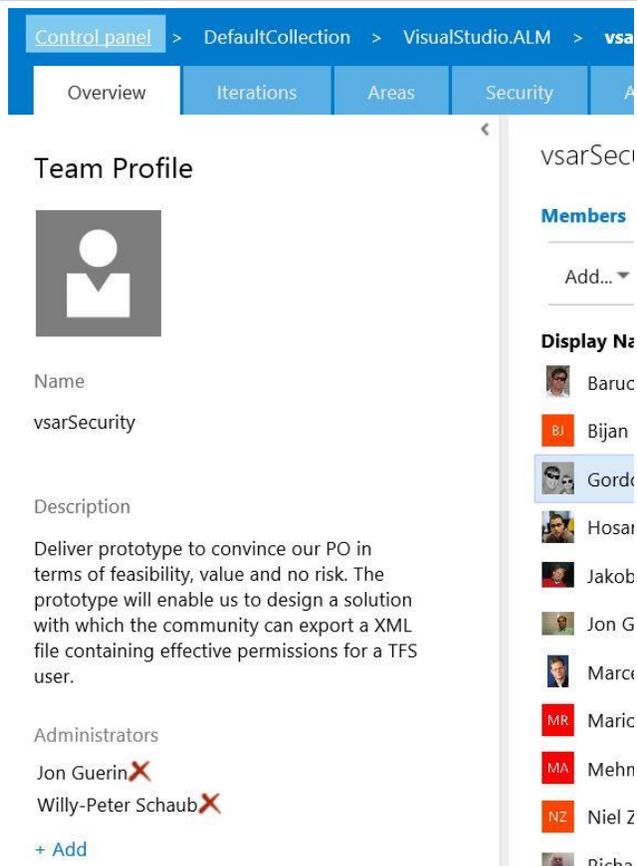


Figure 35. A team within a team project.

- Version control folder and appropriate subdirectories, created by using the TFS Branch Tool or manually. We recommend starting with a main folder, which can be promoted to a branch in the future, or a main-only branching strategy. We define a consistent folder structure (Figure 36) so that team members will intuitively know where to drop source code, documentation, and other artifacts, such as videos.

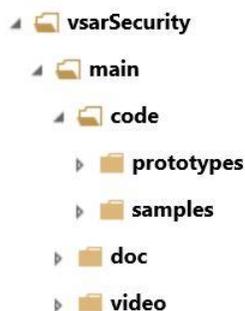


Figure 36. Consistent version control folder structure.

#### TOOLING

[Version Control—Enterprise-ready source code management for teams of all sizes<sup>47</sup>](https://www.visualstudio.com/explore/version-control-vs)

- Assign the iterations to the project, based on the roadmap of the project (Figure 37).

<sup>47</sup> <https://www.visualstudio.com/explore/version-control-vs>

Control panel > DefaultCollection > VisualStudio.ALM > vsarSecurity

Overview Iterations Areas Security Alerts Vers

## Iterations

### Iterations

Select the iterations you want to use for iteration planning (sprint planning). Selected iterations will appear in your backlog view as iterations available for planning.

---

New New child

	Iterations	Start Date	End Date
<input type="checkbox"/>	▸ FY13	2012-07-01	2013-06-30
<input type="checkbox"/>	▾ FY14	2013-07-01	2014-06-30
<input type="checkbox"/>	▸ Sprint 13 (201307)	2013-07-01	2013-07-31
<input type="checkbox"/>	▸ Sprint 14 (201308)	2013-08-01	2013-08-31
<input type="checkbox"/>	▸ Sprint 15 (201309)	2013-09-01	2013-09-30
<input type="checkbox"/>	▸ Sprint 16 (201310)	2013-10-01	2013-10-31
<input type="checkbox"/>	▸ Sprint 17 (201311)	2013-11-01	2013-11-30
<input type="checkbox"/>	▸ Sprint 18 (201312)	2013-12-01	2013-12-31
<input type="checkbox"/>	▸ Sprint 19 (201401)	2014-01-01	2014-01-31
<input type="checkbox"/>	▸ Sprint 20 (201402)	2014-02-01	2014-02-28
<input checked="" type="checkbox"/>	▸ Sprint 21 (201403)	2014-03-01	2014-03-31
<input checked="" type="checkbox"/>	▸ Sprint 22 (201404)	2014-04-01	2014-04-30
<input checked="" type="checkbox"/>	▸ Sprint 23 (201405)	2014-05-01	2014-05-31
<input checked="" type="checkbox"/>	▸ Sprint 24 (201406)	2014-06-01	2014-06-30

Figure 37. Project iterations / sprints.

- Assign the proposed features to the team.

**TOOLING** [Collaborate in a team room](#)<sup>48</sup>

- Create a Program Manager (PM) bucket backlog product item (PBI), shown in Figure 38, to serve as the go-to place to find the release objectives and the acceptance criteria. Assign this PBI to the Program Manager or Scrum Master. Scrum Master is the recommended default.

<sup>48</sup> <https://msdn.microsoft.com/en-us/library/dn169471.aspx>

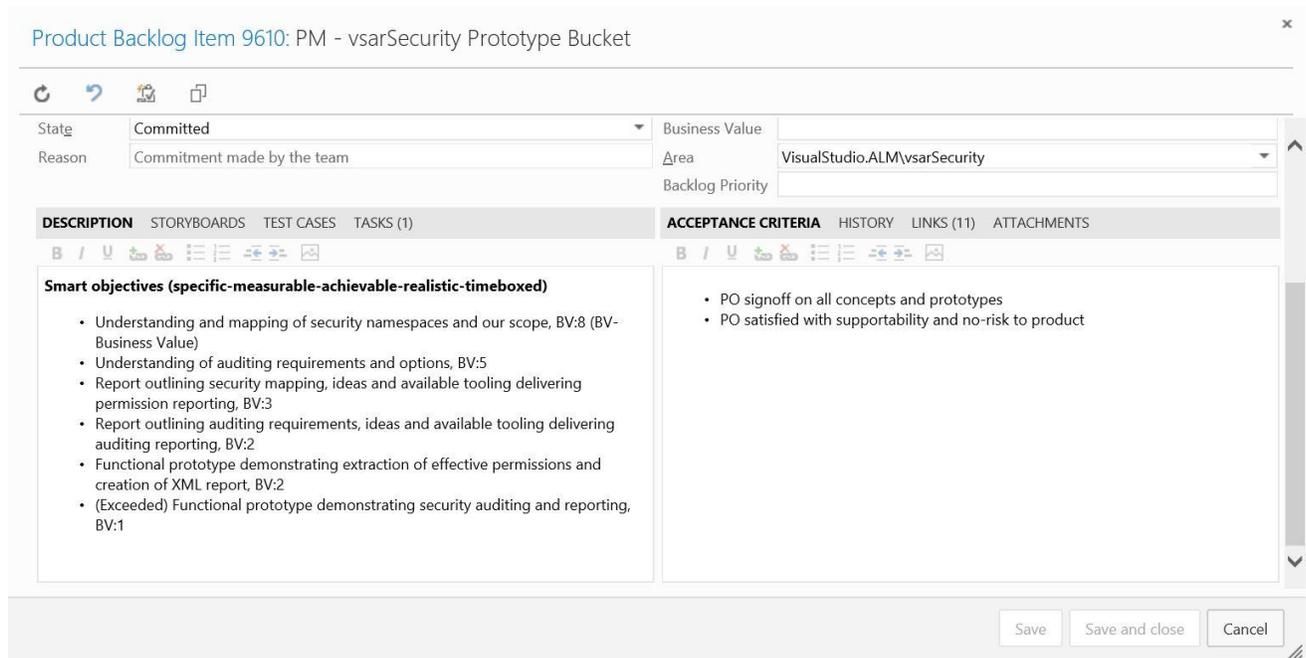


Figure 38. Program management PBI.

- Create the default release and sprint stories (Figure 39) and link them to the program management PBI. The roadmap defines the number of DEV sprints and, as shown in Figure 40, we can use the program management PBI as the parent for scrum tasks containing status information.

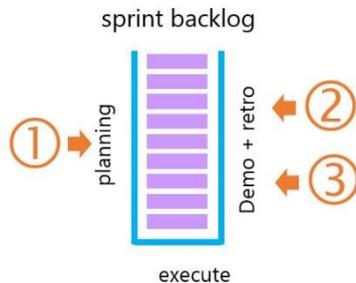


Figure 39. Sprint default stories.

The ① planning story is mandatory for all sprints, whereas the ② retrospective and ③ demo stories are optional for sprint(s) focused on training-research-planning only.

- PM - vsarSecurity Prototype Bucket
  - DEV1 - As a stakeholder, I can view a sprint demo video, so that that I can get a quick overview of deliverables
  - DEV1 - As a team, we can define one innovation for the next sprint, so that we can improve our project continuously
  - DEV1 - As a team, we will plan the next sprint, so that we all agree on what, when and how
  - DEV2 - As a stakeholder, I can view a sprint demo video, so that that I can get a quick overview of deliverables
  - DEV2 -As a team, we can define one innovation for the next sprint, so that we can improve our project continuously
  - DEV2 -As a team, we will plan the next sprint, so that we all agree on what, when and how
  - HIP - As a stakeholder, I can view a sprint demo video, so that that I can get a quick overview of deliverables
  - HIP - As a team, we can define one innovation for the next sprint, so that we can improve our project continuously
  - HIP - As a team, we will plan the next sprint, so that we all agree on what, when and how
  - Stand-up 20140414
  - TRP - As a team, we will plan the next sprint, so that we all agree on what, when and how

Figure 40. Program management default stories.

- (Optional) Customize the storyboard columns (Figure 41).



Figure 41. Custom board columns.

For example, we've added the Proposed Next Sprint column to allow the team to drag stories covered in the next sprint into one common column. The Current Sprint column contains the stories to address in the current sprint, whereas the Ready for Review column identifies all stories that are ready for testing, review, and validation. These columns do not introduce new story states but vastly improve visibility and planning.

- (Optional) Refine the team home page using images, tiles, and lightweight graphs for improved tracking.

## Training . . . learning new things from the SMEs

The first pillar of TRP is the concept of readiness for the team, which includes framework, architecture, and technology training.

- **Framework** training ensures that all team members and stakeholders have an understanding of the selected framework and methodology and the principles of Lean and Agile thinking.
- **Architecture** training ensures that all team members have an understanding of the underlying solution architecture—for example, the Microsoft .NET Framework—which helps the team build valuable solutions on the relevant architecture.
- **Technology training** complements architecture training and delivers focused technology training, especially when the team is using early adopter technologies such as Desired State Configuration (DSC).

## Research . . . investigate and model requirements

The second pillar of TRP includes research about the technology, concepts, functional, and nonfunctional requirements. We must invest time to research and understand the requirements and reduce the unknowns in order to estimate and plan the subsequent sprints and the shippable solution.

Before a scuba diver embarks on a search and recovery, the diver researches the equipment, situation, conditions, and proposed dive profile to reduce the risk and the search time and be in a position to plan and deliver a successful recovery (Figure 42).



Figure 42. A safe, enjoyable, and successful dive starts with research and planning.

## Planning

The third pillar of TRP reoccurs at the end, with a focus on planning and estimating the next sprint accurately and subsequent sprints as best as possible.

[Martin Hinshelwood](#)<sup>49</sup> introduces the Agility metrics (Hinshelwood, 2014) as follows:

- *We need to focus on value (Business Outcomes) to make sure that we are producing something that our customers value while not alienating our employees or spending too much money doing it.*
- *We need to focus on lead time (Time to Market) to make sure that we can take those things of value that we are producing and quickly get them, in short order, into the hands of our customers.*
- *We focus on quality (Ability to Innovate) to make sure that what we do deliver works, is easy to use, has useful features, and does not have a high maintenance cycle.*

Delivering value and focusing on quality are self-explanatory. If we want to invest a lot of money in a new vehicle, a new house, or a new service, we will only consider spending our hard-earned money if we perceive the investment as valuable and of quality.

---

<sup>49</sup> <http://aka.ms/p0gsr3>

## Estimating and prioritization fundamentals

### 1. Why should we care about estimation?

Assume we decide that we want to walk from point A to point B (for example, from Ladner to Tofino). The first thing we would typically do is estimate the time it will take us to walk this distance and break down the journey into manageable and measurable directions.

Using our favorite search engine (see Figure 43), we get ❶ a view of the journey and an indication whether the adventure is viable. By changing the journey from Ladner to Whitehorse, Yukon, the search result would give us visual clues that we are heading for a potential disaster or would need to break down the adventure into smaller and shorter excursions. We also get ❷ nonfunctional requirements and impediments, ❸ an estimation of effort, distance and time, and ❹ directions, the plan.

If we do this for a 300 kilometer stroll, why would we not do the same for a mission-critical and potentially expensive software project?

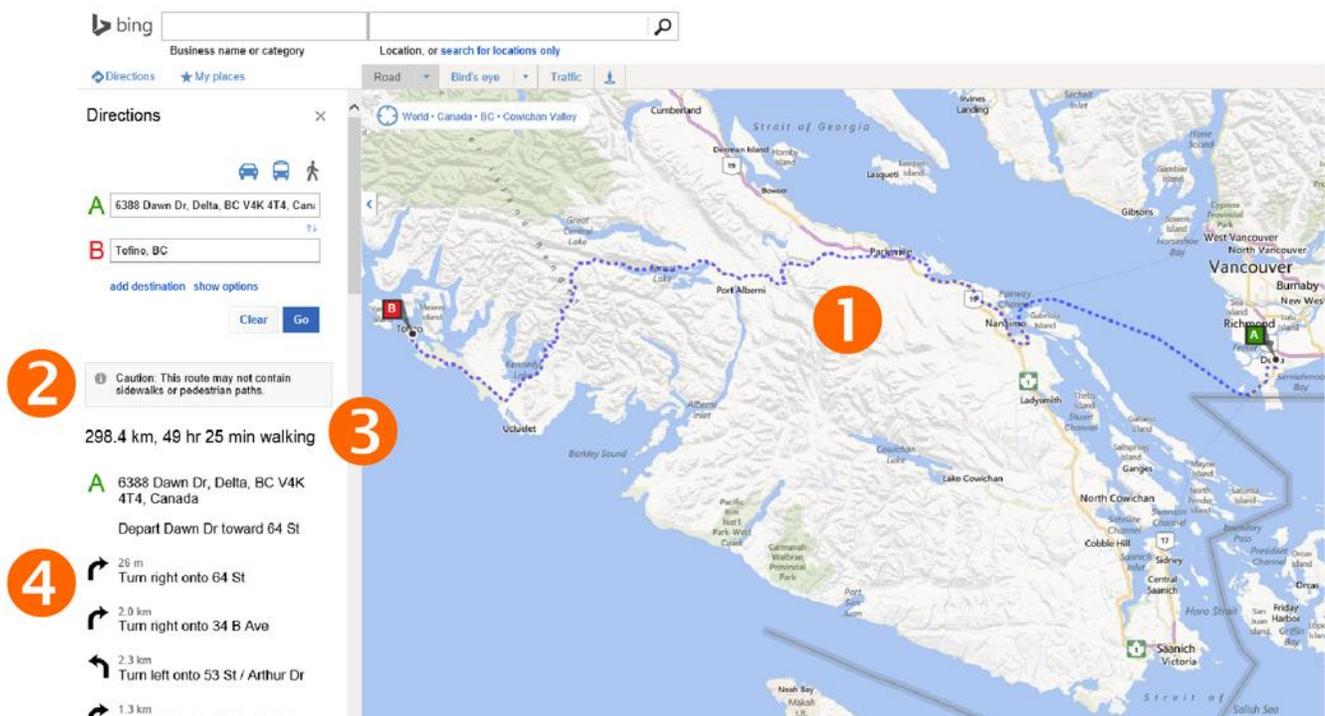


Figure 43. Hypothetical project to walk from Ladner to Tofino.

Even for lightweight and part-time projects, we need an organized and well-maintained backlog to determine the feasibility and estimate the effort, time, and cost of a proposed solution.

Let's take a quick look at features, costs, the ideal-world, and how we deal with normalized estimations.

### Requirements estimations

In the section "[Planning the kickoff](#)," we described how the Product Owner estimated and prioritized features in terms of business value. For product backlog items and stories, we estimate the business value using story points (SP), representing a combination of complexity, the number of features, and a clarity or uncertainty assurance.

Estimations can be based on story points, T-shirt or tractor sizes, Fibonacci series derivations, or any integer value of choice.

## Cost estimations

Seasoned teams will have a cost estimate per story point and an average velocity of story points per sprint. Multiplying the story points by the cost per story point, we can determine an estimate for the release backlog. With part-time community projects, we often misunderstand the resource bandwidth commonly based on volunteer time as zero cost.

Although teams and volunteers might not perceive monetary value in what they do, it is important that we estimate Product Backlog Items (BPI) and stories as accurately as possible. Effort, expertise, and the sacrifice of precious family time must be calculated in monetary value, which is understood by both the technical and business stakeholders.

## Ideal estimations

In an ideal world, we would estimate each functional and nonfunctional requirement in terms of expertise, cost, and hours (time) required to analyze, design, develop, test, and ship the feature. Multiplying cost by time, adding up all the feature costs, and adjusting the loss of value due to delay, would give us a total release cost.

However, in an agile world, we strive to deliver working software over detailed documentation, customer collaboration over a negotiated contract, and response to change over following a plan. These agile values make it very difficult to determine a precise time or cost estimation.

## Normalized estimations

For our distributed, virtual, and (especially) part-time teams, we use a normalized estimation strategy based on a number of prerequisites:

- Customized [Fibonacci](#)<sup>50</sup> (Wikipedia, Fibonacci number, 2015) series (for example, 0, 1/2, 1, 2, 3, 5, 8, 13, 21, and so on.). When we are playing [Planning Poker](#)<sup>51</sup> (Wikipedia, Planning poker, 2015), 0 means no effort required and  $\infty$  implies that it is too complex or impossible to estimate the effort.
- Estimations are based on a perfect world, full-time, and colocated team and the unconstrained availability of expertise.
- We base estimations on one developer/contributor and a tester/reviewer pair per story. This implies that a story is not complete until we have developed and tested it and the story is ready to ship.
- Estimations need team consensus. If there is disagreement, we discuss and reestimate the story or defer the story for further investigation.
- The team identifies a story that can potentially be developed and validated in a full day and associates it with 1 story point.
- We estimate other stories relative to the 1-story-point reference story. A 13-story-point story should therefore take approximately 13 times as much effort (not time) to ship than the 1-story-point reference story.
- We estimate stories considered smaller than the 1 story-point reference as 1/2.
- Stories estimated as 21 or higher per sprint need to be broken into smaller stories.

Whether playing online [Planning Poker](#), using Lync polls, or estimating with cards sitting around a table, the facilitator ensures that there is no bias caused by others showing their estimations or through the team following the estimations of the more experienced members or team leads.

As normalized estimations are atypical, it is important that we ask ourselves a few questions and delve into the details.

---

<sup>50</sup> [http://en.wikipedia.org/wiki/Fibonacci\\_number](http://en.wikipedia.org/wiki/Fibonacci_number)

<sup>51</sup> [http://en.wikipedia.org/wiki/Planning\\_poker](http://en.wikipedia.org/wiki/Planning_poker)

So is 1 story point == 8 hours?

No! A story point represents the perceived and relative sizing of stories.

A story with one (1) story point can be potentially developed, tested, and shipped in a day by the team. Other teams may require more or less than a day, and not every team's day is necessarily eight (8) working hours long.

Once the team has established a known and consistent sprint velocity, we can estimate how many days it might take to implement the team's backlog:

$$\text{Estimated duration in days} = (\text{days per sprint}) * (\text{backlog size estimate} / \text{velocity})$$

This is our degree of accuracy when we estimate at a feature (release) and Story (feature team) level.

What about part-time estimations?

We do all estimations based on a full-time team and day, as discussed. Then we estimate how much the average velocity is on a part-time basis as follows:

- Estimate how much of a calendar week can be committed.
- Multiply the weekly velocity by the percentage of possible part-time commitment.
- Round the result to the nearest integer.
- The result is the maximum a part-time developer/contributor and tester/reviewer team can deliver in one sprint.

We work with a best case of 3 story points per part-time pair per sprint. Any story estimated larger than the 3 story point/sprint velocity is broken down into smaller stories. In other words, any story estimated as 5, 8, and 13 are not potential candidates for a forthcoming sprint. See [Radio TFS and the GREAT curve ball question about \(ex-nightmare\) shorter delivery cycles](#)<sup>52</sup> for a more detailed breakdown.

**GEM**

**1/8 and max 3**

The best-case part-time commitment achieved with our community maxes out at one-eighth, and the maximum sprint velocity estimated as 3 story points. Do not strive for a higher velocity as the family>job>community balance is essential to keep volunteers passionately engaged without burning out!

What about estimations across different teams?

In an ideal world, the velocity is determined on a team level and typically varies from team to team. We do not commonly dedicate resources to a project full-time or long-term. Instead, the team members are engaged in multiple projects and jump from project to project, making it very difficult to establish a consistent velocity per team. (See Figure 44.)

<sup>52</sup> <http://aka.ms/only15min>

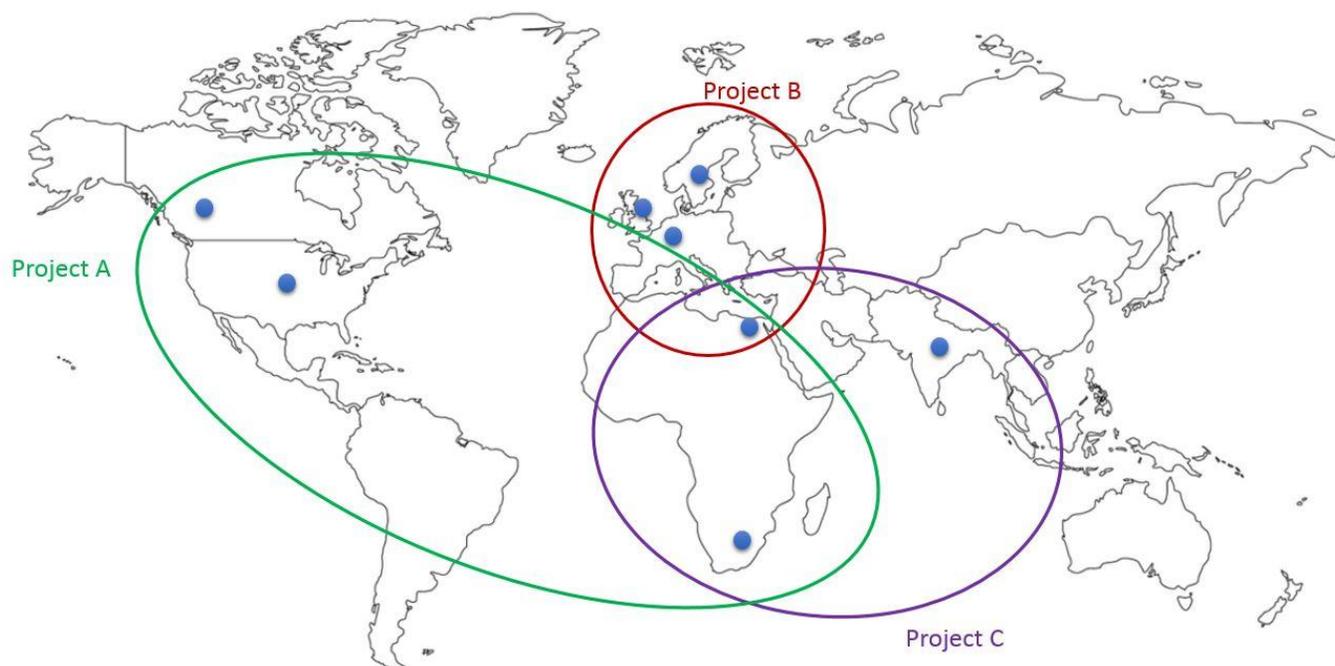


Figure 44. Overlap of projects and team members.

We normalize team velocity across teams, ensuring that the estimation process and its principles are the same for project A, B, and C, as shown above. The two hypothetical resources working on multiple teams and projects work with a consistent process and a relative story-point integer value, representing complexity, the amount of features, and clarity or uncertainty assurance.

### Loose coupling and tight cohesion

#### GEM

#### **Reduce dependency and enable isolated activities**

- Loosely coupled and highly cohesive stories reduce dependency on other team members and specialized knowledge.
- Reduced dependency enables teams or team members to work in isolation.

When we design interoperable systems, we strive for loose coupling and high cohesion. *“Strive for high cohesion and low coupling. The assignment of the correct responsibilities to the correct classes ensures high cohesion. Limiting knowledge or reliance from one class to another class, where the interface of implementation is likely to change, ensures loose coupling”* (Pereira & Schaub, 2006).

If we replace class with story in this quotation, we get a good definition for stories that are suitable for teams scattered across locations and especially time zones.

With loosely coupled stories, we increase versatility and reduce the need for specialized knowledge to answer the “how” questions. If, for example, a team member is in Vancouver and working on a story that requires specialized knowledge from a team member in Johannesburg, the latency of asking a question and receiving feedback can range from minutes to hours as illustrated in Figure 45.



Figure 45. Latency introduced by time zones.

1. A question raised at 11:00 in Vancouver could be acted on within minutes if the team member in South Africa is available at 20:00.
2. A question raised at 19:00 could be acted on within 14 hours if the normal and part-time/community hours overlap.
3. If the times do not overlap, we will incur an additional 8 hours of latency before a team member picks up an answer.

Worse, if the team member in Johannesburg is part-time, the response latency could increase from hours to days. By reducing the need for knowledge or reliance on others, we create loosely coupled stories that any team member can pick up and process.

Stories that have a high cohesion have clear responsibilities to answer the “what” questions. Unfortunately, there is no perfect story in terms of coupling and cohesion other than asking yourself, “*Could this story be done effectively in isolation? If not, can we break it down further?*”

Let us not forget states!

As the stakeholders form consensus around the value and details of features and stories, we must ensure that we maintain the state as outlined in Table 9, “[Important story \(PBI\) events when using Kanban.](#)” The team should only consider features and stories that the Product Owner (PO) marks as Approved.

The Approved state implies that the PO speaks or thinks favorably of the item and is in favor of the team investing bandwidth to implement the added value.

Why should we care about prioritization?

The art of prioritization is not an easy one. Users want all of the features yesterday, and as soon as we involve more than one user with the planning, the relative priorities of features become blurred and debated. The values of features are often difficult to quantify, especially for research, prototyping, dogfooding, and bleeding-edge technologies initiatives.

The Agile Manifesto states that we should collaborate with the user, deliver working software, and react to change. This mandates that we cannot deliver everything, and the reality of time disqualifies prioritizing features for yesterday.

We respect the fixed Roadmap and Resources of the tradeoff triangle discussed in “[Planning the kickoff](#),” leaving us with the prioritization of features. The objective is to repetitively ship as many features as possible, increase the return on investment (ROI), and decrease the [cost of delay](#).<sup>53</sup>

It is common with volunteer and part-time teams that family and “real” jobs interfere with moonlighting projects. It negatively affects the Resources side of the tradeoff triangle, forcing us to reprioritize features continuously.

It is more intuitive and easier to reprioritize small and high-value stories. Working with large, complex, and tightly coupled stories is a disaster waiting to happen, resulting in failure to deliver value, dissatisfaction, and a rapid decline of return of investment (ROI).

<sup>53</sup> [http://en.wikipedia.org/wiki/Cost\\_of\\_delay](http://en.wikipedia.org/wiki/Cost_of_delay)

## GEM

**Ship value quickly!**

- Ship working solutions as soon as possible. Focus on the smallest stories first, and minimize the cost of delay!
- Ship value to the user as soon as possible. Focus on stories that are of high value to the user, not you!

Use common sense and solid prioritization methods to balance these two themes, which often clash.

## Common prioritization methods

Having the Product Owner prioritize the features and guide the team in terms of the story prioritization is a lightweight method that works well for small and part-time based teams, common with our projects.

If we need a more accurate and unbiased prioritization, there are two common methods for planning and prioritization to consider:

- The shortest job first (SJF) method (Wikipedia, 2014) is useful when the cost of delay is equal and recommends doing the smallest story first (Figure 46).

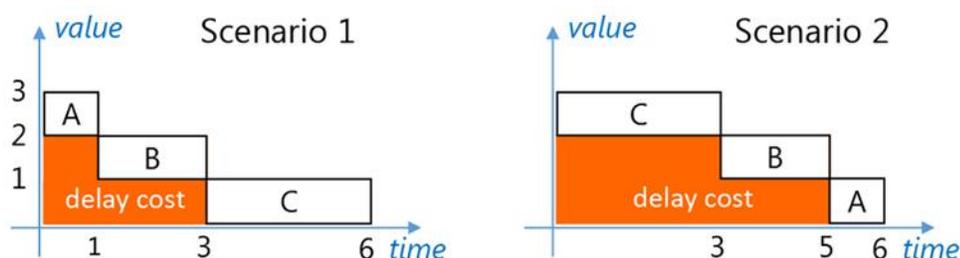


Figure 46. Shortest job first.

Scenario 1 shows the optimal sequence with the least cost of delay. Scenario 2 not only has a higher cost of delay but a higher risk for the smaller tasks to lose their value over time. Most importantly, scenario 2 will make the user wait longer for shippable increments.

- The weighted shortest job first (WSJF) method (Leffingwell, 2014b) is useful when stories have different costs of delay and required effort, which is common. It involves measures for User Value, Time Value, Risk Reduction/Opportunity Enablement Value, and Job Size and is defined by the following formula:

$$\text{WSJF} = (\text{User Value} + \text{Time Value} + \text{RROE Value}) / \text{Job Size}$$

where RROE stands for Risk Reduction/Opportunity Enablement and ranges from 1 to 10.

## Release planning: Offline preparations

In an ideal world, we would schedule and do the release planning in colocated and face-to-face workshops and breakouts. We find that the most efficient way is to ensure that we align all teams and ensure they are ready, that everyone knows exactly what the team expects, how to do it, and when. Most importantly, everyone must know and agree with “the plan” for the solution release and success.

In a typical part-time, geographically distributed, and volunteer-based community, the dream of face-to-face communication is but just a dream. Collaboration is typically limited to email, forums, and teleconferencing events, which requires effective and accurate communication to avoid communication inefficiencies.

The circumstances will vary, but the concepts covered here are based on extensive dogfooding and our “worst case” projects, ranging from 1–2 members to 1–2 feature teams, each with 5–10 members, as well as other stakeholders.

After the kickoff event, we start by reinforcing the key information with all team members and engaging everyone with the planning preparations. It is important that everyone is encouraged to contribute, that everyone gets a feel of involvement and influence, and that we integrate training and research activities effectively with the preparations for the planning event.

Always respect family time: remember that volunteer time is precious and that availability and bandwidth are limited!

Table 7 summarizes the information and offers a template for the offline collaboration thread:

Topic	Description	Owners
Motivation	Reinforce why the solution is important and why it is important for the team to care.	Product Owner (PO) Project Lead (PL)
Vision	Reinforce what the solution we intend to create solves, outlining architectures, technologies, and core deliverables and highlighting the key benefits.	Product Owner (PO) Project Lead (PL)
Roadmap	Reinforce when the solution teams need to achieve key milestones.	Product Owner (PO) Project Lead (PL)
Objectives	Reinforce the Specific, Measurable, Achievable, Realistic and Time-based objectives for the solution release and individual sprints.	Product Owner (PO) Project Lead (PL)
Nonfunctional requirements (NFR)	Mention all known NFRs, such as scalability, reliability, quality, etc., that we need to consider during the planning.	Product Owner (PO) Project Lead (PL)
Features	Reinforce the idea and the prioritized features, which the team needs to expand to stories and optionally tasks during the planning. The Product Owner should start approving all features that must be included in the solution release.	Product Owner (PO) Project Lead (PL)
Process	Reinforce the process and associated frameworks used by the solution team(s), ensuring that every team member knows where to find process guidance and documentation.	Project Lead (PL) Scrum Master (SM)
ASKs	Summarize the key ASKs from the team that need to emerge from the offline collaboration. Limit it to seven (7) ASKs max!	Project Lead (PL) Scrum Master (SM)
URLs	Summarize the key URLs that point at the team's home page, kickoff meeting recording(s), documentation, source control, and other artifacts.	Project Lead (PL) Scrum Master (SM)

Table 7. Release planning agenda showing offline collaboration preparations.

Optionally, consider consolidating the information, sharing it as a poster or other visual aid in a place accessible to the team (Figure 47 and Figure 48).

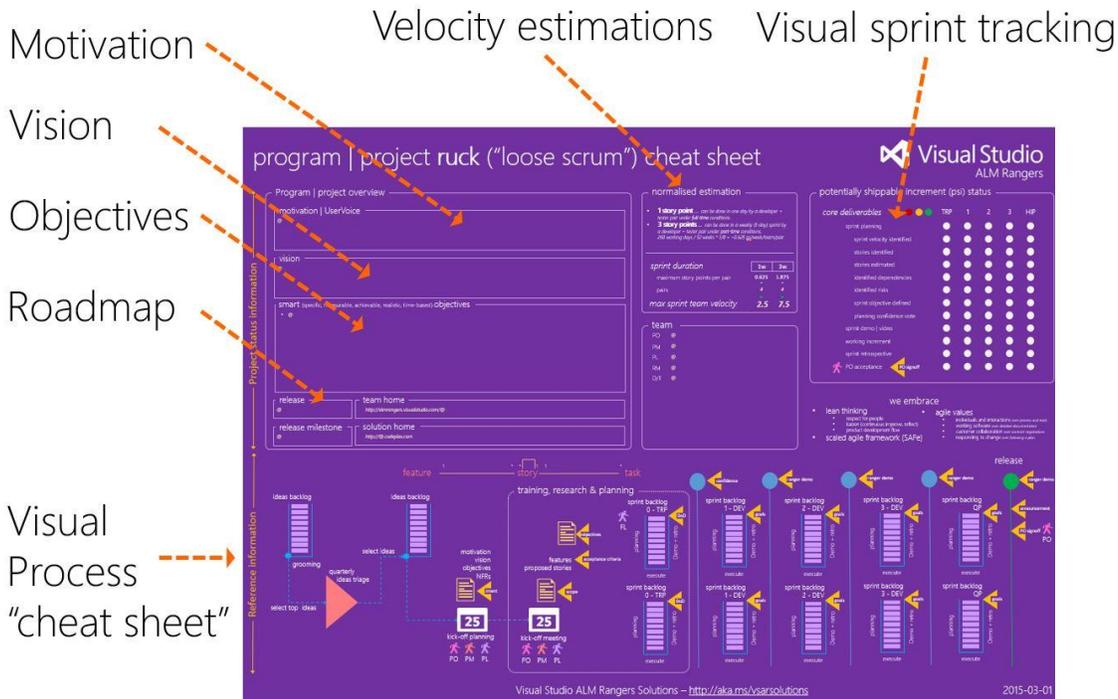


Figure 47. Program | project “all on one” cheat sheet example.

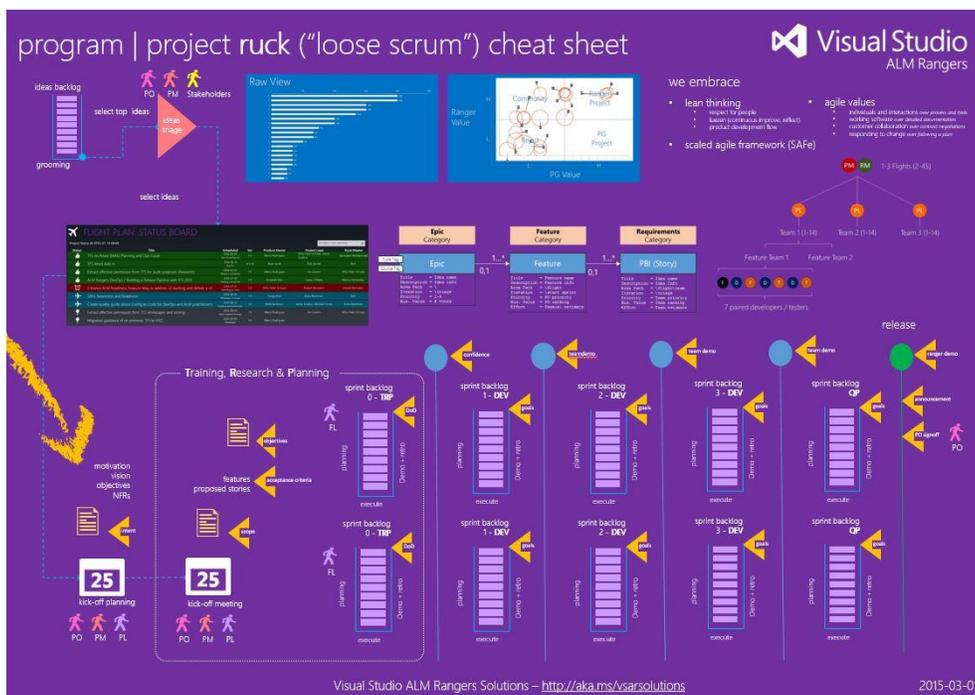


Figure 48. Program | project “separate reference” cheat sheet example.

If the all-on-one quick-reference sheet is too much to present on one poster, we have the alternate version that uses the *A3 problem solving—applying Lean thinking* (Flinchbaugh, 2012) visualization style, as shown in Figure 49.

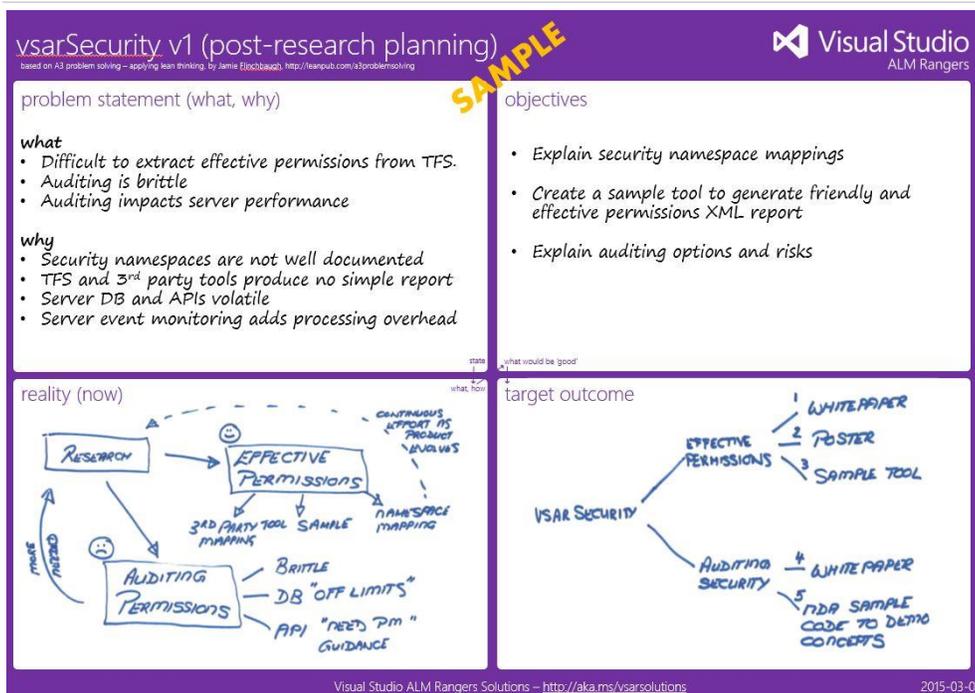


Figure 49. Program | project “separate planning” cheat sheet example

It is imperative that the team collaborates passionately and transparently, typically by using a combination of email threads and teleconferencing calls. During this phase, we collaborate with all stakeholders and iterate through the following phases:

- Understand the features and the acceptance criteria.
- Brainstorm the features, identify risks and dependencies, and identify candidate stories.
- Flesh out each story, ensuring that the title, description, and acceptance criteria are self-explanatory.
  - Each story must describe what the solution must do for the business or architecture.
  - Title template: *As a <persona>, I want <activity>, so that <business value>*
  - See “[General templates](#)” for an acceptance criteria example.
  - Verify that we link each story to a feature parent, the correct area path, and an iteration path, which we use to identify triage candidates—for example, VisualStudio.ALM\Triage.
  - As with features, the [INVEST](#)<sup>54</sup> acronym is useful for stories in this context.
- Flesh out one or more test cases for each story to validate the acceptance criteria and nonfunctional requirements.
  - Verify that we link each test case to a story parent.
- Optionally, estimate each story and identify to which sprint it could be assigned.
- Verify that the Product Owner (PO) is in agreement with the stories and associated test cases.

The interactive discussions and investigations and the effort invested in this preparation will improve the planning experience and the accuracy of estimations. Ensure that all discussions are transparent and that status, visualizations, recordings, and journals are accessible to all team members.

<sup>54</sup> [http://en.wikipedia.org/wiki/INVEST\\_\(mnemonic\)](http://en.wikipedia.org/wiki/INVEST_(mnemonic))

## Release planning: virtualFace-to-virtualFace (vFace-to-vFace)

### Plan the meeting

As before, in an ideal world we would schedule and do the release planning in colocated and face-to-face workshops and breakouts. Instead, we plan and host a teleconferencing vFace-to-vFace meeting, which is long enough to get the core planning done in a professional way, yet short enough so as not to encroach on too much family time.

The optimal duration for a vFace-to-vFace sprint planning meeting is 45 minutes to a maximum of 60 minutes. We usually schedule a morning and an afternoon repeat session, enabling team members to pick the most suitable time slot for their time zone. We record and share the recordings for all sessions, especially if we run repeat sessions or if there are team members who cannot attend the sessions. In the distributed and virtual world, transparency and access to information is key!

#### GEM

#### 45–60x2 meetings

Schedule a 45 minute (recommended) to 60 minute (maximum) planning meeting.

Schedule both a morning and an afternoon session to allow team members in different time zones to select the most productive slot.

Plan the planning meeting ahead of time, giving team members at least one week to accept the invitation and digest the agenda. See the [planning meeting example email invite](#), which defines the intent of the meeting and contains the context and scope.

### Planning meeting

The planning meeting during the sprint(s) that are focused on training-research-planning is usually the longest and most critical vFace-to-vFace collaboration event during the solution flight. The success of the session and the flight is dependent on the offline planning deliverables. If the motivation, vision, objectives, features, and associated stories are accurate, comprehensive, and the result of a consolidated team effort, the subsequent sprint planning is straightforward.

The phase of intense and interactive vFace-to-vFace and face-to-face discussions with stakeholders, Product Owners, Project Leads, Program Managers, and other teams draws to an end. The last step of planning focuses on a discussion of how and when we implement and potentially release the features. Optimally, invite the Product Owner and other stakeholders to participate as “flies on the wall” and assist the team with clarifications or recommendations when asked. While transparency is key and input welcome, it is now time to trust the team and enable its members to create a plan that they feel comfortable with and confident about delivering.

We frown upon changes in scope or requirements at this point. The facilitator should discourage such changes, and the Project Lead should enforce these decisions.

#### GEM

#### Proactive planning “quick” survey

For teams that are reliant on part-time volunteers and geographically dispersed, we’ve found that online proactive planning surveys help the team with preparations and getting a rough estimation. See the [planning survey template: Planning online](#) for an example.

Table 8 summarizes the information and offers a template for the vFace:vFace planning meeting:

Topic	Description	Owners
Overview	Reiterate the release motivation, vision, roadmap, and objectives.	Product Owner (PO) Project Lead (PL)

Topic	Description	Owners
Estimate	<ul style="list-style-type: none"> <li>Identify a story that the team can use as a one story point reference point.</li> <li>Estimate all stories as follows: <ul style="list-style-type: none"> <li>Brainstorm briefly to cement the understanding <ul style="list-style-type: none"> <li>Estimate size relative to the reference story.</li> <li><b>Park</b> a story if unable to reach an agreement or if size does not fall within the maximum size or velocity estimate.</li> <li>Prioritize based on dependencies and importance.</li> </ul> </li> <li>Divide parked stories that are too large into smaller stories and reestimate each.</li> <li>Brainstorm and reestimate all remaining stories until there are none left or there is a <b>stalemate</b>. Postpone the latter and resolve offline to respect the team's time.</li> </ul> </li> </ul>	Team
Outline sprints	<ul style="list-style-type: none"> <li>Assign stories to the first development sprint and optionally to subsequent sprints until the sprint velocity is about to be exceeded.</li> <li>Remember that not all teams can hit the ground running, that not all teams can deliver the estimated velocity, and leave a bit of breathing (slippage) room where possible.</li> <li>Next+1 (subsequent) sprints are rough estimates and will be refined at the next planning session.</li> </ul>	Team
Confidence	<ul style="list-style-type: none"> <li>Determine a confidence vote for the next sprint and the solution release overall</li> <li>Ask the team to vote at the end of the planning session to determine their confidence of completing the next sprint and "shipping" the solution.</li> <li>Scale 0 – 5, where 0 = ☹ and 5 = ☺</li> </ul> <p>If we are face-to-face we can use raised hands and number of fingers to vote. For vFace-to-vFace confidence vote we can use a simple Lync poll or the equivalent.</p>	Team
Innovation	<p>Agree on at least one innovation that the team can consider for the next sprint to complete the planning session.</p> <p>Continue to discuss and agree offline on what went well and what went badly, preferably by including other teams' Scrum Masters to share innovations and learn as a community.</p>	Team
URLs	Reiterate the key URLs that point at the team's home page, team artifacts, and support.	Project Lead (PL)

Table 8. Release planning agenda for the planning meeting.

## Schedule the infamous worldwide scrums

Working with different personas and in different time zones, a distributed and part-time team sprinkles interesting challenges over the scheduling for the Program Manager (PM) and the Scrum Master (SM). The PM or SM typically support the Project Lead (PL) by owning and driving this challenge.

### POP QUIZ

*How do we schedule a meeting that allows all of our team members to meet for an hour, at a reasonable time, and that we can repeat for weekly scrums? Assume person 1 is on the West Coast of the United States, 2 on the East Coast, 3 in Ecuador, 4 in Switzerland, 5 in South Africa, 6 in Pakistan, 7 in New Zealand, and 8 in Australia (Figure 50).*



Figure 50. Pop quiz: Many time zones, one team.

*If we schedule the meeting on Monday at 08:00 AM PDT in Vancouver, we would have other team members attending on Monday at 10:00, 11:00, 17:00, and 20:00, as well as Tuesday at 01:00 AM and 03:00 AM.*

The challenge of scheduling a recurring scrum meeting that respects people's time and is productive should be clear from this simple example.

At the risk of sounding like a broken record, in an ideal world we would meet face-to-face each day for a few minutes for the daily scrum. However, in the distributed, part-time, and volunteer-based community ecosystem, the recommendation is to schedule a weekly "heartbeat" scrum, which is at least 15 minutes long and can optionally extend to 30 minutes if needed.

The intent of the scrum is to share status information, discuss the "state of the nation," address issues and impediments before they turn into raging fires, and manage scope and derivations thereof. Attendance includes the Scrum Master (SM), Project Lead (PL), the team, and optionally the Product Owner (PO) and Program Manager (PM).

## Summary of our process and requirement rudiments

As part of the sprint(s) focused on training-research-planning, we cover a huge and strategic area of the solution requirements. It all starts as a strategic idea, defined by a handful of features and realized by stories. This is a great opportunity to reflect on the overall process and on the requirement rudiments.

1. Epic represent the ideas raised by the community and stakeholders to realize value propositions. Ideas can originate from initiatives, investment themes, product themes, or simply as a great idea.
  - We capture ideas in the ideas backlog by using the Epic work-item type and triage them during the [triage of ideas](#).
  - Epics convey an idea expressed in a formal or bulleted form, and the implementation detail is contained in their dependent features and stories.
  - We do not implicitly implement or test Epics. Instead, we fulfill Epics by features, which we in turn realize by stories.
  - Epics are not tested but validated by the test cases associated with their dependent features and stories.
2. Features emanate from the core idea. As part of the idea triage, the Product Owner identifies and prioritizes a few features for the team to invest in during the next and future releases of the solution.
  - We capture features on the ideas backlog and link them to the parental idea as dependent child objects.
  - We assign features to a solution “flight” by using the area path and a solution “release” by using the iteration path on VSO.
  - Features are of type business objective, architecture objective, process objective, or other custom objectives. The business and architecture objectives are common in SAFe™ or CMMI, identifying architectural infrastructure, business requirements, or user experiences.
3. Stories describe what the solution must do for the business or architecture to realize the encompassing feature. As part of planning, the team identifies, estimates, and prioritizes a few stories, which may optionally be broken down into Task objects.
  - We capture stories on the sprint backlog and link them to the parental features as dependent child objects.
  - We assign stories to a team by using the area path and an iteration or sprint by using the iteration path on VSO.
  - Stories are tracked using the Kanban board or as dependent tasks using the task board.
  - Stories are typically of the same type as their parent. In some cases a business objective, for example, may require business features and innovations to an existing architecture or infrastructure, in which case there could be a mix of story types realizing a feature.
4. Teams represent the passionate volunteers who take ownership of the solution flight, self-organize as feature teams, and innovate their ecosystem through regular reflection.
  - We have one engineering team that owns the operational infrastructure, program management, deployment, and cross-flight collaboration. We organize each solution flight into one or two focused feature teams—for example, the guidance feature team and tooling feature team.
  - Teams are typically short-lived for the duration of a flight and release of a solution. A nucleus of one or both feature teams will evolve into the maintenance and ownership team, which accompanies a solution after release and will kick-start future feature teams with subsequent flights.
  - Long-lived nuclei ensure knowledge retention, increased learning, more accurate estimating, better planning, and less ramp-up waste.

Figure 51 shows the first phases of the overall solution release (flight) and the hierarchy of Epic-Feature-Story and associated work-item types.

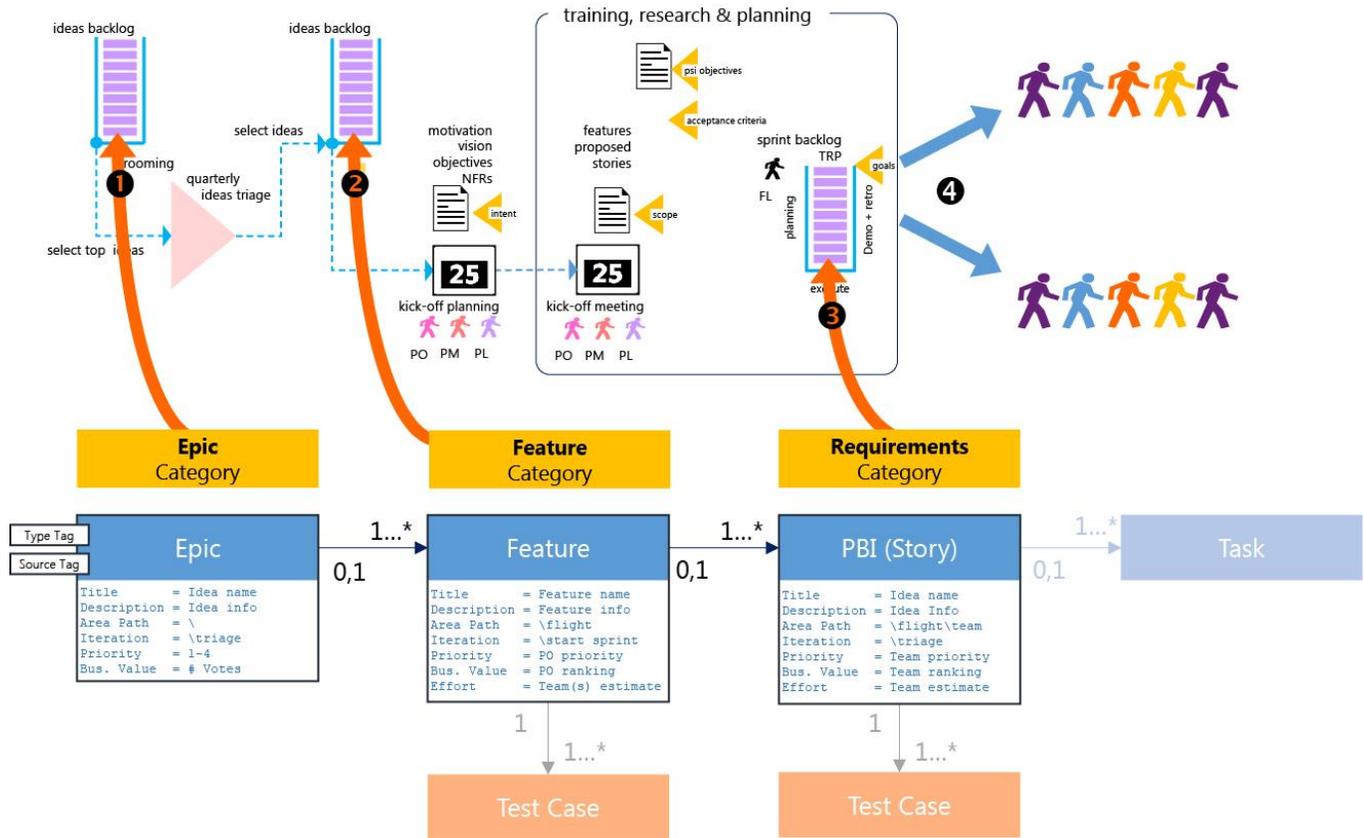


Figure 51. Epics realized by features realized by stories and optionally tasks.

Using Team Foundation Server (TFS), we may visualize an idea and sprint backlog as shown in Figure 52.

Backlog items to Epics

Backlog Board

Mapping Off View Backlog items to Epics

Work Item Type	Title	State	Effort	Iteration Path	Area Path	Requirement Type	Tags
Epic	Innovate standard process templates	In Progress		ProcessDemov2	ProcessDemov2	Business Objective	SAFe Innovation
Feature	Apply innovations to Scrum process template	In Progress		ProcessDemov2\Release 1	ProcessDemov2\SAFe Innovation	Business Objective	
Product Backlog Item	Implement process template customizations	Committed	3	ProcessDemov2\Release 1\Sprint 1	ProcessDemov2\SAFe Innovation\Feature Team	Business Objective	
Product Backlog Item	Document customizations as a walkthrough	Committed	2	ProcessDemov2\Release 1\Sprint 1	ProcessDemov2\SAFe Innovation\Feature Team	Business Objective	
Product Backlog Item	Create a context whitepaper	New	21	ProcessDemov2\Release 1\Sprint 1	ProcessDemov2\SAFe Innovation\Feature Team	Business Objective	
Product Backlog Item	Automate the customization process	New	5	ProcessDemov2\Release 1\Sprint 1	ProcessDemov2\SAFe Innovation\System Team	Architecture	

Figure 52. Example work-item hierarchy using Scrum process template.

## Glimpse of tomorrow . . . tracking with an informative board

**TOOLING** [NEWS extension reminded me to get excited about the latest VSO planning features](#)<sup>55</sup>

Five years ago, the typical duration of one of our projects was 6 to 12 months, with one to two deliverables. Spending days planning and monitoring progress was a feasible and valuable investment. Projects were broken down into detailed tasks tracked using the task board.

<sup>55</sup> <http://aka.ms/b0vwmd>

Today the typical duration of a project is three to four three-week sprints, with a potentially shippable deliverable after every sprint. Although some team members are still breaking down their work into actionable and detailed tasks, frequent anomalies we observe include:

- Teams do not use or track tasks, and task boards are seldom used.
- Features marked as Done often contain orphaned tasks that are still in the new or in-progress state.
- Information documented in tasks—for example, objectives and the definition of done (DoD)—is difficult to find.

Reflection, continuous improvement, a willingness to take risks and embrace change and new features on Visual Studio Online are instrumental to our dogfooding and productivity. This is our tomorrow:

1. New projects use one informative Kanban board (Figure 53) to eliminate waste and monitor and visualize all the work.

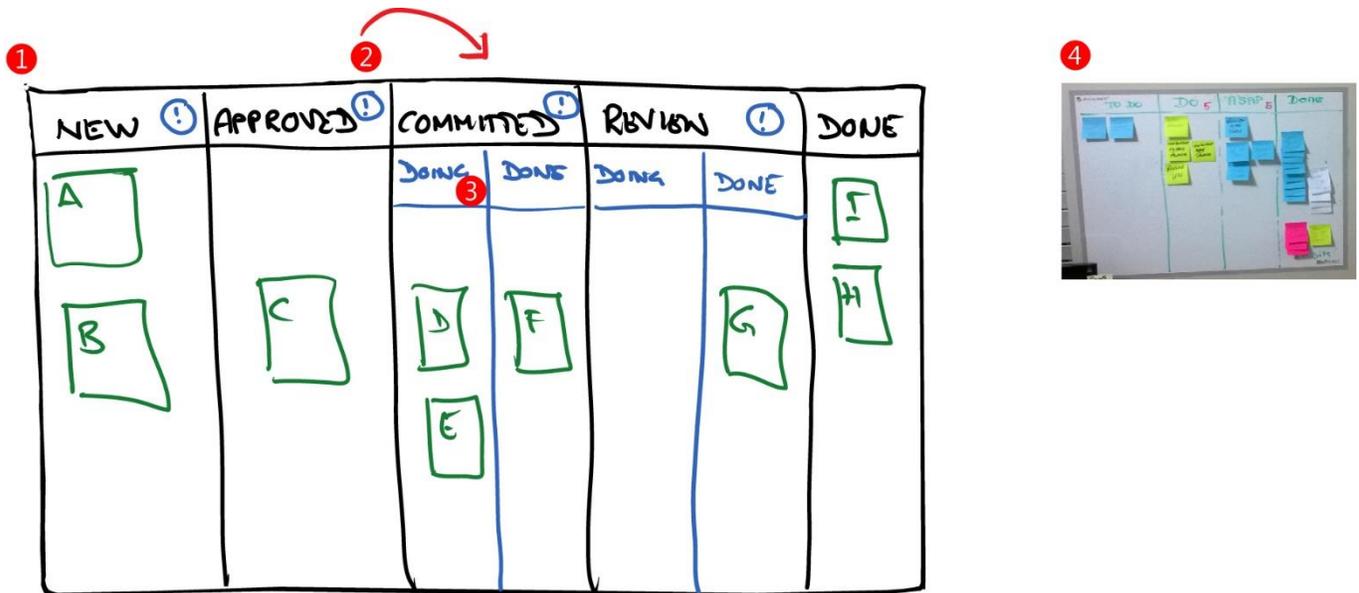


Figure 53. Using the Kanban board of tomorrow.

2. The definition of done (DoD), acceptance criteria, and other relevant information driving state is accessible from the board.
3. Split columns visualize work through our workflow.
4. Personal Kanban boards and sticky notes replace the use of detailed task planning and boards.

## Dogfooding case study: Where is the fire?

Estimating, managing, and tracking requirements with geographically distributed, virtual, and part-time teams are parts of a complex adventure. In this case study, we explore evidence from a few of the previous teams' observations that have influenced solutions.

### Evidence from the field

#### Scenario 1—Burning down

Teams trend below the ideal burndown line, giving most observers a sense of comfort. What raises concern are the three spikes (see Figure 54), typically caused by scope creep or acceptance of new tasks or bugs. This could create a discrepancy between planned velocity and actual velocity.



Figure 54. Scenario ambition.

Teams agree to ambitious estimations, adding large-size stories and tasks to their sprint backlog. The frequent result is a failure to deliver working solutions at the end of sprints and a trend to “carry over” work to the next sprint (Figure 55).



Figure 55. Death march.

Teams start energetically and then deflate and awaken toward the end for the usual death march to meet expectations—or simply wait for the last possible moment to deliver. The result includes frustrations among team members, especially where there are dependencies, and a frequent cancellation of features (Figure 56).



Figure 56. Inadequate planning, tracking, and reflection.

- Teams fail to invest the time needed to plan each sprint or perform microtask planning, leading to inaccurate planning, inconsistent sprint velocities, and overhead in terms of task management during the sprint.
- Team members often report status late—“Oh, I completed that work a long time ago”—resulting in a sudden drop of the burndown toward the end.
- Starting with no clear objectives or milestones, projects gradually turn into an on-going rut. Teams see retrospectives as overhead rather than as an opportunity to improve the team environment, and teams often abandoned retrospectives altogether.

**GEM****Proactive retrospective “quick” survey**

For teams that are reliant on part-time volunteers and geographically dispersed, we’ve found that online proactive retrospective surveys help the teams gather the good, bad, and ugly very quickly. We can use the survey feedback to fast track the retrospective discussions.

## Training-research-planning (TRP) as a remedy?

The focus on training-research-planning encourages teams to practice proactive sprint planning, reducing ambiguity and task management. With realistic and accurate estimations, teams are able to deliver working software frequently and regularly. Over time, a realistic velocity can be determined for each team and normalized across the community. Agreeing on one innovation each sprint reduces the perceived waste of time due to meetings but continuously encourages reflection and innovation.

## Key learnings

- The focus on training-research-planning helps the team to educate themselves and understand and define the what, why, and how of the underlying solution.
- Estimations play a huge role in projects, especially normalized estimations. Normalized estimations help to simplify the estimation effort and improve estimation accuracy. This in turn helps the team to deliver working software frequently and regularly with continuous improvements.
- Recorded vFace-to-vFace meetings provide transparency and access to information.

# Chapter 3: Building the working solutions

*Never tell people how to do things. Tell them what to do and they will surprise you with their ingenuity.*  
—George S. Patton

In the previous chapters, we introduced the different phases of planning and for preparing ideas and team backlogs containing the hierarchy of Epics, Features, and Stories. After careful planning, it is now time to enter the phase where the rubber meets the road—executing our plan, implementing features and associated value-adds, and shipping working solutions quickly and often (Figure 57).

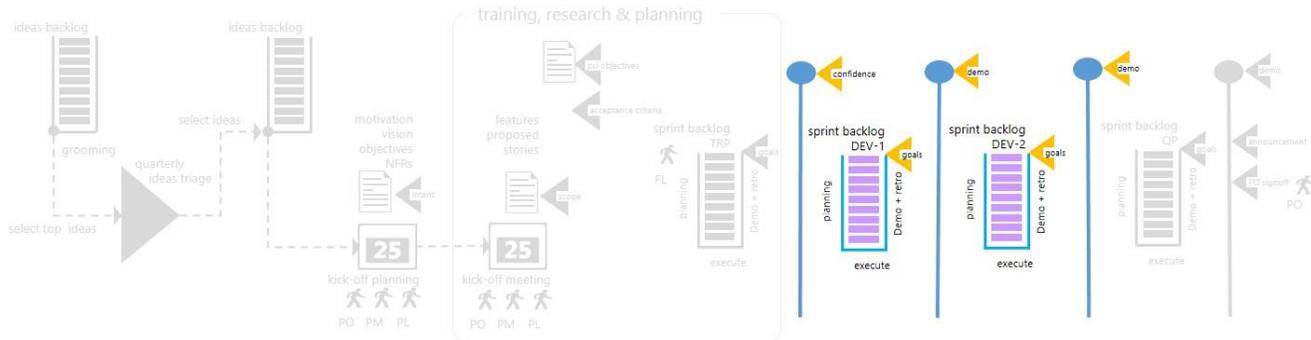


Figure 57. Lifecycle: Development.

## Development (DEV) sprints

The heartbeat and pulse of an Agile team is an iterative and consistent cadence from one development sprint to the next and from one scrum meeting to the next within a sprint. In this section we explore how geographically distributed teams cope with the need for consistency and iteration. As shown in Figure 58, the execution phase is now focused on the development, testing, and validation tasks.

## Execution phase

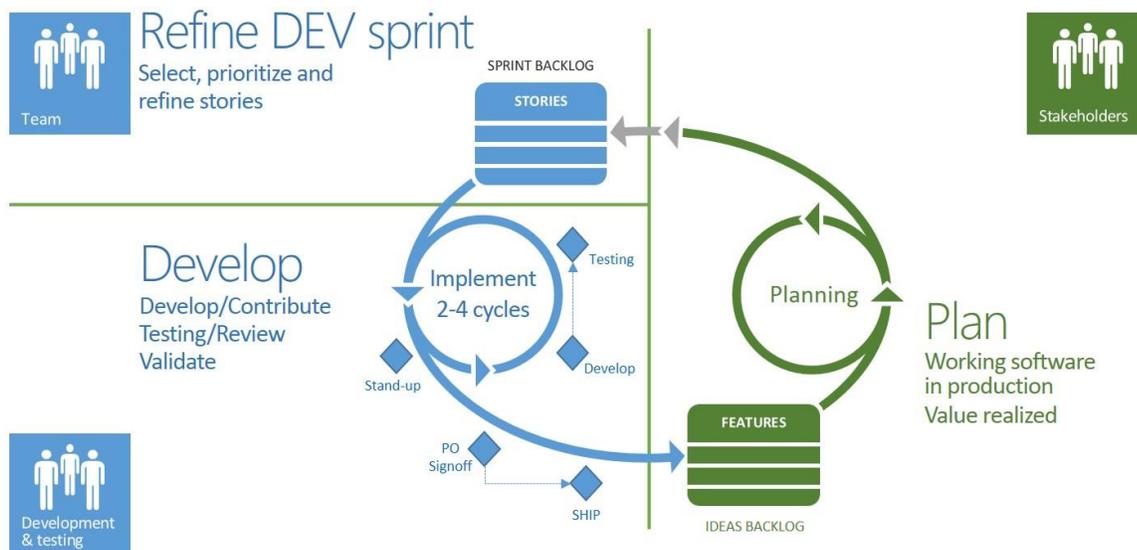


Figure 58. Development iterations/sprints.

## Team realizes features with stories

During the [triage of ideas](#), the focus is on selecting the ideas with the most business and strategic value, represented by an Epic work-item type.

The ideas are realized by features, which are identified, rated, estimated, and refined during [planning the kickoff](#). Additionally, each feature is broken down into stories using the product backlog item (PBI) WIT, resulting in a hierarchy as shown in Figure 59.

ID	Work Item Type	Title	State	Effort	Iteration Path	Area Path	Requirement Type
1	Epic	Innovate standard process templates	In Progress		ProcessDemov2	ProcessDemov2	Business Objective
4	Feature	Apply innovations to Scrum process template	In Progress		ProcessDemov2;Program\Release 1	ProcessDemov2;SAFe Innovation	Business Objective
5	Product Backlog Item	Implement process template customizations	Committed	3	ProcessDemov2;Program\Release 1;Sprint 1	ProcessDemov2;SAFe Innovation;Feature Team	Business Objective
6	Product Backlog Item	Document customizations as a walkthrough	Committed	2	ProcessDemov2;Program\Release 1;Sprint 1	ProcessDemov2;SAFe Innovation;Feature Team	Business Objective
7	Product Backlog Item	Create a context whitepaper	Approved	21	ProcessDemov2;Program\Release 1;Sprint 1	ProcessDemov2;SAFe Innovation;Feature Team	Business Objective
8	Product Backlog Item	Automate the customization process	New	5	ProcessDemov2;Program\Release 1;Sprint 1	ProcessDemov2;SAFe Innovation;System Team	Architecture
12	Bug	Type values need to be updated to reflect latest design	Approved	1	ProcessDemov2;Program\Release 1;Sprint 1	ProcessDemov2;SAFe Innovation;Feature Team	

Figure 59. Epic and features ideas backlog.

### NOTE Epic vs. Feature + Tag

At the time of writing, the Epic work-item type (WIT) was not available on Visual Studio Online (VSO). We used the Feature WIT with a special Epic tag to represent Epics on VSO and were dogfooding the Epic WIT in a [TFS on Azure IaaS](#) proof-of-concept environment. The screen shots in this section are from the proof-of-concept environment, not VSO.

Peeking into the entire backlog, which contains the ideas and sprint backlogs, we find an extensive hierarchical requirements model as shown in Figure 60. Recent sections introduced the origin and evolution of each requirements object (Epic, Feature, and Story). It is now time to take a more in-depth look at the backlog and focus on the development of the requirements as defined by stories, the validation and testing of stories as defined by test cases, and tracking issues or bugs.

### TOOLING [Testing Tool—Test Management](#)<sup>56</sup>

<sup>56</sup> <https://www.visualstudio.com/en-us/explore/testing-tools-vs>

Entire Backlog 12 work items (1 top level, 11 linked and 1 selected)

Results Editor Charts Work item pane Bottom

Save query [Refresh] [Undo] [Redo] [Share] [Print] [Email] [Columns] [Column options] [Filter]

ID	Work Item...	Title	Assigned To	State	Effort	Busin...
1	Epic	Innovate standard process templates	Gregg Boer	In Progress		13
2	Feature	Apply innovations to CMMI process template		New		13
3	Feature	Apply innovations to Agile process template		New		13
4	Feature	Apply innovations to Scrum process template	Willy-Peter...	In Progress		13
5	Product Ba...	Implement process template customizations	Gordon Be...	Committed	3	13
6	Product Ba...	Document customizations as a walkthrough	Gordon Be...	Committed	2	13
9	Test Case	Validate CMMI process template customization		Design		
10	Test Case	Validate Agile process template customization		Design		
11	Test Case	Validate Scrum process template customization		Design		
7	Product Ba...	Create a context whitepaper	Gregg Boer	Approved	21	13
8	Product Ba...	Automate the customization process		New	5	8
12	Bug	Type values need to be updated to reflect latest design	Gordon Be...	Approved	1	

Figure 60. The full Epic, Features, and Stories backlog.

During the development phase, the team typically focuses on the sprint backlog, which is a more focused view of the overall backlog, containing stories and bugs, as illustrated by the proof-of-concept backlog shown in Figure 61.

### ProcessDemov2 Team Sprint 1

Backlog Board Capacity

Create query | Column options | [Email]

ID	Title	State	Assi...	Rema...	Busin...	Effort
5	Implement process template customizations	Committed	Go...		13	3
6	Document customizations as a walkthrough	Committed	Go...		13	2
7	Create a context whitepaper	Approved	Gr...		13	21
8	Automate the customization process	New			8	5
12	Type values need to be updated to reflect latest design	Approved	Go...			1

Figure 61. Features and stories sprint backlog.

## Running through the sprint

### Pushing or pulling stories to team members?

With mature teams, stories and associated tasks are pulled from the backlog. Just as with the planning of the stories, allowing team members to commit to stories by taking (pulling) them from the backlog creates a sense of ownership and independence.

With a geographically distributed team, however, we mix a variation of cultures, skills, and levels of commitment. Some stories and their “best” owners may need to be reconsidered in the current project context, which may require the team leadership to recommend (push) stories to team members for consideration.

Scenarios where stories are pushed selectively to team members include:

- Team member(s) may prefer to be guided by the team as to which story they need to work on.
- Multiple team members volunteer for a story, and the team needs to decide on the best owner.

GEM

**Pull & push**

Adopt a pull strategy for stories by default, but be ready to propose and push stories to team members who hesitate to grab stories from the backlog.

It is important that the commitment for the stories is made *by* the team, not *for* the team. Team members need a sense of ownership, independence, and freedom, especially when volunteering and working in geographically distributed and remote locations.

## Fixed or variable cadence?

Most Agile teams swear by a one-week, two-week, or three-week sprint. In the real world, the majority of teams appear to be using a two-week iteration.

Key factors that affect the duration of a sprint include:

- The shorter the sprint, the greater the backlog and sprint management tax in comparison with development and testing efforts.
- The longer the sprint, the longer it can take for impediments to surface, feedback to be gathered, and, more importantly, to be in a position to reflect and respond to change.

As discussed in [“One maintainable environment ... simplicity rules,”](#) we highly recommend using consistent and shared iterations across our community. Consistency enables team members to collaborate effectively on multiple solution flights and flip-flop between teams when needed. Consistency also simplifies training, look ups, and reporting across flights and feature teams.

The naming, duration, and breakdown of iterations will vary from organization to organization, possibly even within teams from the same organization. We, for example, used monthly sprints, best suited for part-time resources, but recently aligned with the parent development division by using three-week sprints.

GEM

**Partial and full sprints**

TRP and QP focused sprints are well suited for partial or full sprints, whereas DEV sprints are complete sprints to counterbalance the “part-time” nature of our resource bandwidth.

## Repetition!

Irrespective of the length of the sprint, it is important to adopt a sprinting pattern that is natural to the team. (See Figure 62.) Let’s consider what we would typically take action on if we need to make alterations to our home, fly a passenger plane across the Pacific, or fly to moon Europa:

1. Assume ownership of the sprint challenge as a team, for example, *travel to moon Europa*.
2. Review and refine the objectives, requirements, and features from the sprint backlog, *for example, transfer W astronauts to the moon, collect X tons and Y types of samples, and return to Terra within Z years*. Verify that we understand the requirements, are in a position to meet the acceptance criteria, and are in a position to proceed with none or manageable unknowns.
3. Plan and design the sprint development journey.
4. Develop the features to realize the requirements.
5. Integrate, test, and validate the features.
6. Reflect on the sprint, identify improvements, and plan the next actions.
7. Demo and ship working solution(s).

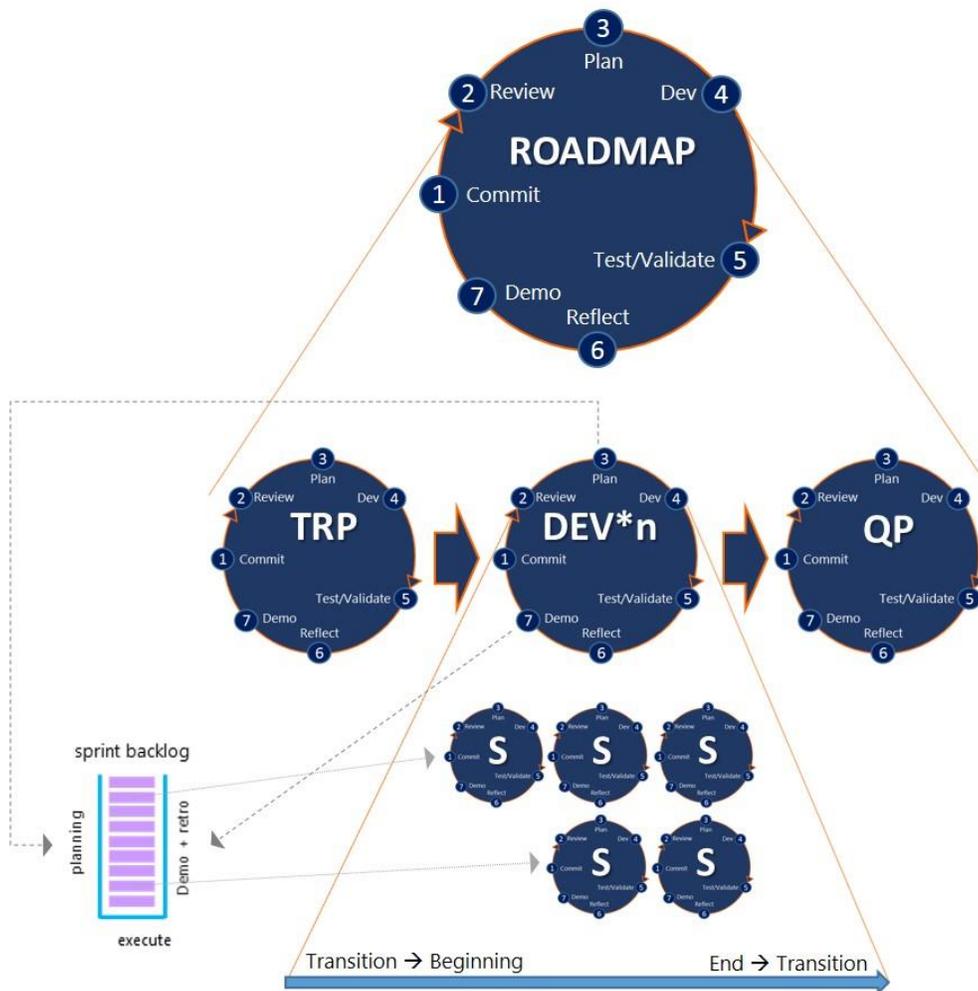


Figure 62. Repetition within and as sprints.

The illustration shows that the roadmap contains the seven actions mentioned and is broken down into an initial focus on TRP, one or more DEV sprints, and a final focus on quality and planning. Each sprint contains the same seven actions. Every iterative lifecycle has a beginning to which we transition from somewhere else and an end from where we transition to somewhere else.

Step 2 is pivotal to the Agile Requirements Management lifecycle. We no longer work with detailed design specifications but with “wish lists” encapsulated in features and stories. The team is responsible for elaborating the requirements for the overall solution in an iterative manner and probably evolving a detailed description and acceptance criteria while iterating through the stories within the sprints.

The Scrum Master (SM), in collaboration with the Project Lead (PL) and Product Owner (PO), has the responsibility of mentoring and guiding the team through this iterative journey. The use of a basic checklist can further assist the consistent iteration through each sprint:

- Align with expectations of the PO and understand the objectives and requirements.
- Review and reprioritize backlog features and/or stories as requirements evolve.
- Verify that all stories are associated with one parent feature and all features with one parent Epic (idea).
- Encourage the team to reflect, innovate, and show what they have done with a sprint and release demo. Seeing is believing!
- Contain scope creep, otherwise the team cannot commit to a schedule or resources as outlined in the [roadmap](#).

The team, the Project Lead, and the Product Owner must negotiate deviations from the original plan and objectives and commitment jointly. The scope must remain static during a sprint, making focus by the team and commitment to the overall flight and sprint feasible. We abort an active sprint if we must adjust the scope before restarting with revised commitments.

## Sprint objectives rule the deliverable

The team commits to a sprint based on resources, features, and objectives negotiated with the stakeholders. At the end of each sprint, we plan the next steps and optionally refine the project roadmap, objectives, features, and resourcing. After a refinement, the team recommits, and thereafter the tradeoff triangle (resource, features, and roadmap) remains inert for the duration of the sprint.

The objectives are not a “big stick” nor “marketing fluff.” The flight/project objectives chiseled on the team dashboard are typically inherited as sprint objectives as a whole or selectively based on the selected stories for each sprint, as shown in Figure 63.



Figure 63. Objective-based sprint backlogs.

The objectives reinforce why we are doing the project, what needs to be done, who will be responsible for what, and how confident the team is about succeeding. We ensure that we display them on dashboards, in communications, and at the start of each scrum to reinforce them.

## (Bi-)weekly scrum

The team has taken responsibility and started development and typically collaborates in a face-to-face, colocated team environment, which encourages the sharing of information and the ad hoc, as needed walk to the whiteboard. In a geographically distributed, part-time, and volunteer-based team environment, as experienced by us, the need to share the status, manage the work queues, take action on the impediments, and view the quality bar becomes paramount.

Team transparency is based on the scrum team meeting, also known as the “heartbeat” or “pulse” of the team. Typically a daily ceremony, this is a weekly (recommended) or biweekly event in our ecosystem, made up of a 15-minute review of status and an optional 15-minute discussion slot.

**GEM****Weekly heartbeats “rock”**

With geographically distributed, part-time, and virtual teams, a weekly scrum, which adheres to a 15[+15] minute time box, is recommended.

Here’s a recommended breakdown and flow of the scrum meeting:

0–15 minutes: status

- Share the overall status with the team, relying on visual information such as the burndown chart.
- The Scrum Master asks the team whether anyone has to go first and then iterates with the rest of the team (for example, alphabetically).
- Every team member shares what was done since the last scrum, what they are and will be doing, and impediments, if any. The Scrum Master ensures the discussions remain within the done/doing/impediments boundaries and makes a note of and defers all other discussions to the optional 15-minute Q&A.
- Review all impediments, both previous and new.

16–30 minutes: Q&A

If there are no topics to discuss, the scrum meeting ends after the status update. During the optional 15-minute Q&A, team members have the option to stay and participate in discussions or to leave the meeting.

What about those who cannot make the scrum?

Team members who cannot make the scrum must notify the team and share their heartbeat (done, doing, impediments), preferably before the scrum but definitely before the poststandup housekeeping is done. Members who fail to show a heartbeat are known as dark zombie suspects!

Postscrum housekeeping

The scrum status, impediments, discussion notes, current backlog, and visual information radiator (dashboard, charts, process/scrum cheat sheet, etc.) are updated and shared with the team and all stakeholders. Remember, transparency is key!

How and where the teams keep the scrum notes is not that important as long as every team member knows where to find the notes. For example, we can use Visual Studio Online and a Task work item to store the scrum notes, associated documentation, and links to other work items. Consider linking the scrum tasks to the Program Manager (PM) bucket backlog product item (PBI), which we introduced in [“Team infrastructure,”](#) and which contains the objectives and the acceptance criteria information. (See Figure 64.)



# TFS on Azure (IAAS)



## vision

Deliver practical guidance and easy-to-consume readiness material to enable the field to educate themselves to plan, implement and maintain an effective TFS on Azure IaaS environment for product evaluation, testing, training and production in the cloud

## deliverables

- Practical guidance (eBook, HOLs, videos)
- TechReady session
- Collaboration with On-Time VM Factory Service

## status

- In-flight
- S21 (Mar) – Silent BETA shipped!
- S22 (Apr) – TFS on Proxy done.
- S23 (May) – Build scenarios and ship!



## motivation

Strategic initiative supporting DevDiv initiatives to enable and support the user to implement a TFS service in their private cloud

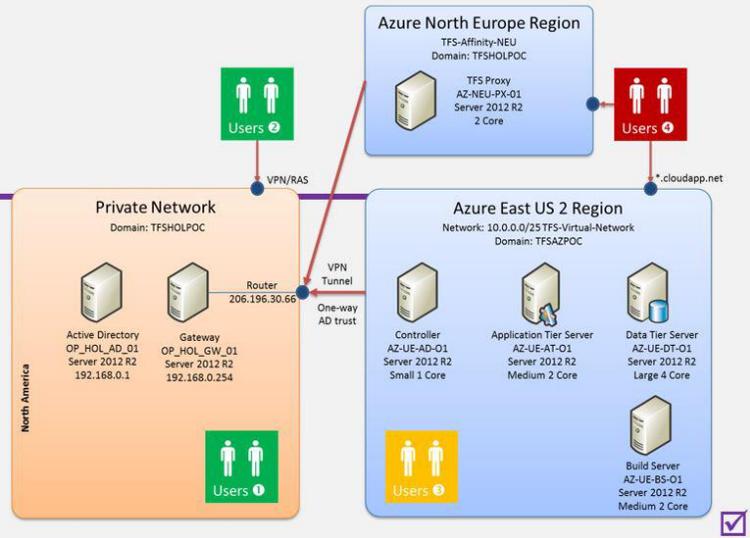


Figure 65. Scrum-of-scrums status slide.

The core objectives of the scrum of scrums are:

- Remind everyone of the flight vision and motivation.
- Present an update on the flight roadmap status (done, doing).
- Raise and discuss impediments.
- Raise and discuss retrospective findings and proposed innovations.
- Raise and discuss dependencies on other flights.
- Share show-what-we-have-done sprint demo videos or a live demonstration.

## TOOLING

- [Integrated Development Environment](#)<sup>57</sup>
- [Create your app](#)<sup>58</sup>
- [Building Azure Web Sites with Visual Studio Online "Monaco"](#)<sup>59</sup>

## Coping to work in isolation

The complexity of working on and leading projects in isolation is a topic of debate. Some argue that the life of a full-time Scrum Master, Project Lead, Product Owner, and Program Manager is far more involved and complex. Some argue that the business value and strategic impact of dedicated projects are far superior to their part-time community counterparts.

<sup>57</sup> <https://www.visualstudio.com/features/ide-vs>

<sup>58</sup> <https://www.visualstudio.com/get-started/create-your-app-vs>

<sup>59</sup> <http://azure.microsoft.com/en-us/documentation/videos/building-web-sites-with-visual-studio-online-monaco/>

## Collaboration and timely sharing of information is key

This guide will not debate these arguments. Instead, we will explore the challenges of sitting in a cockpit and flying a plane compared with guiding a drone on the other side of the planet—or worse, a probe entering the orbit of another planet. Sitting in an actual cockpit allows the pilot to sense the conditions and status of the flight by using sight, hearing, and face-to-face collaboration. While the complexity of flying the plane is not to be taken lightly, the ability to visually experience and interpret the surrounding conditions in real time appears more effective and is a preferred experience to the average person.

In a disconnected scenario, the pilot relies on information transferred back from the remote device. The receipt and analysis of information and the reaction to it is therefore less than optimal. The overall success depends on the accurate and timely sharing of information needed to make critical decisions. Whether a space probe enters the correct orbit, burns up in the atmosphere, or explodes in a fiery impact often depends on split-second decisions based on information received with huge transmission latency.

Similarly, the Project Lead, Product Owner, Program Manager, and Scrum Master all depend on the transparent, accurate, and timely sharing of information and collaboration by the geographically distributed and part-time team members.

Transparency, reflection, and continuous improvement are the core pillars. If we get accurate information on time, if nothing remains hidden, and we have reliable team collaboration, we have the three fundamental pillars of Lean thinking. We will be in a good position to inspect, adapt, and improve, with an understanding that everyone is and will continue to make mistakes. The team rises or falls as one self-organized hive!

Transparency also avoids the common scenarios of the black hole, rude awakenings, and perpetual scope creep. Nightmares we are aware of but avoid:

- **Black hole** Team members vanish off the radar, leaving the leadership and the rest of the team without the information they need to reflect and innovate. If we have team members with the “black hole” syndrome, we transition them to other projects as quickly as possible.
- **Rude awakening** The team iterates without transparency and enjoys a false sense of security. As the team approaches the end of the sprint, it becomes apparent that there will be no demo or working software, but instead a sudden spike of issues and impediments.
- **Perpetual scope creep** The team continuously reacts to scope creep. The result is usually a need to renegotiate scope, quality, and expectations and also reset a sprint or even the release.

We work with the team’s Scrum Master when we encounter signs of these nightmares. With patient yet tenacious mentorship, even teams new to the distributed, part-time, and virtual collaboration environment will adapt.

Be honest. With which of these team environments would us feel most comfortable? The one just described or the one with positive traits that we know about and should encourage:

- **Self-organization** allows the team to get their work done, deliver value, and do so iteratively.
- **Trust** allows the leadership to let teams go about their business with the knowledge that they will deliver business value regularly and raise all issues and impediments when and as they arise.
- **Transparency** ensures visibility of the good, the bad, and the ugly. While everyone loves smooth sailing trips, storms and unexpected challenges will emerge. With transparency, we are not alone when we need to ask for help. In fact, if we have a transparent and passionate team, we will be asked whether we need help before we have an opportunity to ask.
- **Simplistic mindset** keep it simple! Ship something as soon and as often as possible. Through iterative feedback and rapid adaption, we can always do better tomorrow.

## Events are invaluable

Every team must agree to key events, such as Under Development, Ready for Review, or Done, and the tracking thereof to avoid distributed team members from waiting on each other, wasting invaluable time and eventually deadlocking the flight. Whether we track stories, tasks, or both is a decision we should make with the team.

## Coarse story tracking

One option is to encourage teams to break stories down into tasks where needed, but to visually track only the encompassing features and stories using the Kanban board in VSO. The Kanban board allows the team to visually transition stories through a series of states, such as New, Approved, Committed, Ready-for-Review, and Done as shown in Figure 66.

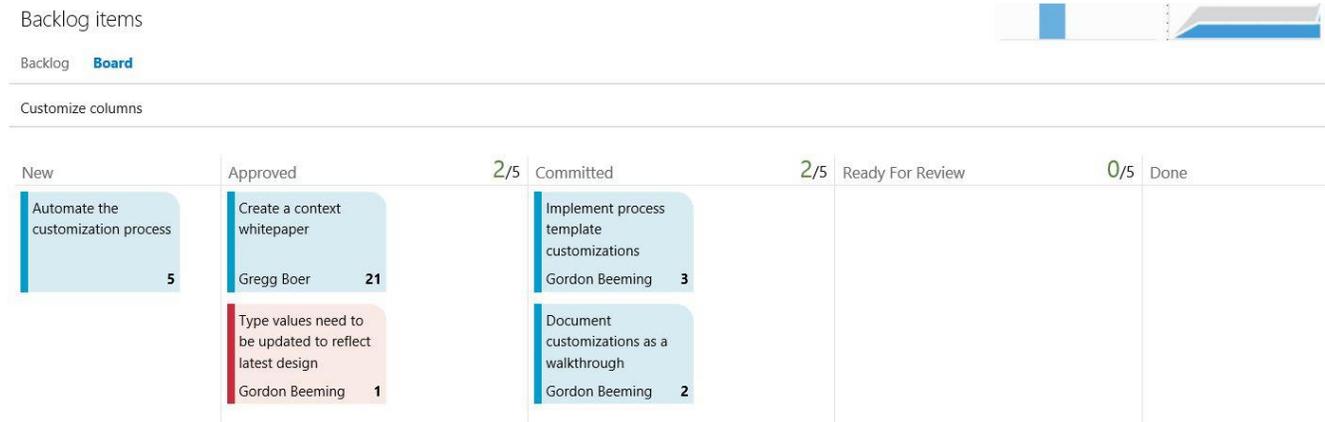


Figure 66. Kanban board and event: committed stories.

The idea is that the developer drags a story to Committed when there is a commitment to develop the feature and to Ready-for-Review when the story is ready for testing or review. Testers/reviewers typically monitor the Ready-for-Review state and grab stories when they appear.

### WARNING

#### Scrum vs. Kanban

Scrum and Kanban approaches can be mixed with caution:

- Scrum is based on fixed-length sprints, continuous reflection, and improvement.
- Kanban is based on WIP limits, visual workflow, and measuring and optimizing lead time.

A few core benefits of the Kanban board are that work-in-progress limits (WIP) can be used to define a healthy amount of work that can be queued in a specific state and, more importantly, that bottlenecks can be visually identified.

Those who know the process templates, associated work items, and states by heart will wonder where the ready-for-review state emerged. As shown in Figure 67, we have only the states New, Approved, Committed, and Done for bugs and product backlog items. Committed appears twice, once in the Committed and once in the Ready for Review column. This effectively allows us to keep the story in the Committed state, but transitioning through two or more stages, before changing it to the Done state.

Similarly, we could decide that a story needs to go through phases, for example Ungroomed, Ready for Triage, Triaged, Current Sprint, Future, etc. while in the Approved state. This is where the extensibility and customizability of Visual Studio Online (VSO) and Team Foundation Server (TFS) prove very powerful.

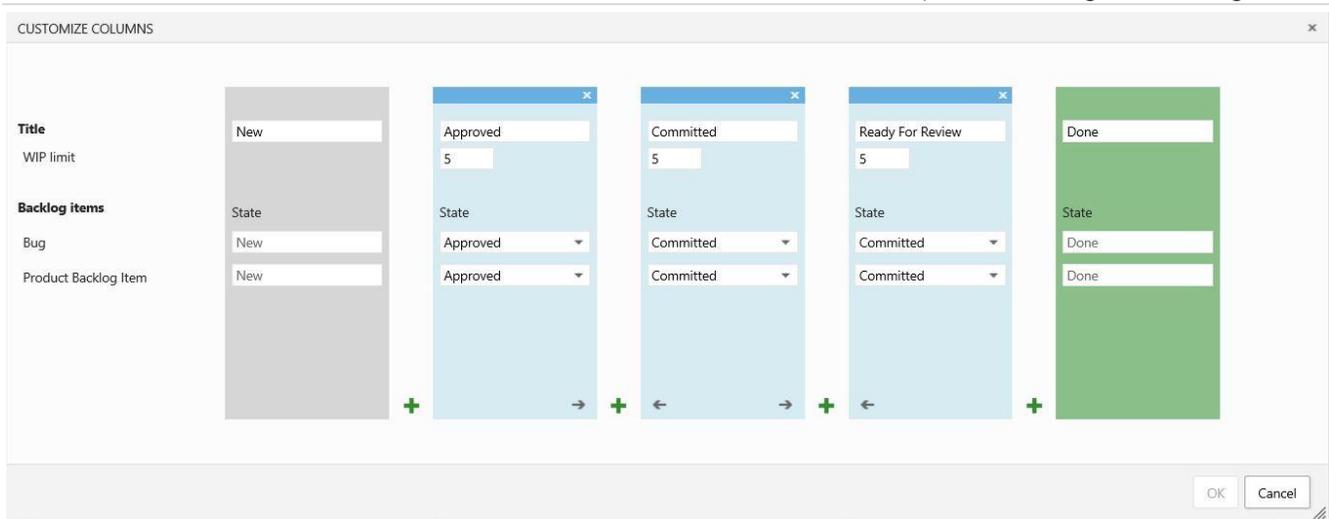


Figure 67. Kanban board customization.

## TOOLING

- [TFS Kanban board Swim lane customization walk through](#)<sup>60</sup>
- [Announcing Kanban for Team Foundation Service](#)<sup>61</sup>
- [Work from the Kanban board](#)<sup>62</sup>

Important state events for stories (PBI)

Table 9 describes states events for stories when you use Kanban for tracking, and Figure 68 shows the Kanban board.

State	Description
New	Story created, captured on the backlog, and ready to be approved by the Product Owner.
Approved	Story approved by the Product Owner and ready to prioritized, estimated, and implemented.
Committed	Team has taken ownership of the story and is committed and busy developing the feature.
Ready for Review	Development of story is complete and ready for review and validation.
Done	Story is developed and tested, meeting the definition of done and its acceptance criteria.
Removed	Restricted state used to indicate that the item is invalid. This could be due to duplication of data, the incorrect capture of data, or the requirement no longer being feasible.

Table 9. Important story (PBI) events when using Kanban.

<sup>60</sup> <http://aka.ms/xn3ov5>

<sup>61</sup> <http://channel9.msdn.com/Blogs/VisualStudio/Announcing-Kanban-for-Team-Foundation-Service>

<sup>62</sup> <https://www.visualstudio.com/en-us/get-started/work-from-the-kanban-board-vs.aspx>

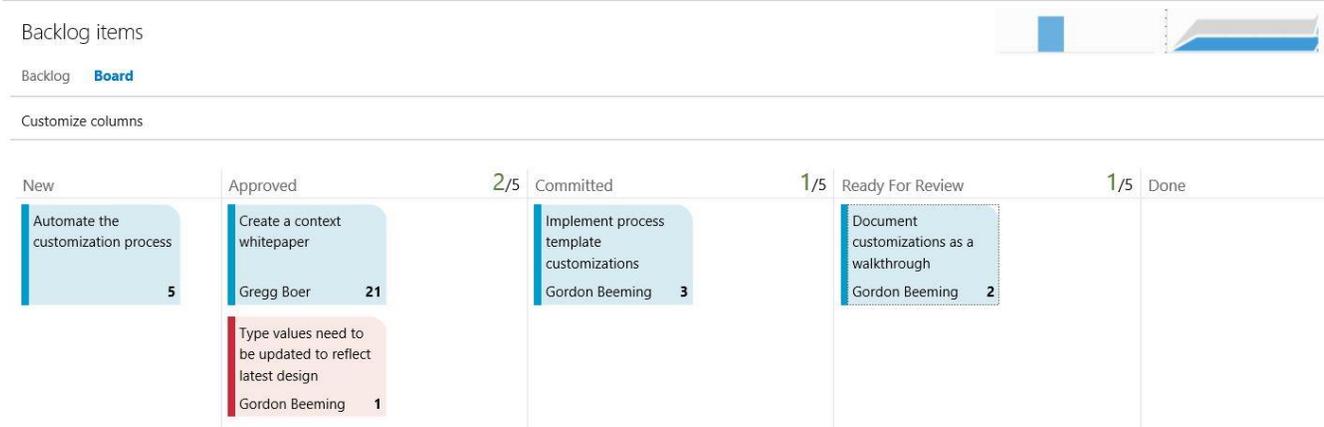


Figure 68. Kanban board and event: Ready for Review stories

### Fine-grained task tracking

An alternative or complementary option to the Kanban board and tracking stories is the use of the Task Board in VSO, shown in Figure 69. The Task Board allows the team to track bugs, stories, and associated tasks in a visual manner.

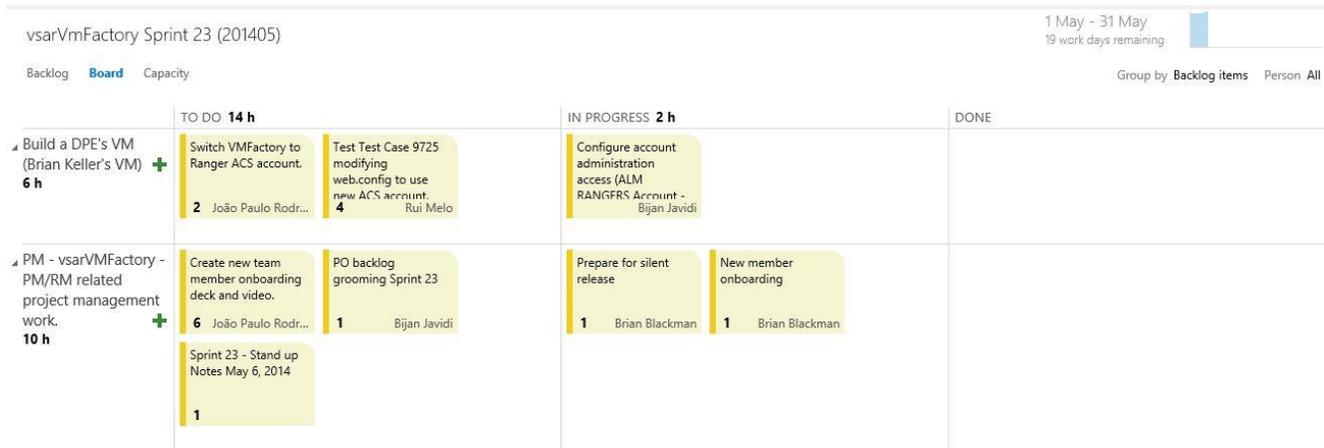


Figure 69. Backlog with stories and associated tasks.

There is no clear guidance for which option is suitable for a team or that delivers the best value. With part-time teams, the tax of the task-level tracking may exceed the value added to the team. On the other hand, the lack of task-level detail may hinder remote team members from committing to work efficiently.

The burndown pattern shown in Figure 70 has been observed in our environments and teams where the tax of task-level tracking seems to exceed the added value:



Figure 70. Sorry, forgot to update the tasks . . . they are all done . . . syndrome.

Work with the team to determine whether its members need mentorship on task-level tracking or whether the team is better off tracking only stories. Also see "[Glimpse of tomorrow](#)" for more on this topic.

## Automation is key

Automation is typically associated with unit and component testing. In the context of distributed and part-time teams, however, automation is invaluable across the board. Any automated task will reduce lead and dependency time, simplify team tasks, and enforce a level of consistency.

For example:

### Build

- Continuous integration automatically triggers a build during a check in. Have at least one continuous integration build defined for each team, running high-value tests, and encourage the team to react to build and test results.
- Gated check in triggers a build and commits changes only if the build and tests pass. Protect our stable branches with a gated check-in build.

#### TOOLING

- [Build your app](#)<sup>63</sup>
- [Hosted build controller](#)<sup>64</sup>

### Test

- Continuous integration automatically triggers automated tests as part of a build during a check in.
- Unit, Coded UI, and Load tests are good candidates to automate.

#### TOOLING

- [Run tests in your build](#)<sup>65</sup>
- [Load testing in the cloud](#)<sup>66</sup>
- [Initiate a Load Test from Hosted Build work-around](#)<sup>67</sup>
- [Traffic is a good thing. Can your app handle it?](#)<sup>68</sup>
- [Automate your lab environment to optimize your Application Lifecycle Management](#)<sup>69</sup>

### Validate

- Quality process validations, such as code scans, can be included in automated builds to validate the team's quality bar consistently.

<sup>63</sup> <https://www.visualstudio.com/get-started/build-your-apps-vs>

<sup>64</sup> <https://www.visualstudio.com/get-started/hosted-build-controller-vs>

<sup>65</sup> <https://www.visualstudio.com/get-started/run-tests-with-builds-vs>

<sup>66</sup> <https://www.visualstudio.com/get-started/load-test-your-app-vs>

<sup>67</sup> <http://aka.ms/iq0pla>

<sup>68</sup> <https://www.visualstudio.com/features/vso-cloud-load-testing-vs>

<sup>69</sup> <https://www.visualstudio.com/en-us/explore/lab-management-vs.aspx>

- Long-running validations should be deferred to nightly/daily builds to minimize impact on the team.

**Delivery**

- Continuous delivery using techniques such as versioning, continuous integration, automation, and environment management enables a decrease in time between an idea and making an impact in production.
- Strive for automated delivery to QA, DEV, and PROD environments using a release pipeline.

**TOOLING**

- [Building a Release Pipeline with Team Foundation Server](#)<sup>70</sup>
- [Deploy continuously to Azure](#)<sup>71</sup>

**Notify**

- Use VSO notifications to alert and notify users when events such as builds and check ins occur.
- Reviewers can monitor notifications for work dropping into the ready-for-review folder in case the developer “forgets” to notify the team in a timely way.

**TOOLING**

- [Set alerts, get notified when changes occur](#)<sup>72</sup>

## Metrics are another key

Detecting issues, solving them in a timely way, and continuously improving are key! Gather metrics by continuously monitoring availability, performance, and issues. Using the metrics, track adoption, usage, and potential failures and share the findings transparently.

<sup>70</sup> <https://msdn.microsoft.com/en-us/library/dn449957.aspx>

<sup>71</sup> <https://www.visualstudio.com/en-us/get-started/deploy-to-azure-vs.aspx>

<sup>72</sup> <https://msdn.microsoft.com/en-us/library/ms181334.aspx>

## TOOLING

- [Application Insights](http://azure.microsoft.com/en-us/services/application-insights/)<sup>73</sup>
- [System Center 2012 R2](http://www.microsoft.com/en-us/server-cloud/products/system-center-2012-r2/)<sup>74</sup>

## Key deliverables

The roadmap defines the number of sprints and the duration of each sprint (Figure 71). When the roadmap time box is over, the project is over, which will be the topic of discussion in the next chapter. When the sprint time box is over, the sprint is over, irrespective of the weather, world peace, or the status of the sprint backlog.

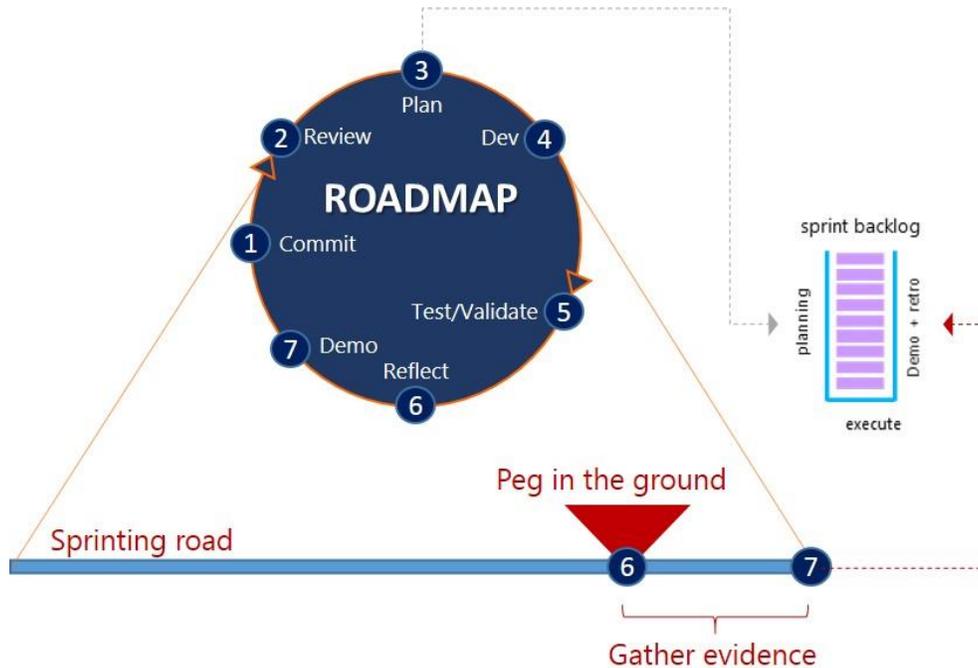


Figure 71. Sprint deliverables and the "peg in the ground."

As the team approaches the sprint "peg in the ground" milestone, we focus on review and the retrospective to gather evidence and be in a position to declare success or failure. In either case it is important to continuously reflect and improve and to recognize the accomplishments by the team.

Whether we put the peg in the ground at the end of the sprint or before is the team's call. With weekly scrums, the recommendation is to mark the scrum a week before the end of the sprint as the "peg in the ground" milestone. That gives the team a part-time week to deliver the core deliverables needed and declare the sprint as a success.

## Working solution

The principle "Working software over comprehensive documentation" (Agile Manifesto, 2001a) implies that the success of the sprint is not measured by the kilograms of documentation produced or the tons of passion invested, but instead in delivering working solutions.

- For guidance projects, working solutions implies professional and practical guidance, walkthroughs, hands-on labs, and videos that have reached the minimum quality bar in terms of readability, accuracy, and usability.
- For tooling projects, working solutions implies installable and functional tools that have reached the minimum quality bar in terms of code quality, code coverage, and usability.

<sup>73</sup> <http://azure.microsoft.com/en-us/services/application-insights/>

<sup>74</sup> <http://www.microsoft.com/en-us/server-cloud/products/system-center-2012-r2/>

The decisive test is how the reviewers, testers, and eventually the stakeholders perceive the solution.

Assume we go to the local grocery and buy a can of baked beans. After we read the brief instructions on how to open the can, what would we expect in the can? Comprehensive documentation for how to plan, harvest, and process beans or a portion of baked beans in tomato sauce, ready for consumption? Most likely the latter.

Every sprint is an opportunity for the team to deliver an update to the can with baked beans and obtain candid feedback and guidance from the consumer about how to improve the value added by the solution.

## Demo

The demonstration is an opportunity for the team, leadership, and stakeholders to synchronize face to face (preferred) or in a virtualFace-to-virtualFace teleconference call. Demonstrate each story that was committed by the team and discuss and harvest candid feedback from stakeholders.

After delivering the working solution and demo, the team should have a clear understanding of whether the objectives for the sprint were achieved—and, whether they were or not, what needs to be planned in the next sprint.

### GEM

#### **2–3 minute demo video “rocks”**

The testers/reviewers are in a prime position to record their validation tests and produce a video used as a demo and visual documentation. Keep your video to 2 (optimal) to 3 (maximum) minutes for maximum impact.

An ongoing debate is the value and effort to create demo videos. Value is undisputed! First impressions last, and a fine-tuned video delivers a much stronger message than a hands-on demo that “could” fail for a number of reasons outside our control. A professional demonstration requires careful planning, practice, and a pinch of good luck, all of which relate to a huge investment of time. Recording validation and test runs is seamless, requires no additional effort, and delivers evidence that can easily be edited into a 2-minute demo video.

Keep the demo (video) *simple*! Plan to create the demo as part of the sprint plan, not as an afterthought or last-minute alternative to a hands-on demo.

## Retro: One innovation

The intent of retrospective reflection is to harvest lessons and experiences during the sprints and to continuously improve and adapt our process to make projects more fun and the shared process more effective. Typically, a retrospective meeting gathers subjective evidence from team members and quantitative evidence in terms of process metrics such as cadence and work planned, completed, and carried over as incomplete or faulty.

### GEM

#### **Agree on one innovation**

Every team should agree on one innovation that will improve its experience and productivity.

In geographically distributed and part-time teams, scheduling a retrospective meeting can be a mammoth task and result in no constructive feedback, especially during “heads-down” sprints. An alternative is for the Scrum Master to gather the quantitative evidence throughout the sprint and ask the team to agree on just one innovation to be considered as part of the next sprint.

## Planning

Planning the next sprint is part of completing the current sprint. For fine-tuned teams that have a groomed and well-defined backlog, this can be included as part of the last scrum meeting and completed in 15–30 minutes.

Another way is to schedule a near-sprint-end meeting where the Product Owner reiterates the core objectives and features and where the team can brainstorm, refine, prioritize, and estimate backlog items. Also known as *backlog*

*grooming*, this planning forces team members to “sit on their hands,” reflect on the past sprint, and discuss the upcoming stories and sprint.

A well-groomed backlog enables transparency, estimations, and insight into what has and needs to be acted on. It is also far more efficient to work with a groomed backlog than have to sift through wasteful noise and distractions at every scrum, planning, or retrospective meeting.

GEM

**Keep your backlog groomed and clean!**

Consider your backlog as the main entrance of your project. Keeping it clean and groomed will ensure that first impressions are positive and visits collaborative and productive.

## Groom future development sprints

When we look up the definition of grooming, we are presenting with the following options (The Free Dictionary, 2014):

- *To care for the appearance of; to make neat and trim: groomed himself carefully in front of the mirror.*
- *To clean and brush (an animal).*
- *To remove dirt and parasites from the skin, fur, or feathers of (another animal).*
- *To prepare, as for a specific position or purpose: groom an employee for advancement.*

When the Product Owner (PO), Project Lead (PL), and Program Manager (PM) get together under the guidance of the Scrum Master to define and prioritize the roadmap, release objectives, ideas backlog, and the proposed sprint backlog, they perform the first grooming. The idea of investing invaluable time and effort in grooming the backlog is to produce a trim and clean backlog of requirements and better understand what needs to be done in which order and by when. In addition, the grooming removes noise and clutter introduced by out-of-scope requirements and prepares the team for the task. We usually note that we have seamlessly applied the definition of grooming the backlog during the sprint(s) focused on training-research-planning.

GEM

**Groom top down**

Ensure that the Product Owner and stakeholders prioritize all backlog items. Work through the backlog from the top (most important) to the bottom (least important) of the backlog items.

During each development sprint, from start to finish, we monitor and groom both the ideas and the sprint backlog. The validity of the backlog items change with scope creep (bad), requirements changes (reality), and the implementation of features (good).

“Responding to change over following a plan” (Agile Manifesto, 2001a) highlights the reality of requirements changing over time. Until we invent instantaneous solution development, we will have to face the reality of continuous change, which in turn requires continuous grooming.

## Objectives/Goals

*What if the Product Owner decides to change the release objectives, acceptance criteria, the definition of done, or goals?*

Remember that these are pillars of the team—the definition of the requirements and, more importantly, the definition of success. A change in any of these pillars requires the team to reflect and respond.

**WARNING****Responsiveness of distributed and part-time teams is less than optimal**

- The more time zones the team is distributed across, the longer it will take team members to receive, react, and respond to a change notification.
- Scheduling short-term and urgent meetings is challenging, especially when working with part-time volunteers.

Remember family > real job > part-time moonlighting adventure

The Product Owner is the team's [Judge Dredd](#)<sup>75</sup> responsible for the solution and “shipped” business value. It is therefore important for the team to react to the changes and negotiate with the stakeholders how to maximize business value and minimize waste on the project.

The exact refactoring depends on the nature of the change. The team should not resist the possible need to reestimate, reprioritize, or even redo already-completed work. The ultimate goal is to resume as quickly as possible, minimize waste, and maximize the value of the solution with respect to the changes.

**GEM****Short sprints == less waste**

The shorter the sprint, the less the waste is incurred during an iteration if you need to respond to change.

## Backlog

*What if the Product Owner decides to change features?*

There is no problem here if the features or associated stories are on the backlog and have not been planned for the current sprint. The team can work with the Product Owner to understand the changes and reestimate during the next planning session.

If the change affects features and associated stories forecast for the current sprint, we need to stop, reflect, and respond to the change (Figure 72). As with changes to objectives and goals, the exact refactoring of the impacted stories depends on the nature of the feature change.

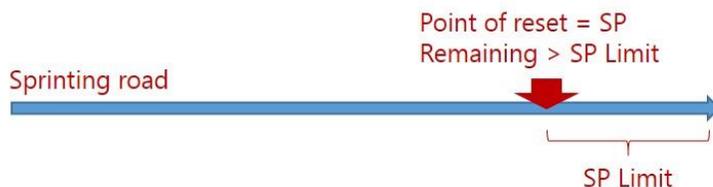


Figure 72 Point of reset.

The team should not commit to more story points than the average velocity in a sprint. Resource bandwidth and sprint and roadmap limits are static, which means that the team must renegotiate scope. If the remaining story points exceed the story-point limit for the remainder of the sprint, a sprint reset is a consideration. In this case, the team can negotiate a switch to hardening and quality improvement for the remainder of the sprint, resuming development in the next sprint.

It is important to maintain momentum, especially with part-time and distributed teams. Instead of a hard reset and having team members sit on their hands until the next iteration commences, time should be invested in testing, hardening, and bug fixing.

<sup>75</sup> [http://en.wikipedia.org/wiki/Judge\\_Dredd](http://en.wikipedia.org/wiki/Judge_Dredd)

## Confidence

*What if the team decides to change its confidence in delivering the sprint objectives?*

Stakeholders should take note of a change in the team's confidence vote. If the confidence increases, we have no problem. An increase typically signals a maturing team, a rise in passion and intent, and confidence in the current and future releases.

If confidence decreases, we potentially have a problem, ranging from a lack of pizza to an imminent [Chernobyl disaster](#).<sup>76</sup> It is important for the stakeholders, in particular the Product Owner, Project Lead, and Program Manager, to analyze the cause of the decline and to reflect, plan, and adjust scope until we restore confidence.

### **WARNING** Low confidence is disastrous with distributed and part-time teams

A part-time and volunteer-based team is driven by passion, ego, and confidence. A loss in confidence is infectious and may result in parts or your entire team going dark.

## Be brave and reset if needed

*What if we realize we have to stop and reset?*

"Responding to change over following a plan" (Agile Manifesto, 2001a), by its very nature, implies that there comes a time when it is a good idea for a team to stop, reflect, and improve the situation. If this requires a reset of the current sprint or even the current release roadmap, the team should not resist. Remember that the Product Owner is responsible for maximizing the value of the solution and the team. However, it is important to remember that the Product Owner is responsible for delivering business value and has the authority to approve or cancel a release.

If a reset or worse, a cancellation of the current release, maximizes the overall value to the business, then it is a drastic decision the team needs to accept.

Always reflect to understand what happened and innovate to keep it from happening again.

What are some of the possible reasons for resetting or cancelling a sprint?

- Architecture or technology used to develop or host the solution is phased out.
- Business strategies change, eliminating the perceived business value of the solution.
- Expertise or resource bandwidth is no longer committed or available to the team.
- Extraordinary circumstances, such as a natural disaster that impacts the team or solution.

## Watch for "smoke" signals

As described herein, the two major negative signals to monitor are:

- **Scope change (creep)** from top down, which causes churn and frustration with the teams, especially those investing their personal time on part-time, volunteer, and moonlighting projects.
- **Decline of confidence** from bottom up, which indicates a loss of passion and commitment and a potential for parts of the team going dark. Figure 73 shows the effect of these signals.

<sup>76</sup> <http://en.wikipedia.org/wiki/Tschernobyl>

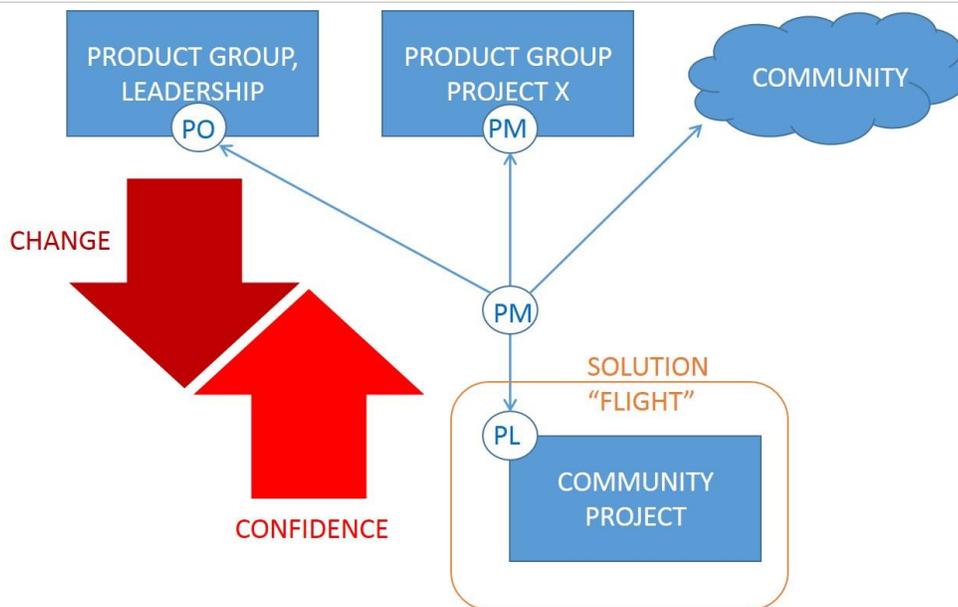


Figure 73. Keep an eye out for negative signals.

## Deliver on demand with silent preview releases

The *silent preview release* is our typical term or process checkpoint. It is a release we can introduce to a small group of “friends” for early feedback and possible realignment. The core intent is to give early adopters an opportunity to evaluate and give candid feedback while the team raises the quality bars. (See Figure 74.) The intent during a silent release is not to awaken the pedantic quality, spelling, grammar, or coding style validations but to determine whether the team is in the “correct movie” and avoid disappointment, confusion, and discouraging dust storms when they ship.

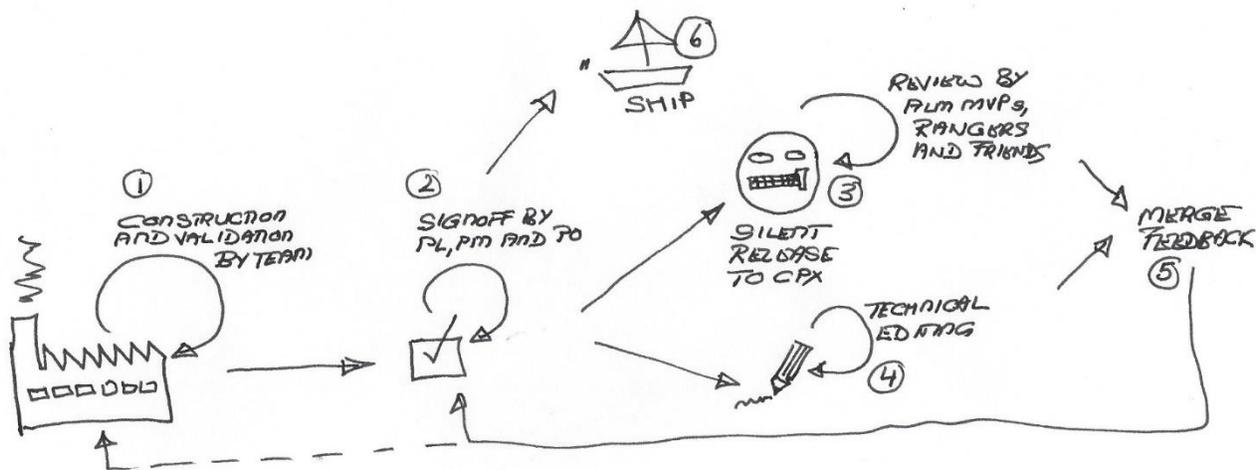


Figure 74. Silent preview release a la whiteboard.

**TOOLING** [TFS Feedback Management Behind the Scenes](http://blogs.msdn.com/b/slange/archive/2012/10/04/vs-tfs-2012-tidbits-tfs-feedback-management-behind-the-scenes.aspx)<sup>77</sup> (Lange, 2012)

Looking at this immaculate whiteboard scribble:

1. The solution feature teams construct the features and validate them as normal.

<sup>77</sup> <http://blogs.msdn.com/b/slange/archive/2012/10/04/vs-tfs-2012-tidbits-tfs-feedback-management-behind-the-scenes.aspx>

2. When the solution reaches a minimal quality level (as agreed at kickoff), the Project Lead (PL), Program Manager (PM), and Product Owner (PO) can consider and agree to “ship a preview silently.”
3. The working solution is shipped silently by using a public delivery channel, for example [www.CodePlex.com](http://www.CodePlex.com), and clearly marked as a beta. There is no public announcement, only an email announcement to selected friends, with clear instructions to evaluate, deliver candid feedback, and not to blog, tweet, or announce in any other form. Silence is golden!

**NOTE****Clarify the beta reality! Example:**

- CAUTION. Wandering into beta territory could ruin anyone's day fast, so beware! We are conducting content-active tests within an evaluation radius and this guidance should be used with caution during beta state.
- PLEASE DO NOT blog, tweet, or make noise about this "silent" beta release!

4. In parallel, the team continues to raise the quality bar. For example, technical editors start reviewing the content, correcting spelling, grammar, context, terminology, and other gremlins. Having volunteers from around the world, with a variety of home languages, often results in adventurous deliverables that need fine-tuning.
5. The silent release feedback is acted on, and when team members believe that the quality bars are met, they once again ping their PL, PM, and PO for sign off. The result is another silent release or a public release with broad announcements.

## Dealing with bugs

In this context, we refer to the work-item type Bug, used to report missing or incorrect features, crashes, or other unwelcome guests in software development lifecycles.

### Self-describing bugs

The concept of “garbage in—garbage out” (GIGO) is old but still very applicable! It is important that we educate and guide anyone who captures a bug to take special care when doing so. For example:

- Title is the first and most often viewed bug field. Make sure it is self-describing.
- Specify the estimated effort and severity to help the triage team.
- Description and attachment data should be used to add context and evidence to the bug.
- Area Path must be set to the appropriate feature team.
- Iteration Path should be set to a predefined triage path.
- Assigned To must be assigned to the Project Lead (PL) or left blank if the originator is unsure who the PL is.

Incomplete or bad data in a Bug WIT results in unnecessary delays or in bugs vanishing off the radar.

Some teams often do not create bugs for those found by the team but only for those found outside the team. The thought here is that the bugs are fixed right away as part of the sprint since they were found in the sprint.

### Triage

The Project Lead takes ownership of new bugs and performs a triage, which can be discussed during the second half (Q&A) of the next scrum meeting.

- Consult the originator of the bug if there is any uncertainty.
- Consult the stakeholders if the correct priority and severity are unclear.
- Revise information to ensure clarity and accuracy.
- Prioritize the bugs in terms of the quality bar. If our quality bar states that no critical bugs are acceptable, we should focus on critical bugs before working on high-priority, medium-priority, or low-priority bugs.
- Assign the bug to a team member, who assumes ownership of the bug.
- Set State to Approved.
- Set Iteration Path (sprint) to the iteration (sprint) during which the bug must be resolved.



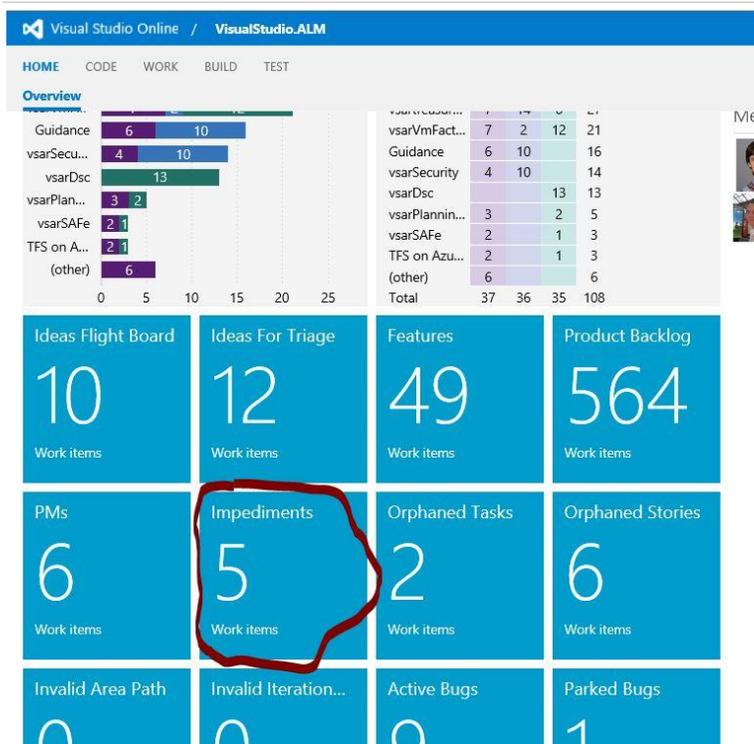


Figure 76. Impediment dashboard.

## Dealing with scope creep

A change in scope can result from an unexpected business or service requirement or from a requirement that's misunderstood.

The former is "valid" scope creep for which the Product Owner determines that the business value of the remaining Features is no longer valuable, resulting in expensive waste if developed.

A misunderstood requirement is a disastrous scenario. If the solution's vision, goals, objectives, or requirements are fluid or not understood, the resultant features and stories will be proportionally unstable. Some of the most common signs of "bad" scope creep include:

- The Product Owner is not engaged with the team and shows little interest or passion in the solution or team.
- The Product Owner's expectations and team deliverables are not aligned at the end of a sprint.
- The Product Owner continuously introduces feature, vision, goal, or objectives release changes.
- The Product Owner cannot link scope creep to valid architecture, business, or technology changes.

Transparency and trust are key! Without transparency, there can be no trust. Without complete trust in the Product Owner, the team cannot be committed. It is therefore paramount for Scrum Masters and Program Managers to get involved if there is dysfunction due to lack of trust, transparency, an inexperienced Product Owner, or poor backlog grooming or management.

The Product Owner has the responsibility to determine whether a sprint, roadmap, or release reset is required.

Nothing frustrates a team of volunteers more than investing precious family time in a solution that's canceled and moth-balled. The Product Owner must take this into consideration before pulling the eject handle.

## Dealing with critical chickens

Lastly, we need to recognize and deal with "critical chickens" who demotivate the team. Critical chickens are team members or stakeholders who are determined to always point out what could, has, or is going wrong but neither

take responsibility nor get involved in resolving the issue or impediment. All of us know a “movie critic” persona who never has anything positive to say about any movie, a “technology critic” for whom no technology is good enough, and a “doomsday critic” who always has a million and one things that can go wrong.

Not all critics are “critical chickens”! Team members who feel accountable for the success of the project deliver candid and constructive feedback and then are engaged in finding a resolution. They know that they cannot succeed if the team fails . . . we rise or fall as a team!

With distributed, part-time, and (especially) volunteer-based teams and community projects, critical chickens are poisonous and must be nudged off the release “flights.”

## Dogfooding case study: Triage quadrant

Visualizations are key, and different stakeholders react “positively” to different views.

### Background information

In “[Triage of ideas](#),” we introduced the ideas triage and associated visualizations that our community is familiar with. If we assume we have 19 hypothetical prioritized project ideas, titled A-S, we get the following raw view by using the sample triage workbook. Based on the size, business value, and priority as defined by the stakeholders from the ALM community and ALM Rangers, we clearly see in Figure 77 that the top two ideas would be A and B.

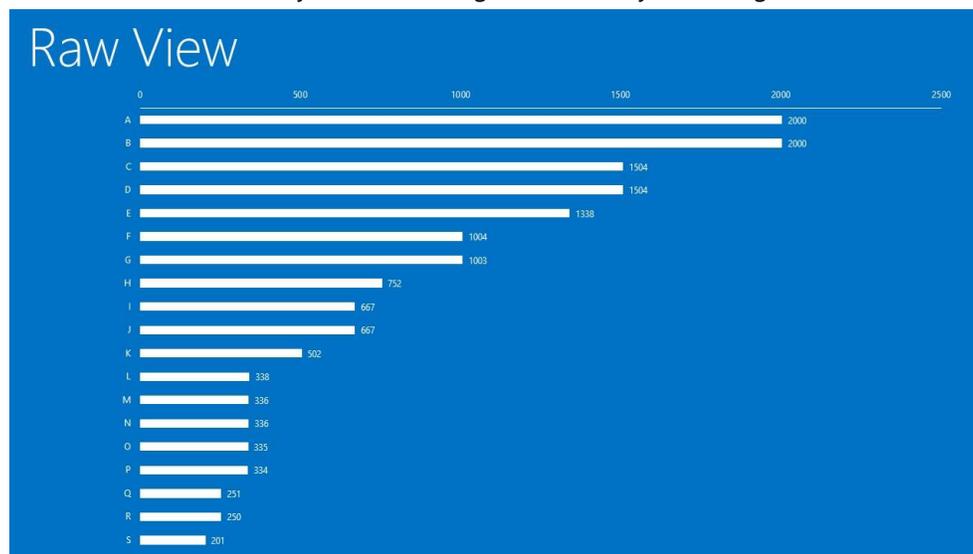


Figure 77. Raw view.

[Wouter de Kort](#)<sup>78</sup> commented, “*the one we're used to. I like it that it's very easy to see the relative value,*” when we reviewed a number of potential visualizations. He also said, “*What could help is split the bar in two colors for community value and PG value,*” but more on that later.

### The quadrant triage experience

The consolidated triage workbook introduces a new quadrant visualization, which we dogfooded for a while. The view is generated after our PMs have normalized the raw data, collaborating with product group PMs and other stakeholders. The intention is to position each idea in one of four quadrants: Community, product group, win-win or reject. The quadrant that contains the real gems is top right, which has a high community and product group

<sup>78</sup> <http://aka.ms/vsar-wdk>

value . . . a win-win. We notice in Figure 78 that both idea A and B are included, as well as E, which received a confidence boost during the raw data normalization.

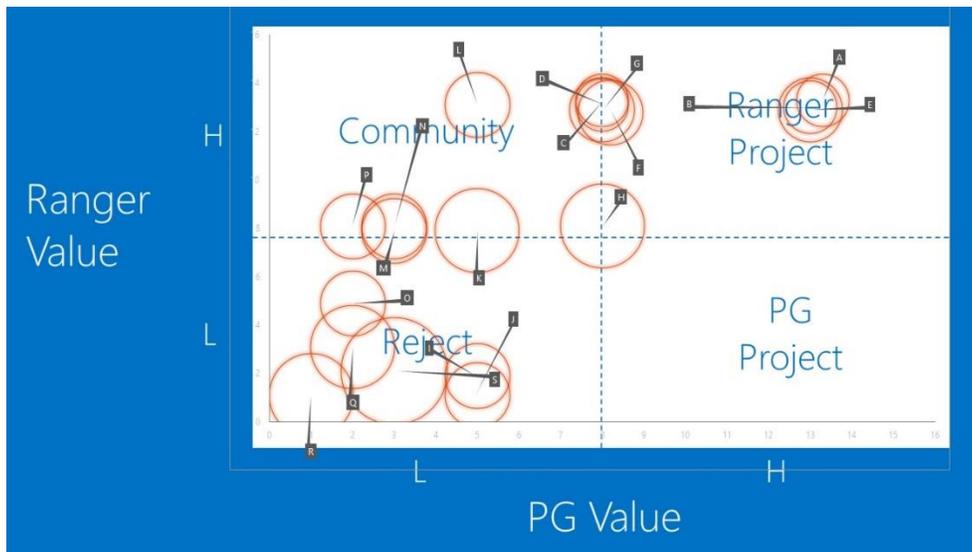


Figure 78. Quadrant view

When we turned the above view into a stacked view (Figure 79), [Wouter de Kort](#)<sup>79</sup> was happy: *“That’s what I meant. Helps in understanding the position of an item. Also shows what the PG likes.”*

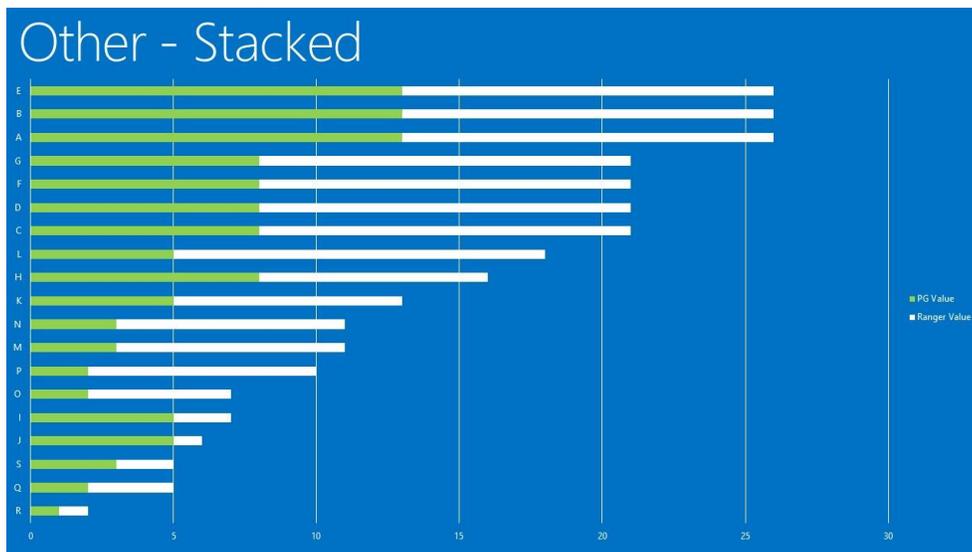


Figure 79. Stacked view.

Be careful not to introduce fancy looking but noisy and low-value visualizations such as those in Figure 80. We collaborate with stakeholders and determine what works and what does not. We conclude this case study with two more visualizations, for which [Wouter de Kort](#) commented, *“Hard to read.”*

<sup>79</sup> <http://aka.ms/vsar-wdk>



Figure 80. Other views.

## Key learnings

- It is key to adopt an iterative lifecycle pattern that is natural to the team.
- By having weekly team scrums, the team is fully aware of the status of the project and any impediments. This raises the visibility and overall transparency of the project.
- A regular scrum of scrums, once every sprint, assists with transparency and helps remind everyone of the flight vision and motivation, but it also helps to bubble up any questions and dependencies with other flights and teams.
- With distributed and part-time teams, automation can reduce lead and dependency times and enforce a level of consistency. This allows team members to focus on what they do best.
- It is important that teams continuously reflect, improve, and recognize the accomplishments of the team. Trust and transparency are the key.
- With shorter sprints, the team can ship more quickly, which will reduce waste and keep team passions high.

# Chapter 4: Raising the quality bar

*Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.*

—Winston Churchill

We complete our roadmap and associated release flight by raising the quality bar and planning the future (Figure 81). This is an opportunity for team members to innovate, plan, and reflect on the project, themselves, and the overall process in order to support continuous improvement. It provides the time to complete activities often forgotten or postponed during critical development and testing phases, especially with geographically dispersed and part-time project teams.

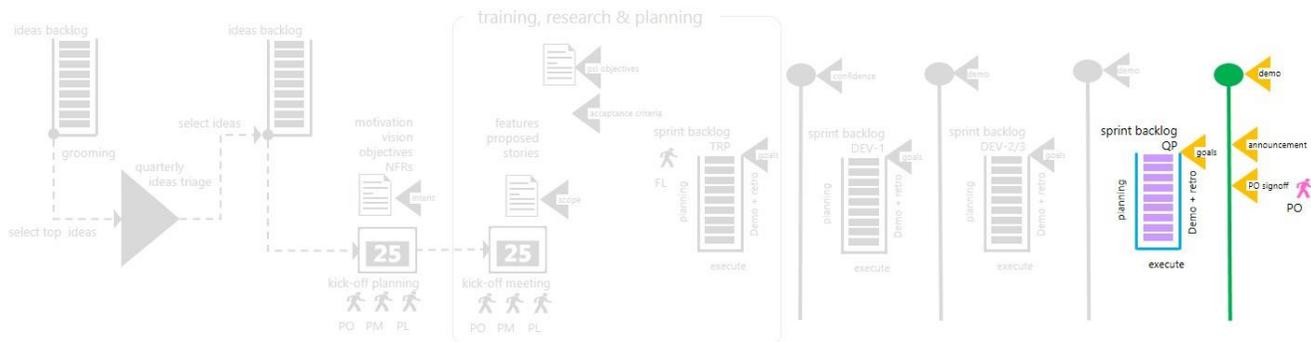


Figure 81. Lifecycle: Quality and planning.

## Quality and planning (QP)

At the end of every sprint, we ship a working solution and offer evidence that we are making progress. The idea is to release early and often and be in a position to react to business value changes and candid feedback. What makes the final focus on quality and planning (QP) different is that we have a goal of shipping (also referred to as *landing*) the release. The team completes the roadmap and releases the team resources for other project adventures or another release (vNext) of the solution.

The core themes of the focus on quality and planning (QP) are to raise the quality bar and to plan the future. In addition, the team may use this opportunity for additional research, innovation, and hackathons for the future.

## What it is not

It is important that every team have a discussion to understand what the focus on quality and planning is *not*.

- It is not a dumping ground for technical debt, such as code refactoring, review, or quality processes. We need to schedule and include these technical-debt activities during every sprint—to continue them with a view on learning and improving.
- It is *not* about reaching a minimum quality bar. We need to reach a minimum quality bar at the end of every sprint.
- It is *not* an opportunity to schedule incomplete features. In fact, developers should be “sitting on their hands,” and teams do not permit new feature development during this sprint.

## Hardening

With an understanding of what the focus on quality and planning (QP) is not, the time is right to talk about the hardening, innovation, and planning objectives. At this stage of the flight we have reached the minimum quality bar as agreed, using the definition of done and acceptance criteria. We continue to harden or improve the solution,

ensuring that first impressions are positive, support and maintainability cost and resources are effective, and the ability to switch to the vNext release or other adventures is seamless and effortless.

In “[Triage of ideas](#),” we introduced the notions of a quality bar, acceptance criteria, and a definition of done. Essentially, the quality bar defines the minimum levels of nonfunctional requirements, code, and documentation quality; standards and process compliance; and bug resolution that the team must meet, before our Product Owner (PO) will be happy to sign off the release and trigger the shipped, or landed, status.

In our ecosystem, the quality bar has a dial ranging from low to high. When we are shipping quick-response samples, for example, the dial will be closer to Low, which means that intensive reviews and quality essentials process compliance are not required.<sup>80</sup> The closer we get to shipping tooling that contains binaries and service-level agreements, the closer the quality bar dial moves toward the quality and standards of the Developer Division (Figure 82).



Figure 82. Quality bar.

Common criteria included in a definition of done (DoD) and acceptance criteria that determine the minimum quality bar include:

- All critical or high-priority bugs must be triaged, resolved, or deferred to vNext.
- Teams must remove from scope, hide, or defer to vNext all incomplete features.
- All code must comply with:
  - A predefined namespace, for example, Microsoft.ALMRangers.
  - Consistent code analysis dictionary and common assembly information.
  - StyleCop, using a consistent and common rule set.
  - Code Analysis for both Debug and Release builds.
  - ReSharper rule sets, which should be mandatory.
- X, Y, and Z must review all code, which typically includes personas such as development lead, subject matter experts, and Program Manager.
- X, Y, and Z must review all documentation, which typically includes personas such as copy editor, subject matter expert, and Program Manager.

**NOTE** The quest for quality should not be an afterthought. Start on day one, and your final review experience will be a much better one. Typically, you need less time, effort, and anti-review-depressants during this phase if you define quality and validation criteria from the start.

**TOOLING** [Code Analysis for Managed Code Overview](#)<sup>81</sup>

## Stabilize deliverables

*“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software”* (Agile Manifesto, 2001b). The Agile Manifesto emphasizes the need to satisfy our users by delivering valuable solutions early and continuously. The hardening topic covered herein should not be limited to print(s) focused only on quality and planning (QP) but should be a guide for each sprint, because we are in a continuous quest for quality.

To satisfy the user we need to ensure that solution features comply with user expectations as defined by the feature description and acceptance criteria. Using VSO we can define one or more test cases for each feature and story we implement, which completes our requirements model as illustrated in Figure 83.

<sup>80</sup> Quality essentials is the definition of engineering policies, standards, and procedures in our organization.

<sup>81</sup> <https://msdn.microsoft.com/en-us/library/3z0aeatx.aspx>

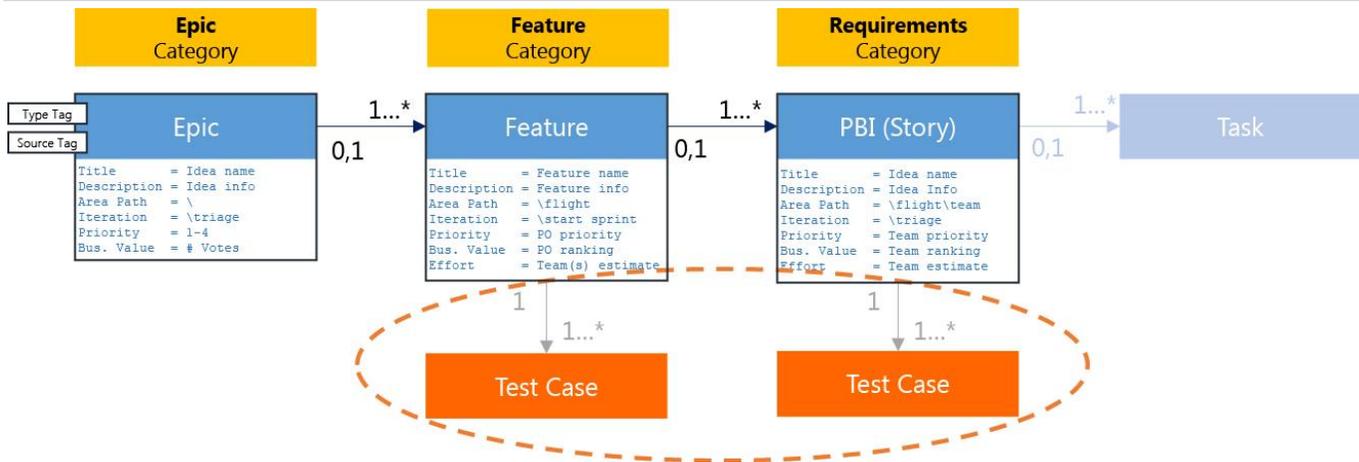


Figure 83. Requirements model: Test cases.

We always need to remember that test cases validate both the users' and the technical view of the system. The former validates expectations from a user—for example, if we try to log on with invalid credentials and are presented with a logon failure error. The latter validates technical expectations from the architecture and solution—for example, if the number of active users exceeds 2,500, we will enable a load balancer automatically.

**TOOLING**

For more guidance and information on Visual Studio Test Tooling, see [Visual Studio Test Tooling Guides](#),<sup>82</sup> [Visual Studio Testing Tools](#),<sup>83</sup> and the *Better Unit Testing using Microsoft Fakes* e-book software testing section from the (Visual Studio ALM Rangers, 2014).

**GEM****Automation “rocks”**

While there definitely is a place for manual testing, test automation allows teams to run more tests, more often and continuously. Carefully evaluate and select the test tooling that suits your team and investigate this: [When to Test Manually and When to Automate](#).<sup>84</sup>

Why unit testing is important

When someone asks why unit tests or testing in general is important, ask whether they believe it is important to thoroughly test commercial space probes, frequently used to travel across vast space, before each flight. Is it important that the probe is flight tested? Yes. Okay, now is it important that the navigation system is itself well tested? Testing the navigation system of a space probe is an example of unit testing. However, the navigation system does not guarantee the probe will fly, but the probe really cannot travel accurately without it.

**NOTE**

[Climate Orbiter](#),<sup>85</sup> the unfortunate confusion with the metric system and subsequent crash of the ~\$125 million Mars orbiter, is a stark reminder that quality testing and continuous analysis of metrics is invaluable.

The importance of testing begins at the start of the project and continues throughout the application lifecycle. Testing is important when we are striving for:

- Higher-quality code from the start
- Fewer bugs
- Self-describing code
- Reducing the cost of fixing bugs by fixing them earlier rather than later

<sup>82</sup> <http://aka.ms/treasure27>

<sup>83</sup> <https://www.visualstudio.com/en-us/explore/testing-tools-vs>

<sup>84</sup> <http://blogs.msdn.com/b/steverowe/archive/2008/02/26/when-to-test-manually-and-when-to-automate.aspx>

<sup>85</sup> <http://mars.jpl.nasa.gov/msp98/orbiter>

- Solution responsibility and ownership through pride down to the bits and bytes
- Customer satisfaction

**NOTE** The core value of test-driven development and unit testing is to ensure that the team thinks and even dreams about the code, scrutinizes the requirements, and proactively avoids problems and scope creep.

## Quality essentials

The concepts and definitions of quality, process compliance, and validation vary from organization to organization. With every solution, teams need to consider the quality essentials that define engineering policies, standards, and procedures. These quality essentials influence all our adventures and projects and associated flights.

As we have already mentioned, the quest for quality should not be an afterthought! We review and implement the requirements for our environment as early as possible. The quality essentials embrace both mandatory policies and practical guidance that ensure that our solution is compliant with the latest standards and user experience guidelines and results in satisfied users.

Examples of some of the typical engineering policies may include:

- **Accessibility** Focused on making it easier and more intuitive for users to see, hear, and use the solution on a computer device. For example, using the color red ● and green ● to signal states can be a compliance issue. The most common form of color deficiency is “red-green color deficiency,” not to mention that many screens and printers are still monochrome.
- **Export controls** Products will be subject to export and import control laws in many countries and regions. While this policy is not often applicable to community solutions, awareness of potential laws is good to have when planning and designing a solution that targets an international audience.
- **Geopolitical** Solutions must be politically and culturally sensitive and cognizant of local laws and regulations. Avoid using national symbols, flags, or icons or political, religious, or contentious images.

### WARNING

#### First impressions last!

In an ALM workshop many, many years ago, a colleague had a slide with a “knight in shining armor” to represent the Scrum Master who comes to the aid of the team in a gallant and professional manner. The workshop was a roaring success until presented in the Middle East, where the attendees associated the knight with the Crusades. Although there was no intent for disrespect, the regional interpretation resulted in a culturally sensitive situation.

- **Privacy** The collection, use, and transfer of user data are sensitive issues. Teams must predicate the collection and usage of data on obtaining user permissions. Teams must transparently explain privacy policies to all users.
- **Licensing** Licensing terms protect intellectual property and clearly outline the rights, restrictions, and obligations. The use of open source software (OSS) must be clearly identified to the user and be compliant with review and usage licensing.
- **Security and integrity policy** Protect all solutions, associated supply chains, and users against internal attacks and security breaches. Many organizations, such as law enforcement, rely on and therefore enforce software integrity.

The quality bar “dial” determines which of the quality essentials engineering policies are optional, recommended, or mandatory. Avoid lengthy delays and frustrating reengineering caused by noncompliance. Include a focus on quality essentials before the release takes off rather than after it lands.

Whether we track the process as a quality-essentials story or as a detailed list of quality-essentials policy tasks is up to us and our organization. Figure 84 shows the task-based QE list for the [ALM Readiness Treasure Map](http://aka.ms/treasure4)<sup>86</sup> solution from our VSO backlog.

<sup>86</sup> <http://aka.ms/treasure4>

Product Backlog

Results Editor Charts

Save query [refresh] [undo] [redo] [print] [share] [column options]

ID	Work Item...	Backlog P...	Title	Assigned To
5523	Product Ba...		Quality Essentials Process	Willy-Peter !
5524	Task		Create QE Release for v2	Willy-Peter !
5525	Task		QE - Accessibility - D12MAS 05: Visual Focus Indicator and Focus Order Procedure (Mandatory)	Anisha Pindi
5526	Task		QE - Accessibility - D12MAS 08: Seizures Procedure (Mandatory)	Anisha Pindi
5527	Task		QE - Accessibility - D12MAS 09: Animated Content Procedure (Mandatory)	Anisha Pindi
5528	Task		QE - Accessibility - D12MAS 14: Sensory Indicators Procedure (Mandatory)	Anisha Pindi
5529	Task		QE - Accessibility - D12MAS 15: Images of Text Procedure (Mandatory)	Anisha Pindi
5530	Task		QE - Accessibility - D12MAS 16: Using Color Procedure (Mandatory)	Anisha Pindi
5531	Task		QE - Accessibility - D12MAS 17: Visible Text Contrast Procedure (Mandatory)	Anisha Pindi
5532	Task		QE - Accessibility - D12MAS 18: Color or Contrast Selection Procedure (Mandatory)	Anisha Pindi
5533	Task		QE - Accessibility - D12MAS 20: Describing Content, Controls, or Links (Mandatory)	Anisha Pindi
5534	Task		QE - Accessibility - D12MAS 24: State of Accessibility Settings (Mandatory)	Anisha Pindi
5535	Task		QE - Accessibility - D12MAS 25: Consistent UI Procedure (Mandatory)	Anisha Pindi
5536	Task		QE - Accessibility - D12MAS 26: Consistent Navigation Procedure (Mandatory)	Anisha Pindi
5537	Task		QE - Accessibility - D12MAS 27: Efficient Navigation and Interaction Procedure (Mandatory)	Anisha Pindi
5538	Task		QE - Accessibility - D12MAS 19: User Input Instructions (Mandatory)	Anisha Pindi

Task 5536: QE - Accessibility - D12MAS 26: Consistent Navigation Procedure (Mandatory)

This procedure is required because: Your product, service, or tool has, or allows users to generate, content or a visual user interface.

Figure 84. Quality essentials sample list (extract).

## Copyediting/Reviews

Can you remember the cool product you bought and how your frustration increased while your excitement declined as you pondered the incomplete, complex, or illogical user documentation?

Subject matter experts (SME), content editors, and copyeditors must review all documentation:

- SMEs verify that the documentation is factually correct and that its content, length, and level of detail are complementary for a successful deployment and use of the solution.
- Content editors support the team with proofreading and fact checking, often getting involved with the content development. These editors are usually involved from start to finish.
- Copyeditors are the last line of defense and focus on proofreading, fact checking, flow, and readability. When teams deem a document ready for consumption, these editors often review the material.

Using VSO and the power of Team Foundation Version Control (TFVC), we effectively manage documentation throughout the flight (project). As shown in Figure 85, most of our solutions have a documentation node in TFVC with a predefined set of folders.

1. **Work in Progress** For content that is under construction, whereby we recommend using templates to simplify common formatting tasks and ensure that the editors receive consistent-looking content. Move to the next folder when done to signal a state change and notify the team that the documentation is ready for the next step—for example, review or copyediting.
2. **Copyediting** For content that is ready for copyediting. Content developers should “sit on their hands” while the copyeditor edits the content. The use of comments and inline change tracking is highly recommended.
3. **Ready for Review** For content that is ready for SME, community, and content review. The use of comments and in-line change tracking is highly recommended, whereby reviewers can check in the document with revisions if there are no merge conflicts or submit it to the content owner for manual merging.

4. **Ready for Integration** For content developed in separate documents when it's ready for us to merge into the master document.
5. **Master** For content that is ready for distribution.

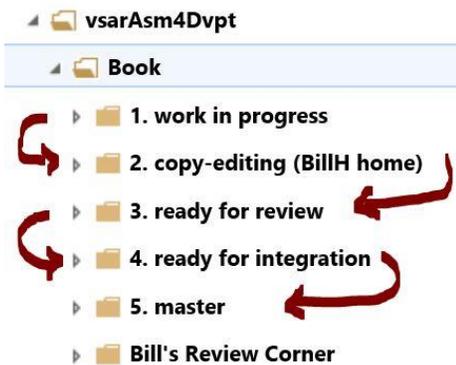


Figure 85. Documentation review using folders to support the workflow.

The documentation process and use of the folders illustrated in Figure 86 depicts the workflow used for this publication. Every solution has its own workflow and associated folders. For example, in most of our guidance projects, copyediting is done last, after the SMEs and community have reviewed the guidance and the master has been merged.

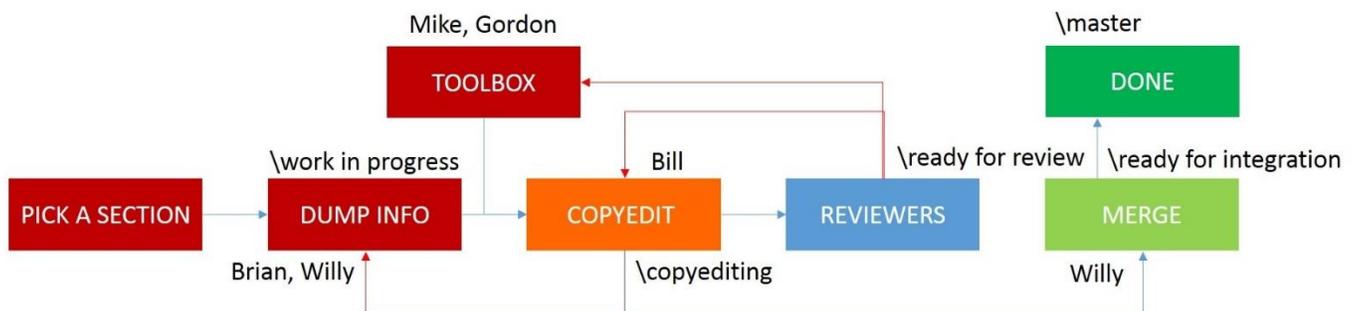


Figure 86. Review process.

A picture tells a thousand words. Together with crisp, accurate, and readable documentation, we satisfy the customer and ensure that first impressions are positive.

## Shippable release

The question of when a release is shippable is easier to answer than when should we release.

### When is a release shippable?

Simple! When we get the “thumbs up” signal from the Product Owner, we have a shippable release.

We base this simplicity on the assumption that the Product Owner defined, owned, and tracked the vision, objectives, acceptance criteria, and features on the backlog. A successful flight and landing requires an on-going participation in scrum meetings, show-what-we-have demonstrations, and accepting the sprint deliverables from the Product Owner (Figure 87). While this is usually a primary role and expectation in solution lifecycles, actual involvement may be sporadic, part-time, and require a lot of support from the Program Manager and trust in the overall team for geographically distributed, part-time community projects.

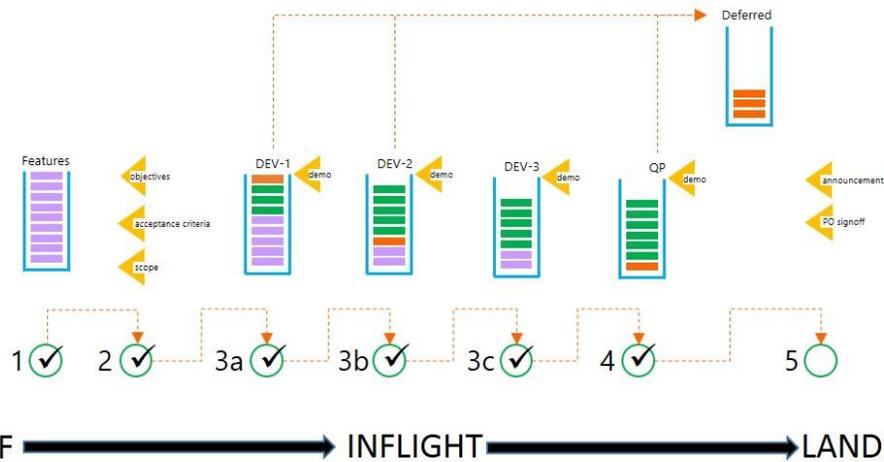


Figure 87. Simplistic view of Product Owner's involvement during a part-time flight.

As we've discussed earlier, the Product Owner determines the ① features and owns the backlog, ② defines and tracks the objectives, acceptance criteria, and scope, ③ tracks and collaborates with the team during 1–3 development sprints, and ④ validates and accepts the quality bar. In addition, the Product Owner may defer features and associated stories or remove them from scope entirely while gearing up for the release.

The Product Owner "owns" the solution release before takeoff, during its flight, and after landing. It is a critical role and often referred to as the "soul" of the project. The "bits and bytes" stop with the Product Owner! No "soul," no "pulse," means no shippable release.

Step ⑤ is ceremonial which we will discuss shortly.

### When should we release?

The more difficult question, and the reason we pause after step ④ in the previous diagram, is *when* do we land the flight to "ship"? At this stage, our team is ready to ship, confidence is high, and excitement is rising with the ominous drumroll.

The next important responsibility of the Product Owner is to synchronize with the stakeholders and ensure that the landing will have an optimal impact for the organization and community. Some of the important considerations include:

- The system-support team is ready and capable of supporting the new release.
- The release will not have a negative impact on any other imminent release.
- The community is not currently swamped with events or other product releases.
- The community is not overwhelmed with too frequent and too many releases.

Optimally, the "when" is before or coincides with the end of the sprint (Figure 88).

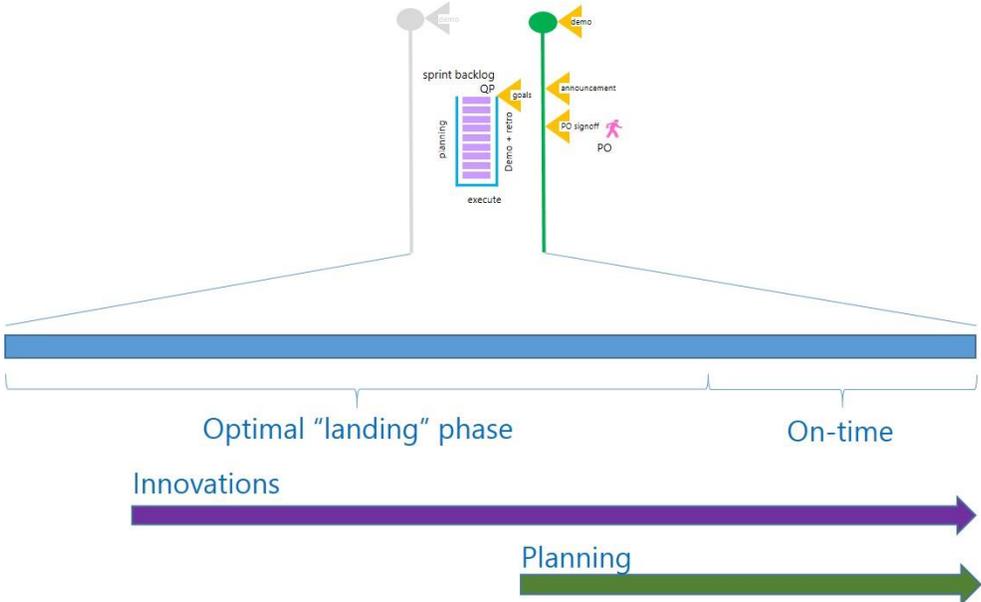


Figure 88. The sooner the team has a shippable release, the sooner it can focus on innovation and planning.

As shown, we have three options for the Product Owner: ① ship early (optimal), ② ship on time, or ③ postpone and place the flight into a holding pattern if shipping during the sprint is not optimal.

Do we ship a beta or an RTM?

In [“Deliver on demand with silent releases”](#), we discussed the concept of one or more silent releases of a beta. This is a good time to take a deep breath, take a step back, and look at the various maturity milestones used by our development teams (Figure 89 and Table 10).

We view the milestones and the workflow as a practical example, not as practical guidance, as *you* need to find and define a workflow that works for you and your environment.

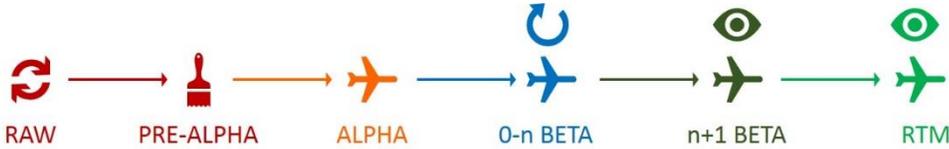


Figure 89. Solution maturity milestones.

Milestone	Description	State of the nation
Raw	Refers to all activities performed during research and prototyping, typically performed during the sprint(s) focused on training-research-planning.	Unknown
Prealpha	Refers to all activities performed during the development sprints delivering features but not including testing and reviews.	Unknown
Alpha	The first phase in which to begin and establish testing and reviews, with the knowledge that we are working with an incomplete feature and unstable solution. The team may decide to make an internal-only, private alpha release to obtain early feedback from the team and a “restricted” community.	Unstable, feature, incomplete, and likely to crash.

Milestone	Description	State of the nation
Beta "silent"	This phase is when the solution is feature complete but infested with impediments and bugs. While working on bug resolutions, stabilization, and overall quality improvements, the team may decide to release one or more internal-only, private, and silent beta preview releases to expose the solution to a selected audience and get real-world feedback.	Stable, feature complete, and unlikely to crash.
Beta  "public"	The Product Owner signs-off on the quality bar and acknowledges that the solution is feature complete and ready for release as a public beta. The intent of a public BETA is to release the solution as is and show it as landed or shipped. Potential users are made aware that they are entering beta territory and there could be nonbreaking changes.	Minimum quality bar met.
RTM 	The release to manufacturing (RTM) or "golden CD" milestone is the milestone when the team has completed all activities, improved quality as much as possible, and resolved all bugs assigned to the release. As with the public beta, the solution is shown as landed or shipped.	Minimum quality bar met or exceeded.

Table 10. Maturity milestones.

Now that we have a better understanding of the milestones, we can return to the question: "Do we land a beta or an RTM release at the end of the sprint focused on quality and planning?"

It depends! If the Product Owner is happy to sign off on the solution and it meets all the release requirements, landing an RTM is definitely the preferred option during or at the end of the sprint. If, however, the minimum quality bars have been met, the Product Owner is happy to sign off, but there are still ongoing activities, we can consider a public BETA at the end of the sprint. This allows us to be in a position to show the project as "landed" and release resources to other adventures. The remaining activities can then be concluded by a skeleton team, after which the public beta is replaced with the RTM release with minimal ceremony.

We frequently use the public beta strategy with guidance projects if the copyediting is bottlenecked or expected to take longer than the sprint. While this may not be an ideal or typical scenario, it enables the team to deliver its solution to the community.

### "Ship-it" or "landing" preparations

There is no way to predict exactly when and how a flight will land. To be honest, there is no real need to predict accurately for part-time community projects; just ensure that the team is ready to ship when the Product Owner gives the signal. Proactive preparations we include as part of a sprint focused on quality and planning include:

- Prepare the public announcement, aligning it with Product Owner, marketing, and team expectations.
- Prepare the release vehicle, for example on [CodePlex](#),<sup>87</sup> including the home page and downloads page.
- Prepare the team to be in a position to monitor and react to feedback, issues, and other events.
- Validate that announcements and documentation mention and credit all team members.
- Validate that all processes—that is, the quality essentials—are ready for completion.
- Gather and analyze flight metrics to measure, reflect, learn, and improve predictability.

Once the team starts to relax and "idle," we are in a good position to spin up parallel activities to innovate and plan for the future. See "[General templates](#)" for an example of a checklist.

<sup>87</sup> [www.CodePlex.com](http://www.CodePlex.com)

## Innovate for next time

The focus on quality and planning is an ideal opportunity to commit spare bandwidth to research and innovation. We empower the team, inspire the individual, and fuel the passion for technology by enabling the exploration of new technology, processes, and techniques. Technology “play time,” technology garages, “hackathons,” brown-bags, and other exciting mechanisms all foster unrestricted innovation, learning, and information sharing.

While we should align the overall innovation theme with the current and future solutions, we need to be careful not to be restrictive or autocratic, inadvertently stifling the “fun” factor.

*Why would anyone invest his or her invaluable personal time to volunteer to “work” in a dictatorship?*

## Planning what is next

The second pillar of the sprint focused on QP concludes the final sprint with a focus on determining whether a subsequent release is needed and starting to plan it.

Planning the next release will be part of the next sprint(s) focused on training-research-planning. At this stage, we can revisit the core objectives and brainstorm future features that could potentially add value or address other known business problems.

As in previous sprints, the planning forces the team to “sit on their hands” (Figure 90), reflect on the past release, and openly discuss the future.



Figure 90. If all else fails, serve pizza!

Observation of numerous projects has shown that many volunteer team members go dark and vanish during the final sprint, when everyone is just waiting for the flight to land and the ship-it event to trigger.

Both innovation and planning help maintain the team’s spirit and ensure that its heartbeat is retained until the next adventure dawns. Keeping the team busy and informed is important during all sprints but paramount during and especially toward the end of the sprint.

## Product Owner sign off

The final ceremony occurs when the Product Owner signs-off on the features, sets the feature state to Done, and gives the team a thumbs up to ship the solution.

Whether the team, a system team, or a centralized support team ships the solution varies from organization to organization. In our context, the ship-it event typically triggers the following actions:

- Solution uploaded to [CodePlex](http://www.codeplex.com)<sup>88</sup> as a download.
- Stable main branch of the solution is branched to a release branch in version control.
- [Solution catalogue](http://aka.ms/vsarsolutions)<sup>89</sup> is updated to reflect the solution and associated version.

<sup>88</sup> <http://www.codeplex.com>

<sup>89</sup> <http://aka.ms/vsarsolutions>

- Bulletin boards showing the flight plan are refreshed to highlight the landing.
- An announcement is published.

## Announcement and noisy release

The style of announcement is an organizational, team, and often personal preference. Consistency across flights is an advantage but not a necessity, especially in the world of community projects.

Figure 91 shows an example announcement using the blogging channel:

### Practical guidance for TFVC and TFS on Azure IaaS

Willy-P. Schaub 15 May 2014 3:21 PM 0

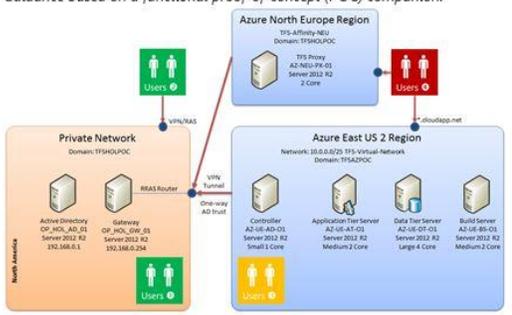
The Visual Studio ALM Rangers are pleased to highlight new and updated practical guidance that landed recently.

TFS Planning, DR avoidance and TFS on Azure IaaS v1.4.BETA has shipped!



Downloads

The new guidance now includes practical, scenario-based guidance for the implementation of Team Foundation Server (TFS) on Azure Infrastructure as a Service (IaaS). We guide you through the planning and setup, based on a real-world proof-of-concept production deployment and experience from the ALM Rangers “in-the-field”.



TFS Planning Guide has been **updated**

TFS on Azure IaaS Guides are **new**

Search MSDN with Bing

Search this blog Search all blogs



Subscribe



Comments



Contact

**Menu**

- [Home](#)
- [Atom](#)

**Tags**

- [ALM Debugging](#)
- [Diagnostics Guidance](#)
- [Rangers Team](#)
- [Foundation Server](#)
- [Testing TFS TFSservice](#)
- [Visual Studio 2010 Visual Studio ALM Work](#)

More ▾

**Videos**

- [Continuous Delivery](#)  
over 3 years ago by Jez Humble
- [ALM in the Cloud](#)  
over 3 years ago by Doug Neumann

**Archives**

- [May 2014 \(9\)](#)
- [April 2014 \(24\)](#)
- [March 2014 \(14\)](#)
- [February 2014 \(15\)](#)
- [January 2014 \(8\)](#)
- [December 2013 \(13\)](#)
- [November 2013 \(10\)](#)
- [October 2013 \(19\)](#)

Version Control guidance v3.BETA flight has landed



Third version of this blockbuster guidance has been split into separate topics as summarized below, allowing you to pick the “world” (guide) you are interested in. This release delivers a new crisper, more compact style, which is easier to consume on multiple devices without sacrificing any content. The content is updated and aligns with the latest Visual Studio technologies and incorporates feedback from the readers of the previous guidance.

Downloads

The guidance uses “friendlier” strategy names. For example, the new **Release Isolation** name is more meaningful than the **Basic – Single Branch** used in the past.



Figure 91. Blog based announcement for two beta releases.

Once we have landed and shipped the solution, marketing, visibility, and “noise” are important. A solution accompanied by passionate blogging, videos, and vibrant community members will be infectious. Hyperactive community members sell community solutions more than professional and expensive marketing.

## Why do we ship on CodePlex and not VSO?

**NOTE** This part of our process will evolve with Visual Studio Online, allowing us to manage and release open source solutions on Visual Studio Online.

As Visual Studio Online introduces guest access and the ability to publish solutions publicly and accept feedback and contributions, we will retire our presence on CodePlex. Our “dream” is that one day we can accept feedback and contributions from the community and you by using a pull request to request a feature update.

## Dogfooding case study: vsarVersionControl and vsarTreasureMap ship ringing victory bells differently

Wrapping up and landing a release is the last part of the adventure. No project ever finishes the same way as its predecessor or other solutions. There is simply no perfect recipe for landing the solution on time, above the quality bar, and with 100 percent satisfied customers. In this case study, we explore two projects, each with a spectacular and successful landing but using a completely different strategy.

### Background information

We introduce the ALM Readiness Treasure Map team in the [case study](#). We welcome this team back for the quality and planning sprint. Another team, which did not dogfood the innovations covered in this book, ended their lifecycle focused on quality and planning. These teams are great real-world examples to include as part of this last case study. The [Version Control \(ex Branching and Merging\) Guide](#)<sup>90</sup> had been bogged down with three or four guidance e-books complete, with the fourth failing to achieve the minimum quality bar. A “Judge Dredd” decision was to spin up a new release for the ill-fated fourth guidance e-book and dogfood the quality and planning focused sprint concepts for the others.

### Different strategies, same objective . . . “land”

The lifecycle of both the teams and their solutions shown below the quality and planning bar in Figure 92 illustrate that neither team stuck to the “optimal” plan. Both team adapted and shipped, and this introduces an important concept: *Reflection, continuous improvement, and a willingness to take risks and embrace change are instrumental to our success!*

---

<sup>90</sup> <http://aka.ms/treasure18>

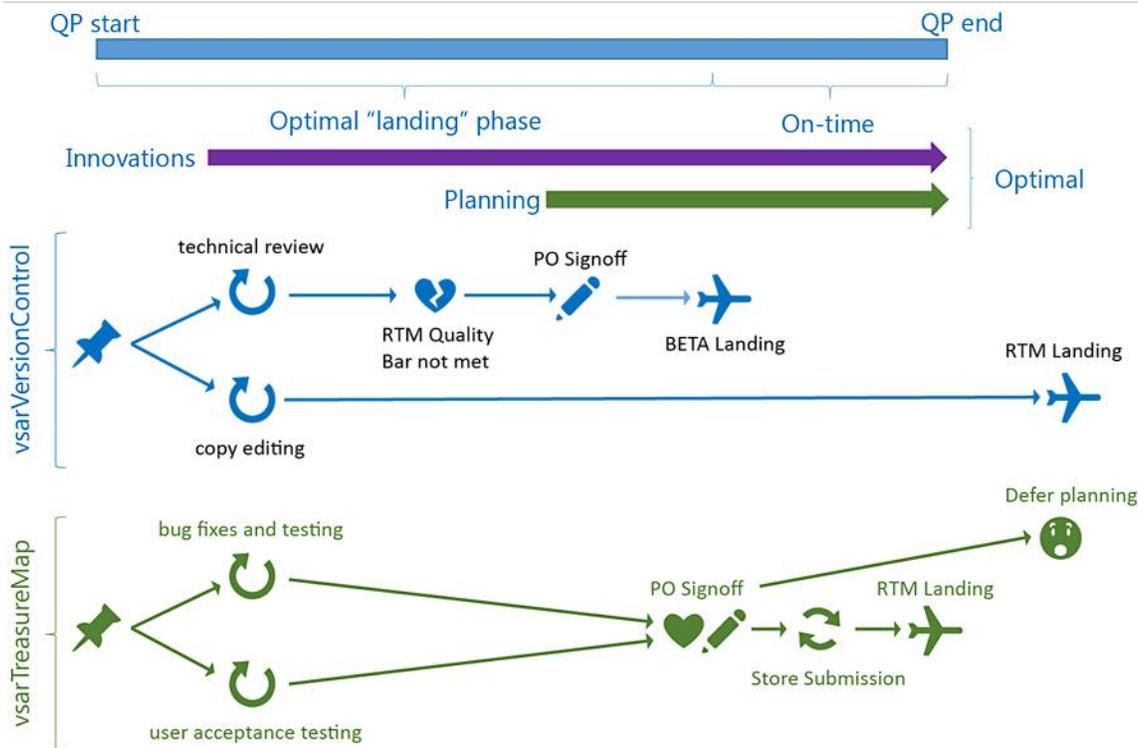


Figure 92. Different sprints focused on quality and planning strategies.

Here is their story ...

## Readiness treasure map

The pirates spun off two quality-improvement activities, one focused on fixing known bugs and testing the updates, and the second on user-acceptance testing. Every team member, or in this case shipmate, was engaged with testing the solution, giving continuous feedback to the developers and responding to the chief tester, Darren, who likely created more and more-complex test cases than the team testing the Eagle landing module for the first moon landing. Innovation occurred as part of bug and test feedback analysis and resolution, with every shipmate engaged and focused. After the Product Owner sign-off ceremony, the solution was put through the store certification process and shipped as a RTM v3 in time for the internal TechReady 19 event and yet another "fun" demo session.

An obvious anomaly is the lack of noticeable planning, intentionally deferred to the TechReady 19<sup>91</sup> session where the captain engaged the audience in feature and future-direction planning. This is an innovative, transparent, and engaging concept. ARRGH!

## Version control guide

The team also spun off two parallel activities, one focused on a technical review and the other on a copyedit to align the three guides in terms of consistency, tone, and style. When the technical review was complete, the team was ready to ship, but the quality bar required the copyediting to be completed.

With the copyediting queue bottlenecked and estimated completion dates far exceeding the end of the sprint focused on quality and planning, the team made a call to ask the Product Owner to sign off and give permission to ship a public beta to avoid further delays in sharing the e-book with the community. Each e-book was transferred

<sup>91</sup> TechReady is an internal technical event at Microsoft and prime opportunity for the Rangers to connect with the field engineers.

to the public download channel as it emerged from the copyediting process, and the release state switched from beta to RTM with the last e-book.

An obvious anomaly is once again the lack of noticeable planning, deferred to the associated project dealing with the ill-fated fourth e-book.

## Innovate elsewhere

Both teams clearly did not focus on the optional innovation and research during the sprint focused on quality and planning. Focus was on improving the quality of both solutions, landing something, and delivering tangible value to the community. While the pirates will likely continue innovations and research while waiting for the next release, the Version Control guidance team members dispersed to other projects, for example DevOps and DSC, where they embraced research as part of the sprints focused on training-research-planning. Innovation and research during the sprint focused on quality and planning is optional and should not distract from the core objectives: *Improve the quality and land the flight!*

## Keep everyone busy and informed

Members on the Version Control guidance team were not busy during the sprint focused on quality and planning. The solution was essentially feature complete, with no subsequent releases planned. Consequently, there was a gradual resource drain as team members moved to new projects, for example vsaDsc. While not an issue in this instance, it provides a stark reminder that idle and bored team members will jump ship as new and exciting project opportunities emerge.

In contrast, we infected the Treasure Map team with a “fun” pirates theme and a passionate Project Lead (PL) and kept the team busy with testing (Figure 93). Not all team members react to such a “fun” factor and a theme like “pirates,” but in this case, we created a vibrant crew that never seemed to sleep. A key is to keep the team engaged and aligned with the vision and objectives as we finalize the release.



Figure 93. A fun theme helps!

## Make noise

When the teams shipped, whether a beta release or RTM, they made noise, announced the release and constraints, and most importantly asked everyone to do likewise. The passionate and vocal team demonstrated commitment and created infectious interest in the solution.

Figure 94 shows sample blog post announcements and a tweet. The messaging was crisp, transparent, and encouraged the community and users to deliver candid feedback.

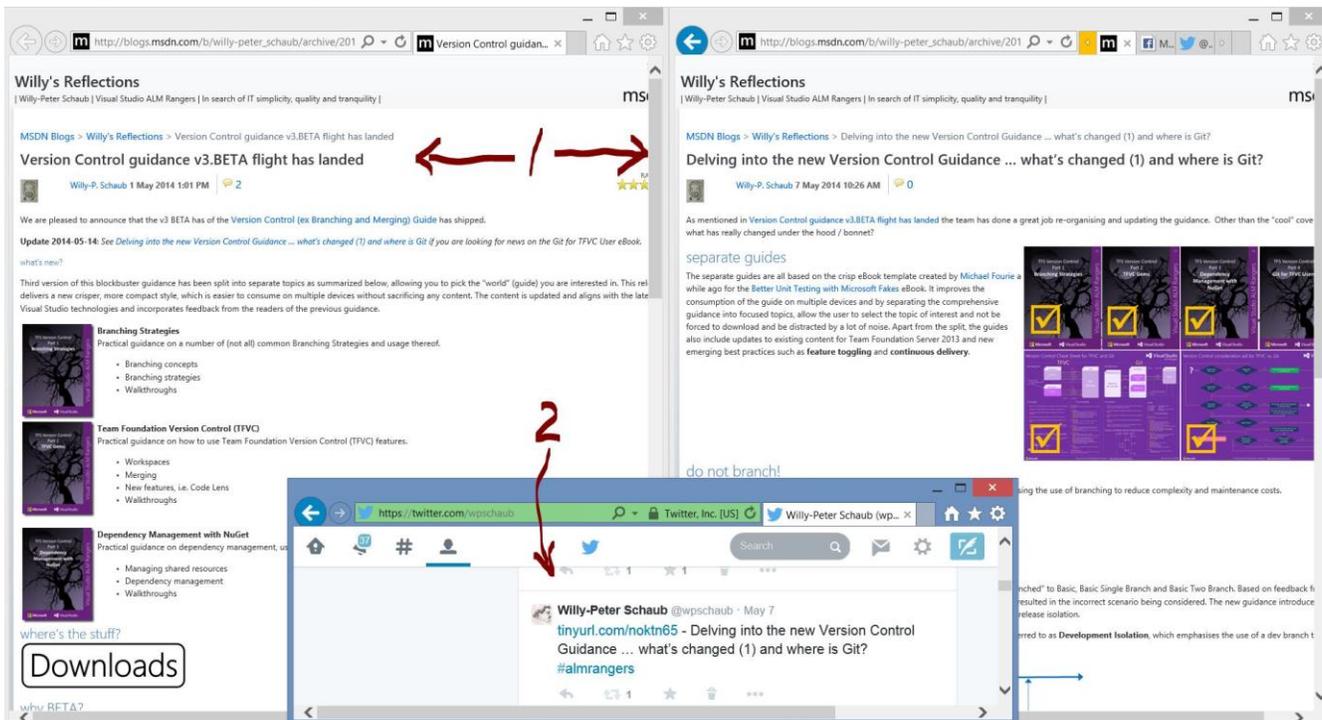


Figure 94. Announcements using blogs, Twitter, and so on.

In addition, the team members were encouraged to add their comments to downloads and announcement blogs, to highlight why they saw the solutions as invaluable, and to instigate vibrant discussions.

## Key learnings

- By releasing a shippable increment as early and frequently as possible, the team can gather candid feedback and be in a position to react to any business value changes.
- The Quality and Planning sprint helps raise the bar. It's the ideal opportunity for the team to research, innovate, and focus on revisiting the core objectives and future features. This increases the team's passion.
- Quality Essentials and copyediting and reviews are important because they ensure that the engineering policies and user documentation is accurate.

# Appendix A: Supporting toolbox

This appendix serves as a quick reference to many of the gems, checklists, and supporting tools and templates we have identified throughout the book. Once you have read the previous chapters, this appendix provides you with a quick point of reference, and you won't have to scan the chapters for the information you are seeking.

**NOTE** We recommend that you read this appendix after reading the previous chapters

## Triage of ideas

### Gems and checklists

Roles, responsibilities, and ownership (summarized here for the triage phase):

Persona	Responsibility	Ownership
Program Manager	Works with the teams, triage group, and leadership to elaborate ideas, participate in planning reviews, coordinate, and drive the ideas triage. Maintains the operational environment and the ideas backlog, produces a report of ideas for the triage group, and helps launch new flights.	<ul style="list-style-type: none"> <li>• Triage group</li> <li>• Triage</li> <li>• VSO environment</li> <li>• Ideas gathering</li> <li>• Pre-triage</li> </ul>
Scrum Master	Mentors the teams, enforces the framework guidelines, eliminates impediments raised during the triage phase, and actively leads the continuous improvement and training efforts.	<ul style="list-style-type: none"> <li>• Process</li> <li>• Framework</li> <li>• Scrum Masters</li> </ul>

**Encourage ideas and feedback** There is no such thing as a bad idea or question. They may, in fact, highlight a very different perspective and produce surprising results and conversation points.

**Capture ideas** Every idea, no matter how small or far-fetched, should be captured. Revisiting an idea could provoke a totally different reaction than when it was logged, perhaps leading to solutions and more ideas that were never imagined at the time. We tag ideas by type and source.

Category	Values	Description
<b>Type</b>	Architecture	Ideas focused on architectural or infrastructure changes that typically are reusable and evolve.
	Business	Ideas focused on business functionality or user experience. This is our default type.
	Ruck	Ideas focused on our "loose-Scrum" framework, as documented herein.
<b>Source</b>	Community idea	Ideas sourced from the ALM communities, typically during events.
	Research idea	Ideas sourced from Rangers who intend to research or future-proof technologies.
	Strategic idea	Ideas sourced from the Visual Studio leadership, marketing, or product groups.
	UserVoice idea	Ideas sourced from the UserVoice Forum.

**Triage ideas** Ideas should be triaged regularly to see how they meet priorities, strategies, and return on investment. Value decays while dissatisfaction grows over time. Ideas that are quickly developed or can be broken down into smaller shippable increments should be favored over ideas that will decay over time. For more information, see [shortest job next \(SJN\) and shortest job first \(SJF\)](#).<sup>92</sup> *Knowing your limits is an important metric when triaging ideas and selecting your next adventures.*

#### GEM **Top ideas**

The best candidates are those with the highest business value that can be delivered most quickly, and impose the least risk to the main product line and mission statement.

**Service mode** Do not drag the world with you. Sometimes ideas and projects need to be put in “service mode” because they no longer add value that aligns with strategy.

#### GEM **Top service-mode candidates**

Good service-mode candidates are those with declining business value, that require maintenance investment, and that are misaligned with the main product strategy. Do not invest invaluable resources in solutions that no longer add value.

**KISSS** Keep it simple and ship soon!

- Do less, better and quicker! We can always improve and do better next time!
- Share solutions sooner, even if they are in alpha or beta state, to get early candid feedback.
- Empower the community to influence and take over solutions that are entering maintenance-only mode.

**Shared ownership** We speak of having multiple owners and backup leads, etc. A key to success is really growing a “shared ownership” awareness in the deliverable. Once that sets in, the effect of losing a team member is greatly reduced because his or her responsibilities can more easily be taken up by the rest of the team.

**SMART objectives** We define [SMART](#) objectives as:

Property	Description
Specific	Objectives need to be precise and unambiguous. Every team member should be able to explain each objective. There should be no disagreements among the team members.
Measurable	Objectives must be measurable and aligned with expectations and acceptance criteria. The team and Product Owner must be able to decide when they have achieved the objectives.
Achievable	Objectives need to be achievable in terms of timeline and the capabilities of the team.
Realistic	Objectives must be aligned with the vision and relevant to the overall idea. For example, it is unrealistic to expect the ALM Readiness Treasure Map team to deliver version-control guidance or release-management tooling. Such expectations will result in frustration and failure.
Time-based	We must set the completion date for our objectives within a specified time box and align these dates with the overall roadmap. We naturally prioritize objectives that we must deliver by a specified date (when) over those we can be deliver any time (whenever).

<sup>92</sup> [http://en.wikipedia.org/wiki/Shortest\\_Job\\_First](http://en.wikipedia.org/wiki/Shortest_Job_First)

**INVEST in features** The [INVEST](#)<sup>93</sup> acronym is useful for managing features that owners identify for the team to invest in during the next and optionally future releases of the solution.

Property	Description
Independent	Self-contained, loosely coupled, and with tight cohesion.
Negotiable	Until a feature is committed, we can renegotiate and revise it.
Valuable	Feature must add value.
Estimable	Feature must align with our estimation strategy.
Small	Size of the feature must be realistic and fit into the iteration cadence and release milestone.
Testable	Feature must have clear acceptance criteria that we use to create test cases and enable sign off of the feature as done.

**Time box and limit work in Progress (WIP)** We have found that by using time boxes and limiting work in progress, we reduce the strain on project management and typically deliver smaller bits of value more rapidly. Always keep in mind the tradeoff triangle.



## GEM

### One to three features per flight

The number of features defined per Epic depends on the granularity of each feature. We recommend defining one to three “must-have” features and optionally one to two “exceeded” features. Defining more features disperses the team’s focus, complicates coordination, and introduces the potential of scope creep.

**Identify passionate owners** We typically utilize a Product Owner, Project Lead, and Program Manager, but we feel free to combine roles where necessary. Based on our dogfooding observations, we recommend that:

- A team consists of one or two feature teams, each with up to seven feature team members.
- A Project Lead owns one in-flight project release.
- A Program Manager is responsible for one to three in-flight project releases.
- A Scrum Master is responsible for one to three in-flight project releases and serves as a potential emergency backup for another Scrum Master and associated teams.

**Visibility is key** Make use of dashboards to provide status and trend information.

<sup>93</sup> [http://en.wikipedia.org/wiki/INVEST\\_\(mnemonic\)](http://en.wikipedia.org/wiki/INVEST_(mnemonic))

# Getting ready

## Gems and checklists

**Training, research, and planning (TRP)** These are the core pillars of success for our projects. Train the team; research the technologies and requirements; understand the estimation and prioritization planning fundamentals. The focus on training-research-planning allows the team to understand and optionally refine the project:

- Motivation (why)
- Vision (what)
- [SMART](#)<sup>94</sup> objectives
- Features
- Acceptance criteria
- Definition of done

**Know your destiny and provide context** To get a team onboard, we must be able to communicate the end goal and provide context so that team members can engage in the adventure, share the understanding, and be motivated to join the team.

### **GEM** Provide context . . . but not too early!

It is valuable to enable the attendees to prepare themselves by sending them context for the meeting a few hours beforehand. Typically, we share the kickoff meeting slides in PDF format for preview and in the event that the teleconference infrastructure acts up during the meeting. Do not send out the information too early. Otherwise, you may influence the attendees and their prioritization accordingly.

**A3 problem solving** This simple technique can help to clearly define the problem, current situation, objectives, and target outcome.

**Transparency** The whole team should be able to see the requirements, plan, and status so that team members remain engaged and committed to working through the backlog.

**Infrastructure** We need to plan our technical and human infrastructure well to ensure that it works effectively. Technically, we may use an on-premises instance of TFS or decide to use Visual Studio Online. Regardless, we will need to define the team project and team layout, including source control.

**Estimations** These will be the most difficult to calculate. Refer to [Chapter 2](#) for several ideas and scenarios.

**Dependency isolation** Reduce dependency and enable isolated activities. Loosely coupled and highly cohesive stories reduce dependency on other team members and specialized knowledge. Reduced dependency enables teams or team members to work in isolation.

**Ship value quickly** Ship value to the user as soon as possible. Focus on stories that are of high value to users, not us! Ship working solutions as soon as possible. Focus on the smallest stories first and minimize cost of delay!

**Surveys** For teams that are geographically dispersed and that may not meet every day, a quick survey is a great tool to help stimulate discussion when the team meets or to provide guidance if a team is blocked.

**Organize your team** After hosting the kickoff meeting, decide whether we will need one or multiple feature teams.

### **GEM** $MAX = (3-9)n + 2$

The recommended team size is  $6^{+-3}$  development team members per feature area (n). Moreover, the Scrum Master and Product Owner span all feature areas of the one release identified by the Epic. For geographically distributed teams, we recommend that  $n \leq 2$ .

<sup>94</sup> [http://en.wikipedia.org/wiki/SMART\\_criteria](http://en.wikipedia.org/wiki/SMART_criteria)

- [TFS Planning and DR Avoidance Guide](#)<sup>95</sup>
- [Version Control \(ex Branching and Merging\) Guide](#)<sup>96</sup>

## Templates: Email

### Kickoff (outline)

To: @@

Subject: [ @PROJECTCODE@ ] - Project @PROJECT@ Kickoff Event

Do you have a passion for XYZ? This may be the right project for you.

We will host our kickoff meeting at *TIME/DATE* and encourage you to attend.

#### **PROJECT IDEA**

*BRIEF DESCRIPTION.*

(Link to external influencers if any, e.g. UserVoice)

#### **PROJECT LEADS**

*YOUR NAME*

*PO NAME*

*ANY BACKUPS IF KNOWN*

#### **RECORDING**

We will record this session but would love to have you online for the kickoff.

Thanks - *YOUR NAME*

### Kickoff agenda (outline)

#### **AGENDA**

- *Introduction of Product Owner, Program Manager, and Scrum Master*
- *Introduction of the problem and proposed solution*
  - *Problem statement*
  - *Current reality*
  - *Objectives*
  - *Target outcome*
- *Next steps*

---

<sup>95</sup> <http://vsarplanningguide.codeplex.com/>

<sup>96</sup> <http://vsarbranchingguide.codeplex.com/>

## Release planning (outline)

To: @@

Subject: [ @PROJECTCODE@ ] - @PROJECT@ Release Planning

### **Motivation**

Reinforce why the solution is important and why it is important for the team to care.

### **Vision**

Reinforce what the solution we intend to build solves, outlining architectures, technologies, and core deliverables and highlighting the key benefits.

### **Roadmap**

Reinforce when the solution teams need to achieve key milestones.

### **Objectives**

Reinforce the Specific, Measurable, Achievable, Realistic, and Time-based objectives for the solution release and individual sprints.

### **Nonfunctional requirements (NFR)**

Mention all known NFRs, such as scalability, reliability, quality, etc., which we need to consider during the planning.

### **Features**

Reinforce the idea and the prioritized features, which the team needs to expand to stories and optionally to tasks during the planning. The Product Owner (PO) should start approving all features, which must be included in the solution release.

### **Process**

Reinforce the process and associated frameworks used by the solution team(s), ensuring that every team member knows where to find process guidance and documentation.

### **ASKs**

Summarize the key ASKs from the team that need to emerge from the offline collaboration. Limit it to seven (7) ASKs max!

### **URLs**

Summarize the key URLs that point at the team's home page, kickoff meeting recording(s), documentation, source control, and other artifacts.

Thanks - *YOUR NAME*

## Welcome (outline)

To: @@

Subject: [ @PROJECTCODE@ ] - Welcome to the @PROJECT@ team

We would like to welcome all of you to @PROJECT@ and the focus on training-research-planning (TRP)!

Thank you for joining this project! It is going to be a challenging prototyping project, but we have a ROCKSTAR group, and I expect that this should be an interesting prototype! There were quite a few people sent to other projects, so let me know if there are any conflicts or reasons you think you may not have the time.

### Infrastructure

- **HOME** @@URL
- **TFVC** @@PATH
- **CVTAG** @@TAG
- **DL** @@DL URL

### Team

We would like to introduce everyone on the team so that we are all familiar with each other:

#### Core Team

- @@List of names

#### Contributor Role

- @@List of names

#### Reviewer Role

- @@List of names

#### Silent Reviewer Role (shipmates who are still engaged on other projects)

- @@List of names

### Next Steps

- Meetings
  - We may send out a meeting invite for next week to have a group discussion ... in the meantime collaborate by email.
  - We will send out a meeting invite for the last week of @@month to plan the scrum series and to estimate the work as a team.
- Please peruse the scrum innovations that we will use on this project
- Please read and understand our core features for this prototype . . . we can always do more next time:
  - @@Features
- Comment on our proposed objectives:
  - @@Objectives
- Comment on our proposed release plan:
  - @@Roadmap

Thanks - *YOUR NAME*

## Team limit reached (sample)

Dear Volunteer Name,

Unfortunately, we have reached the practical team-size limit for this project. I want to thank you for attending the kickoff and volunteering. It was a tough choice, and I would like to keep you as a backup if anyone drops, or you are free to jump to another project below.

The “Extract effective permission from TFS for audit purposes” project just kicked off, with two more strategic projects in the pipeline:

- Desired State Configuration (DSC) Guidance and Scripts Library to support DevOps
- Migration guidance of on-premises TFS to VSO

This tough dilemma was the reason for today’s [I volunteered for a project but am booked on another voyage . . . why](#)<sup>97</sup> blog post from Willy.

Please do not hesitate to contact Willy or me if you have any concerns or candid feedback.

Regards - *YOUR NAME*

## Planning meeting (sample)

**To:** vsarSecurity

**When:** March 31, 2014 12:00 PM-1:00 PM (UTC-08:00) Pacific Time (US & Canada).

**Where:** Lync Meeting

### AGENDA

- Overview
- Agree on weekly scrum ... day + time.
- **Estimate** the stories on the backlog
- Agree on **DEV-1** stories
- Confidence **vote** (1 finger = ☹, all-five fingers = ☺ we will do it!)
- Agree on ONE **innovation** we would do in terms of the process, if we could do this sprint again.

### PREPARATIONS

- Install, configure, and test the TFS Agile Poker application
- Review the release backlog, using the \shared queries\ATPoker\vsarSecurity query

Thanks - *YOUR NAME*

<sup>97</sup> [http://blogs.msdn.com/b/willy-peter\\_schaub/archive/2014/03/14/alm-ranger-projects-i-volunteered-for-a-project-but-am-booked-on-another-voyage-why.aspx](http://blogs.msdn.com/b/willy-peter_schaub/archive/2014/03/14/alm-ranger-projects-i-volunteered-for-a-project-but-am-booked-on-another-voyage-why.aspx)

## Scrum notes (sample)

**To:** vsarSecurity;

**Subject:** [ vsarSecurity ] Scrum 20140414

### Scrum Notes

We are making great progress!

9731 Scrum 20140414

### Remember our objectives!

- Understanding and mapping of security namespaces and our scope, BV:8 (BV-Business Value)
- Understanding of auditing requirements and options, BV:5
- Report outlining security mapping, ideas, and available tooling delivering permission reporting, BV:3
- Report outlining auditing requirements, ideas, and available tooling delivering auditing reporting, BV:2
- Functional prototype demonstrating extraction of effective permissions and creation of XML report, BV:2
- (Exceeded) Functional prototype demonstrating security auditing and reporting, BV:1

### ASK #1: TR19 - Session

Here is a proposed session title, description, and objectives. Can everyone please reply all with candid feedback for Jon? Jon, can you use the feedback and submit the session before the end of next week please?

- **Title**  
Understand the challenges of unraveling effective permissions in TFS
- **Description**  
With increased proliferation of Team Foundation Server in the enterprise, we are often asked to determine the effective permissions of a user for auditing purposes. Sounds easy! This interactive discussion shares our research, the tooling available to assist you with the investigations, and gives you an opportunity to share your requirements.
- **Type and Level**  
Chalk Talk, 300
- **Objectives**
  - Understand the challenges of determining effective permissions across all security namespaces.
  - Describe the tooling available to help me determine effective permissions.
  - Share your permission and auditing requirements and experience from the field.

Thanks - *YOUR NAME*

## Templates: Survey

Figure 95 shows an example of an online planning survey, and Figure 96 shows an example of an online retrospective survey.

## Planning online

### Visual Studio ALM Rangers - vsarSecurity Planning S27-28

Estimate stories on the S27 - 28 sprint backlogs



Make sure you understand "normalised estimations" as used by the ALM Rangers before your estimate. Also our motto is to "ship today and do better tomorrow", in other words estimate based on practical deliverable ... not the PERFECT planet vision.

If you have ANY questions ping Willy-Peter Schaub. You can always return to the survey and edit changes and continue making estimates.

You can revert to the team backlog for more detail, acceptance criteria and DoD defined in the parent features.

1. FEATURE AREA #1: Security Auditing Research Findings and Recommendations

	0	1	2	3	5	8	13	?
Research Impact of Auditing Samples on TFS performance	<input type="radio"/>							
Research Other Options	<input type="radio"/>							
Research Recommendations	<input type="radio"/>							
Supporting Whitepaper	<input type="radio"/>							

Figure 95. Online planning survey example.

## Retrospective online

### Visual Studio ALM Rangers - vsarSecurity Retro



\* 1. Overall, how effective was this sprint?

- :- Great  
 :- OK  
 :- Bad

\* 2. I think that using a Survey for Retrospectives is useful?

- Yes  
 No

3. What went well during this sprint?

4. What didn't work during this sprint?

5. What changes, if any, would suggest to make things better within this team, or overall?

6. OPTIONALLY specify your email address, to allow us to contact you if we have questions on your feedback. Completing this field breaks ANNONIMITY of your survey feedback

Submit

Figure 96. Online retrospective survey example.

## General templates

### Acceptance criteria (Context: Customer == Product Owner)

#### Sign off

- Product Owner must sign off on a feature before it is closed or the deliverable shipped.
- Product Owner must approve all design changes.
- All code and documentation is stored in team repository.

#### Bugs

- Bugs must be triaged and associated with a feature.
- Bugs must be resolved or cut, not carried over.

#### Code

- Code must be peer reviewed.
- Code must be compliant with the minimum quality bar.

#### Build

- Build and unit test automation must pass.
- Build automation must use a gated check-in build.

#### Test

- Unit tests must pass and achieve 80 percent or higher code coverage.
- User acceptance testing by testers must pass.

### Definition of done (DoD)

- Transparency and simplicity is important.
- As a team, agree on what “done” means within a specific context and within a given sprint.
- See [Done and Undone](#)<sup>98</sup> for more details.

#### Context examples:

- Guidance documentation
  - Must be checked for spelling and grammar before submitted for review.
  - Must be peer reviewed.
  - Must be copyedited before submitted for integration.
  - Must be compliant with sensitive geopolitical terms and phrase guidelines before being released.
- Source code
  - Must be peer reviewed.
  - Must be documented.
  - Must include unit tests with a minimum 80 percent coverage.
  - Must pass a build and test run before committed to central source control.

---

<sup>98</sup> [https://msdn.microsoft.com/en-us/library/hh765983\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/hh765983(v=vs.110).aspx)

## Ship-it checklist

Step	Description	Time (h) estimate	Owner <sup>99</sup>
1	Complete early adopter programs.		PM
2	Select license agreement.	0.5	PM
3	Verify that all quality bars are met.	2.0	QL
4	Verify that the team's definition of done is met.	0.5	PM
5	Verify compliance with sensitive geopolitical terms and phrases in product and documentation.	0.5	PM
6	Verify that product and code contains no obsolete versions.	0.5	PM
7	Verify compliance with API usage standard of interoperability policies.	0.5	PM
8	Obtain permission from the Product Owner to ship.	0.5	PM
10	Sign the binaries.	1.0	DL
11	Revise the release notes	2.0	DL
13	Release the solution.	0.5	PM
14	Unless released silently, announce the release.	0.5	PO

[https://msdn.microsoft.com/en-us/library/hh765983\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/hh765983(v=vs.110).aspx)  
ad, **QL**=Quality Lead

## Work items

### Epic

#### Title examples

- Unit Test Generator Extension for Visual Studio
- Upgrade the Test Release Management Guidance
- Adopt PowerShell Desired State Configuration

#### Description

<b>Value statement</b>	<b>for</b> <user> <b>who</b> <does> <b>the</b> <solution> <b>is a</b> <the how> <b>that</b> <value> <b>unlike</b> <competition>
<b>Source</b>	Where is the Epic idea from and who is the contact.
<b>Acceptance criteria</b>	When will the Product Owner be happy to ship this Epic?

### Feature

#### Title examples, both linked to Epic: Adopt PowerShell Desired State Configuration

- Practical Guidance for PS DSC for DevOps and ALM Practitioners
- Create a PS DSC Resource Factory
- Create Quality Resources for the PS DSC Resource Kit

#### Description

What	Describe what the feature is all about, linking it back to its parent Epic.
Benefit	Describe the benefits and value that the feature delivers for the user.
Acceptance criteria	When will the Product Owner be happy to ship this feature?

### Product backlog item

#### Title

- **As a** <user role, persona> **I can** <do> **so that** <value>

#### Description

What	Describe what the product backlog item (story) is all about, linking it back to its parent feature.
Prerequisites	Describe any prerequisites needed to implement the story.
Acceptance criteria	When will the team be happy to ship this story?

# Building the working solutions

## Gems and checklists

**Heartbeat** The heartbeat and pulse of an Agile team is an iterative and consistent cadence.

**Pull and push** Adopt a pull strategy of stories by team members by default, but be ready to propose and push stories to team members who hesitate to grab stories from the backlog.

**Sprint objectives** Rule the deliverable. These reinforce why we are doing the project, what needs to be done, who will be responsible for what, and how confident the team is about succeeding

**Repetition** Adopt a sprint pattern that is natural to the team and repeat, repeat, repeat. For example:

1. Assume ownership of the sprint challenge as a team--for example, *travel to moon Europa*.
2. Review and refine the objectives, requirements, and the features from the sprint backlog—for example, *transfer W astronauts to the moon, collect X tons and Y types of samples, and return to Terra within Z years*. Verify that we understand the requirements, are in a position to meet the acceptance criteria, and are in a position to proceed with none or manageable unknowns.
3. Plan and design the sprint development journey.
4. Develop the features to realize the requirements.
5. Integrate, test, and validate the features.
6. Reflect on the sprint, identify improvements, and plan the next actions.
7. Demo and ship working solution(s).

**Scrums** These meetings provide the project pulse and team transparency. Schedule a short, effective meeting as often as possible.

### GEMS

#### Weekly heartbeat “rocks”

With geographically distributed, part-time, and virtual teams, a weekly scrum, which adheres to a 15 [+15] minute time box, is recommended.

#### Regular heartbeat “rocks”

With geographically distributed, part-time, and virtual teams, we recommend a once-per-sprint scrum of scrums that adheres to a 2–3 minute per team time box.

**Demos** These are an opportunity for the team, leadership, and stakeholders to synchronize.

### GEM

#### 2–3 minute demo video “rocks”

The testers/reviewers are in a prime position to record their validation tests and produce a video used as a demo and visual documentation. Keep your video to 2 minutes (optimal) to 3 minutes (maximum) for the greatest impact.

**Retrospective** Use the retrospective to gather subjective and quantitative evidence that can be used to improve the sprint process.

### GEM

#### Agree on one innovation

Every team should agree on one innovation that will improve its experience and productivity on the team.

**Planning** Planning the next sprint is part of completing the current sprint!

### GEM

#### Keep your backlog groomed and clean!

Consider your backlog as the main entrance of your project. Keeping it clean and groomed will ensure that first impressions are positive and visits collaborative and productive.

**Grooming** Produce a trim and clean backlog of requirements and better understand what needs to be done, in which order, and by when.

**GEM Groom top down**

Ensure that the Product Owner and stakeholders prioritize all backlog items. Work through the backlog from the top (most important) to the bottom (least important) in the list of backlog items.

## Raising the quality bar

### Gems and checklists

**Quality and planning** We end projects with a focus on raising the quality bar and planning the future.

**GEM Automation “rocks”**

While there definitely is a place for manual testing, test automation allows teams to run more tests, more often and continuously. Carefully evaluate and select the test tooling that suits your team and investigate [When to Test Manually and When to Automate](http://blogs.msdn.com/b/steveowe/archive/2008/02/26/when-to-test-manually-and-when-to-automate.aspx).<sup>100</sup>

**Review process** We have a well-defined review process to ensure that we can gather feedback and track its integration through the deliverable. Make sure the team is aware of the process to ensure that quality is maintained from the start.

**Maturity milestones** Shipping early to targeted audiences helps drive early feedback.

Milestone	Description	State of the nation
Raw	Refers to all activities performed during research and prototyping, typically performed during the sprint(s) focused on training-research-planning.	Unknown
Prealpha	Refers to all activities performed during the development sprints delivering features, but not including testing and reviews.	Unknown
Alpha	The first phase in which to begin and establish testing and reviews, with the knowledge that we are working with a feature incomplete and unstable solution. The team may decide to release an internal-only/private alpha to obtain early feedback from the team and a “restricted” community.	Unstable, feature, incomplete, and likely to crash.
Beta “silent”	This phase is when the solution is feature complete but infested with impediments and bugs. While working on bug resolutions, stabilization, and overall quality improvements, the team may decide to release one or more internal-only/private and silent beta releases to expose the solution to a selected audience and get real-world feedback.	Stable, feature complete, and unlikely to crash.
Beta “public”	The Product Owner (PO) signs-off on the quality bar, acknowledging that the solution is feature complete and ready for release as a public beta. The intent of a public beta is to release the solution as is and show it as landed or shipped. Potential users are made aware that they are entering beta territory and there could be nonbreaking changes.	Minimum quality bar met.

<sup>100</sup> <http://blogs.msdn.com/b/steveowe/archive/2008/02/26/when-to-test-manually-and-when-to-automate.aspx>

Milestone	Description	State of the nation
RTM 👁️	The release to manufacturing (RTM) or “golden CD” milestones is the milestone when the team has completed all activities, improved quality as much as possible, and resolved all bugs assigned to the release. As with the public beta, the solution is shown as landed or shipped.	Minimum quality bar met or exceeded.

## Eating your own dogfood is key

### Gems and checklists

**Visual Studio Online** We run our business using Visual Studio Online. It is improving every three weeks and is truly our technical foundation.

**Reduce complexity to one** We try and keep things simple in our world, and that can often produce much better results than a more complex solution:

- We use one team project collection
- We use one team project
- We all align to one shared sprint
- We follow one process, Scrum

**Collaboration** We promote frequent collaboration to break down barriers and friction. To be effective we must know our team. Do not underestimate the effect of seeing each other; use those cameras in the virtual meetings.

**Process** The Microsoft Solutions Framework forms the basis of our practices and is highly comparable to the Agile Manifesto and Lean principles.

MSF		Agile Manifesto	Lean principles
<b>Principles</b>	<b>Mindsets</b>	<b>Values</b>	
<ul style="list-style-type: none"> <li>• Foster open communication</li> <li>• Work toward a shared vision</li> <li>• Empower team members</li> <li>• Establish clear accountability and shared responsibility</li> <li>• Focus on delivering business value</li> <li>• Stay agile, expect change</li> <li>• Invest in quality</li> <li>• Learn from all experiences</li> </ul>	<ul style="list-style-type: none"> <li>• Familiarize team members to how they should approach solution delivery</li> <li>• Foster a team of peers</li> <li>• Focus on business value</li> <li>• Keep a solution perspective</li> <li>• Take pride in workmanship</li> <li>• Learn continuously</li> <li>• Internalize qualities of service</li> <li>• Practice good citizenship</li> <li>• Deliver on our commitments</li> </ul>	<ul style="list-style-type: none"> <li>• Individuals and interactions over processes and tools</li> <li>• Working software over comprehensive documentation</li> <li>• Customer collaboration over contract negotiation</li> <li>• Responding to change over following a plan</li> </ul>	<ul style="list-style-type: none"> <li>• Eliminate waste</li> <li>• Build quality in</li> <li>• Create knowledge</li> <li>• Defer commitment</li> <li>• Deliver fast</li> <li>• Respect people</li> <li>• Optimize the whole</li> </ul>

We advocate the use of Scrum, Kanban, and SAFe where applicable.

We have several [key personas](#) that provide the glue in our system (Table 11). In some cases their roles may overlap:

Secondary role ▶ Primary role ▼	PO	PL	BL	PM	C	R	M	SM
Product Owner (PO)		✘	☹	☹	☹	✓	☹	☹
Project Lead (PL)	✘		✘	✓	✓	✓	✘	✓
Backup Project Lead (BL)	✘	✓		✓	✓	✓	✘	✓
Program Manager (PM)	☹	✓	☹		✓	✓	✓	✓
Contributor (C)	✘	☹	✓	☹	✓	✓	✘	☹
Reviewer (R)	✘	☹	✓	☹	✓	✓	✘	☹
Mentor (M)	☹	☹	✓	✓	☹	✓		✓
Scrum Master (SM)	☹	✓	✓	✓	☹	✓	✓	

Table 11. A description of primary and secondary roles and where they overlap.

Symbols used:

- ✓ **Good** secondary role for the primary role within a specific project
- ☹ **Feasible** but not recommended secondary role for the primary role within a project
- ✘ **Not feasible** secondary role for the primary role within a specific project. Avoid!

# Tooling

All the tooling we use is free and usually open source:

[Controlled Vocab](http://mikefourie.github.io/ControlledVocabulary/)<sup>101</sup> to organize our emails for subsequent retrieval.

[MyHistory](https://myhistory.codeplex.com/)<sup>102</sup> provides a summary of work item, change set, and shelve set history for personal or colleague activity. It's based on the MSDN article [Extending Visual Studio Team Explorer 2012](http://msdn.microsoft.com/en-us/magazine/dn201751.aspx).<sup>103</sup>

[Flight Plan Board](https://vsarflightplan.codeplex.com/)<sup>104</sup> to visualize the progress and status of our projects.

[News](https://visualstudiogallery.msdn.microsoft.com/6c4d14c3-e66f-40d0-8d77-ff7883a40f6a)<sup>105</sup> delivers Visual Studio Online and ALM Rangers' news and updates inside Visual Studio.

See the walkthrough for more details

The image shows two side-by-side screenshots of Visual Studio Team Explorer. The left window is titled 'Team Explorer - My History' and displays a list of work items, change sets, and shelve sets. The right window is titled 'Team Explorer - Colleague History' and displays a similar list but for colleague activity.

The image shows a screenshot of the 'FLIGHT PLAN: STATUS BOARD' tool. It displays a table with columns for Status, Title, Scheduled, Ver, Product Owner, Project Lead, and Ruck Master. The table lists various projects and their current status.

Status	Title	Scheduled	Ver	Product Owner	Project Lead	Ruck Master
🛠️	Lab Management Guide Upgrade for TFS 2013	2014-02-28	3.0	Vijay Meenraj	Mathias Claesson	Willy-Peter Schaub
🛠️	On-Time VM Factory Service	2014-02-28	4.0	Brian Jassid	Riz Mido	Brian Blackman
✈️	Project Flight Plan (Status) Board	2014-03-15	1.0	Willy-Peter Schaub	Robert MacLean	Willy-Peter Schaub
✈️	Version Control Guidance Upgrade	2014-03-15	3.0	Matthew Mink	Michael Learned	Willy-Peter Schaub
✈️	TFS on Azure (IAAS) Planning and Ops Guide	2014-04-01	1.4	Mario Rodriguez	Willy-Peter Schaub, James Saldry	Jahangeer Mohammed
✈️	Enhance ALM Readiness Treasure Map to address v2 backlog and deliver a v3	2014-06-30	3.0	Willy-Peter Schaub	Robert Bensen	Robert Bensen
💡	SAFE Awareness and Readiness	2014-06-30	1.0	Gregg Boer	Brian Blackman	Brian Blackman
💡	As a TFS Administrator, I would like to extract permission reports to audit purp...	2014-06-30	1.0	Mario Rodriguez	Jon Goatin	
💡	ALM Rangers DevOps / Building a Release Pipeline with TFS 2013	2014-06-30	2.0	Roopesh Nar	Cindy O'Mera	

The image shows a screenshot of Visual Studio with a news article open. The article is titled 'Aligning our planets with a common cadence and synchronization' and is from Willy-Peter Schaub. The article content is partially visible, discussing sprint cadence and synchronization. A blue arrow points to the 'News' tab in the right-hand pane.

<sup>101</sup> <http://mikefourie.github.io/ControlledVocabulary/>

<sup>102</sup> <https://myhistory.codeplex.com/>

<sup>103</sup> <http://msdn.microsoft.com/en-us/magazine/dn201751.aspx>

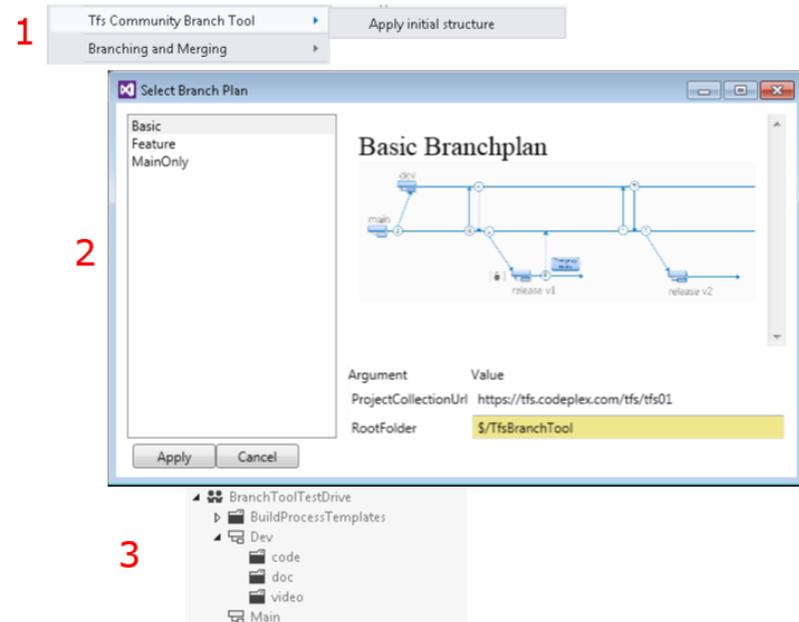
<sup>104</sup> <https://vsarflightplan.codeplex.com/>

<sup>105</sup> <http://visualstudiogallery.msdn.microsoft.com/6c4d14c3-e66f-40d0-8d77-ff7883a40f6a>

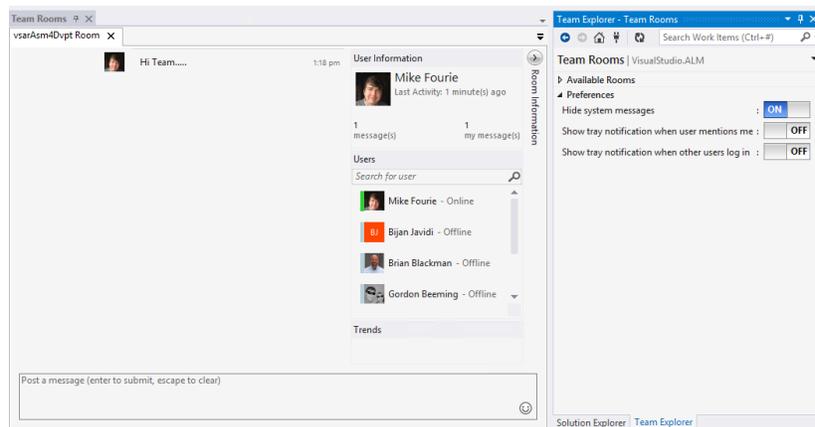
[TFS Agile Poker](http://tfsagilepoker.com)<sup>106</sup> is a cross-platform tool for distributed and colocated teams to *point* the backlog.



[TFS Community Branch Tool](#)<sup>107</sup> to automate branching tasks in the ALM Rangers' solutions.



[TFS Team Rooms](#)<sup>108</sup> provides an enhanced experience of team rooms inside Visual Studio.



<sup>106</sup> <http://tfsagilepoker.com>

<sup>107</sup> <https://visualstudiogallery.msdn.microsoft.com/bc38a262-2522-40bc-ab18-78cab3fd5524>

<sup>108</sup> <http://visualstudiogallery.msdn.microsoft.com/c1bf5e4f-5436-465d-87da-09b2f15ff061>

## Controlled Vocabulary: Walkthrough

Email remains the most common form of electronic communication, and coping with the volume of messages can be challenging. To assist with initial processing and future searching, we recommend adopting a structured format to the subjects of email, as mentioned earlier in this book. To assist with this, use the Controlled Vocabulary Outlook add-in. The quick start below is based on release 1.6.0.0.

### Quick start

Open the Controlled Vocabulary Quick Start folder in the supporting toolkit (part of the downloadable content mentioned in the book's introduction). In this folder, you will find the following files and folders:

**QuickStart** This folder contains the images used in the add-in, as well as the button.xml file containing the makeup of the button and menus you will add and the configuration.xml file that your button will use to auto-update itself.

**QuickStart.cvcf** This is a file used by the add-in to install your button.

**QuickStart.xml** This file has the same content as button.xml. The idea is that you place this on your web server or file share along with any other buttons you have configured.

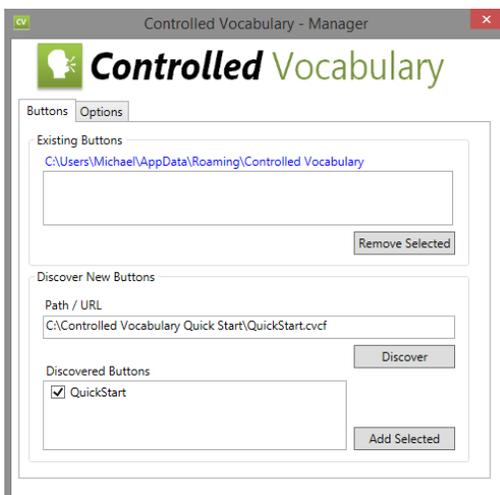
**QuickStart.zip** This is the QuickStart folder zipped up, which the add-in uses to update the button when necessary. Again, this would be placed on the web server or file share you are using to host your resources.

To get the QuickStart sample working, let's use a file share as our host.

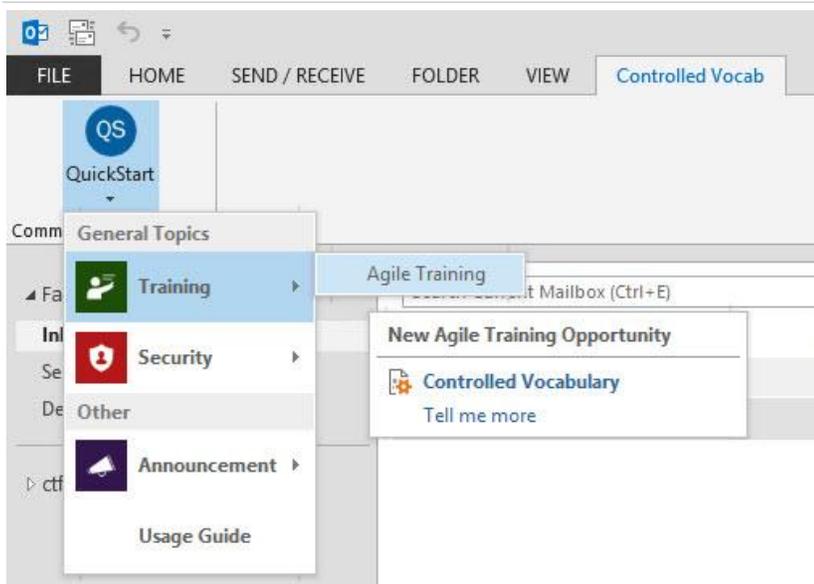
1. Run the following from a command line to create a folder and share it:

```
md c:\QuickStart
net share QuickStart=c:\QuickStart /grant:everyone,READ
```

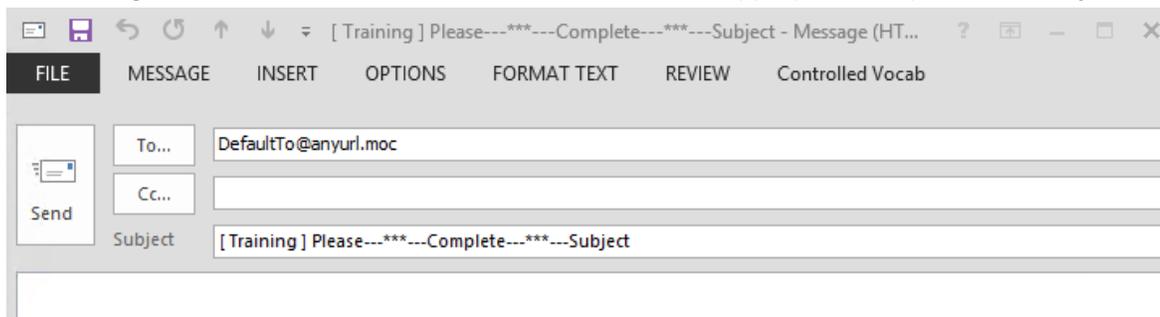
2. Copy QuickStart.zip and QuickStart.xml to the share.
3. Double-click QuickStart.cvcf, and then click Add Selected.



4. Open Outlook to see that the button has been added.



5. Clicking on the selected button creates an email with the appropriate recipients and subject line:



Once you have an idea of how Controlled Vocabulary works, spend some time thinking about how you think your team could collaborate most efficiently and which subject or themes you deal with can be modeled in your button(s). For more information on the capabilities, you can visit the [development site](https://github.com/mikefourie/ControlledVocabulary)<sup>109</sup> and even contribute any enhancements you may have.

<sup>109</sup> <https://github.com/mikefourie/ControlledVocabulary>

# Appendix B: Eating your own dogfood is key

*Software is a great combination between artistry and engineering.*

—Bill Gates

The concepts covered here are not silver bullets but rather a collection of proven technologies and techniques that we have used on our projects. We have cherry-picked and evolved a tailored framework based on our context but kept deep roots and a foundation based on Lean thinking and Agile values.

Sharing our framework is not an attempt to trigger another process debate, such as Kanban (Toyota, 2014) versus Agile (Agile Manifesto, 2001a) versus Lean (Lean Enterprise Institute, 2009) versus Scaled Agile Framework(SAFe) (Leffingwell, 2014a). Instead, we want to share our learnings, allowing everyone to continuously inspect, discover possible improvements, and continuously adapt the way they work.

Dogfooding gives us access to the latest and greatest technologies, enables collaboration with the product group, and provides early feedback to the product team. Within this chapter, we will provide an overview of the various frameworks and methodologies that we have leveraged in churning out projects over the past several years.

## Agility is key

The Microsoft Solutions Framework (MSF), the Agile Manifesto, Lean, Scrum, Kanban, and SAFe have all influenced our approach to shipping open source solutions and guidance. If you are familiar with them, you can go straight to “Using technology wisely.”

## ALM Rangers manifesto

We provide professional guidance, practical experience, and gap-filling solutions to the ALM community. We expect every team member to virtually sign and commit to our manifesto:

### **Razor sharp focus on quality and detail on the work we do**

- Favor simplicity and low tech over complexity

- Expect and adapt to change, delivering incremental value

### **Accountability and commitment**

- Actively manage the project triangle attributes: features, bandwidth, and cost

- Never go dark . . . always share the good, the bad, and the ugly with the team

### **Nonstop and unrestricted collaboration**

- Empower the ALM community

- Embrace open communications

### **Global transparency and visibility through collaboration and shared infrastructure**

- For all initiatives, track and publish status on a timely basis

- Access to everything for everyone

### **Empathy, trust, humility, honesty, and openness at all times**

- No one knows everything; as a team, we know more

- Learn from and share all experiences

## Regular dogfooding of Visual Studio Application Lifecycle Management tools

- Improve productivity of teams
- Gather and share real-life experiences

## Microsoft Solutions Framework

Since the early days of the Microsoft Solutions Framework (MSF), we have emphasized that agility is extremely important and that we must embrace change. MSF has principles and practices where the former was about the mindset of the team and the latter about the actual practice of the individual. In our mind, these have always closely aligned with the Agile Manifesto. MSF forms the base of our practices.

Microsoft Solutions Framework is a set of software engineering processes, principles, and proven practices intended to enable developers to achieve success in the software development lifecycle (SDLC). The following are the eight foundational principles and nine mindsets that form the backbone for the other models and disciplines of MSF:

Principles	Mindsets
<ul style="list-style-type: none"> <li>Foster open communication</li> <li>Work toward a shared vision</li> <li>Empower team members</li> <li>Establish clear accountability and shared responsibility</li> <li>Focus on delivering business value</li> <li>Stay agile, expect change</li> <li>Invest in quality</li> <li>Learn from all experiences</li> </ul>	<ul style="list-style-type: none"> <li>Familiarize team members to how they should approach solution delivery</li> <li>Foster a team of peers</li> <li>Focus on business value</li> <li>Keep a solution perspective</li> <li>Take pride in workmanship</li> <li>Learn continuously</li> <li>Internalize qualities of service</li> <li>Practice good citizenship</li> <li>Deliver on our commitments</li> </ul>

Comparing and contrasting the MSF principles and mindsets with the Agile Manifesto shows like-minded thoughts.

## Agile Manifesto

The [Agile Manifesto](http://www.agilemanifesto.org)<sup>110</sup> for Agile Software Development shares some of the same considerations with MSF. The Agile Manifesto is “uncovering better ways of developing software by doing it and helping others do it.” (Agile Manifesto, 2001a). We are encouraged to value:

Individuals and interactions	over processes and tools
Working software	over comprehensive documentation
Customer collaboration	over contract negotiation
Responding to change	over following a plan

In addition, the principles behind the manifesto are:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

<sup>110</sup> <http://www.agilemanifesto.org>

- 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 and then tunes and adjusts its behavior accordingly.

## Lean

Lean software development is a translation of Lean manufacturing and Lean IT principles and practices to the software development realm as adapted from the Toyota Production System.<sup>111</sup> The following are the Lean principles:

- Eliminate waste
- Build quality in
- Create knowledge
- Defer commitment
- Deliver fast
- Respect people
- Optimize the whole

“Eliminate waste” is at the top of the list because it has the most potential for changing how software is developed. Addressing and eliminating waste provides the greatest opportunity to reduce costs and increase the effectiveness of the solution under development (Table 12).

Manufacturing	Software development
In-process inventory	Partially done work
Overproduction	Extra features
Extra processing	Relearning
Transportation	Handoffs
Motion	Task switching
Waiting	Delays
Defects	Defects

Table 11. Seven Wastes.

## Scrum

Scrum has been around for some time now, and experience teaches us that while it is easy to understand, it is challenging to implement. We do not declare that we are strict Scrum teams. As we have written before, we are in many ways anti-Scrum, as we are not colocated and we are not full-time. However, there are process gems that are

<sup>111</sup> Mary and Tom Poppendieck, *Implementing Lean Software Development* (Addison-Wesley, 2007)

of value to our teams, and we use what facilitates and enables our teams to ensure frequent and consistent delivery of quality solutions (Table 13).

Scrum recommendations	Our adaptation
Product Owner owns product backlog	Product Owner owns product backlog
Team completes work in sprints	Team completes work in sprints
Daily "scrum" for 15 minutes to assess progress	Weekly "scrum" for 15+15 minutes to assess progress
Scrum Master keeps team focused and removes blockers	Scrum Master keeps team focused
Each sprint delivers potentially shippable product	Each sprint delivers potentially shippable product
Each sprint ends with a sprint review and retrospective	Each sprint ends with a sprint review and retrospective
Team is typically full-time and colocated	Team is typically part-time and not colocated

Table 12. Comparing pure Scrum recommendations to our reality.

## Kanban

Kanban is a method for process improvement. Initially, Kanban for software development was positioned as a process and later positioned as a tool for process improvement that works on top of our existing process. In software development, teams apply Kanban to their software development process to help the team improve. Kanban pulls from the principles of Lean manufacturing, which is based on systems thinking, to take an overall view of our software development process from end to end, identifying wasteful activities. Eliminating waste is a key Lean principle. Kanban provides tools to aide in the identification of waste for elimination.

### Principles of Kanban

In software development, Kanban focuses on the time it takes for work to go through one's process from conception to final delivery, also known as "concept to cash." We call this *lead time*. The delivery to the customer represents realized value. So, through reduced lead time, the customer swiftly receives the value.

### Visualize the workflow

Visualization of workflow is key in understanding the flow of work through our software development process and identifying the phases in our process. Visual Studio Online and Team Foundation Server provide such visualization with a Cumulative Flow Diagram and a Kanban board. These visualizations are useful in viewing the work and its phases and tell a story at a glance about the team and its work.

### Limit work in progress

Kanban is based on setting work-in-progress (WIP) limits at each stage of the workflow and relies on a pull versus push mechanism for new work. This emphasizes the completion of work over the acceptance of new work. The net effect is people are able to concentrate on fewer things at the same time; remember that one of the wastes is task switching. WIP limits reduce a buildup of unfinished work, another waste previously mentioned. By preventing people from starting (pulling) new work, WIP limits expose constraints or bottlenecks in an organization's process. Focusing improvement methods on these constraints ensures the highest payback.

## SAFe

[Scaled Agile Framework \(SAFe\)](#)<sup>112</sup> does a very good job of describing approaches to scaling Agile and Scrum from top to bottom in larger organizations where importance is placed on budget and ship dates that may have known states. SAFe does this via a well-established and well-outlined framework, principles, and best engineering practices that are based on the fundamentals of Lean, Kanban, Agile, Scrum, and Extreme Programming.

We are adopting some of the best practices from SAFe. For example, we use the SAFe naming convention for sprints and have standardized our estimation so that we normalize the values across all projects. In other words, a value of 3 for estimation is uniform across all projects and teams.

Microsoft published the [Scaled Agile Framework: Using TFS to support epics, release trains, and multiple backlogs](#)<sup>113</sup> paper on the implementation of SAFe in the Team Foundation Server process templates. In addition, we provide collateral tools to implement SAFe support in our process template.

## Using technology wisely

We base our work on the Microsoft Visual Studio solution stack: Team Foundation Server for experimentation and Visual Studio Online for our operational backbone. Yet, this does not preclude us from leveraging other third-party solutions to deliver on our mission. Identifying the right tools is as important as using the right process.

## Team Foundation Server

Before we used Visual Studio Online, we leveraged the internal Microsoft IT managed Team Foundation Server. This worked well for us, and our challenges were exposing the internal services to team members who are not full-time Microsoft employees and managing process template customizations. We could customize work-item templates for existing projects, but we could not upload custom process templates with our changes for use in new team projects. Customizing work-item templates for each project was more effort with little return on our investment. The upside is we worked with the existing process templates without customization, enabling us to provide valuable feedback to the product group that would often result in product changes for the next release. In addition, we avoided process template changes that may have enabled process changes but may not have provided business value. This limitation continued for us as we moved to Visual Studio Online.

## Visual Studio Online

We were one of the first organizations to use Visual Studio Online (VSO); initially, Microsoft named it TFSPreview. VSO maintained the same constraint for process template customization and added a new constraint for work-item template customization. However, these constraints resulted in us providing valuable feedback to the product team. The new delivery cadence, feature flags, and cloud deployment meant our suggestions made their way into the product quickly. We would test these changes before we enabled the feature for all users. In VSO, we manage all of our work in one team project collection and primarily in one team project using teams.

In fact, in April 2011, we moved all teams from on-premises environments to the VSO service, with no easy or painless return strategy . . . we were that committed to making it work!

## CodePlex

[CodePlex](#)<sup>114</sup> is a Microsoft website for hosting shared development of open source software. CodePlex is where engineers and computer scientists share projects, ideas, and the source code. It includes features such as wiki pages,

---

<sup>112</sup> <http://www.scaledagileframework.com/>

<sup>113</sup> <http://msdn.microsoft.com/en-us/library/dn798712.aspx>

<sup>114</sup> [www.codeplex.com](http://www.codeplex.com)

source control, discussion forums, issue tracking, project tagging, RSS support, statistics such as number of project downloads, and releases, including ClickOnce. We release the output of the value we create primarily to CodePlex, although we have also released to the Visual Studio Gallery.

## Collaboration

The geographically distributed nature of our teams requires us to leverage tools that enable collaboration between teams and individuals.

We rely on tools such as Skype and Lync to facilitate planning, estimation, communications, and collaboration.

**GEM****Seeing is believing!**

To ensure that geographically distributed, virtual, and part-time team members, many of whom will never meet face to face in their lifetime, get to know each other by sight and voice, we encourage the use of webcams.

## TFS Agile Poker

[TFS Agile Poker](#)<sup>115</sup> is a free [Windows 8 tool](#)<sup>116</sup> for estimating by using Team Foundation Server. One of the great things about this tool is that it can work with Team Foundation Server and Visual Studio Online.

## MyHistory

[MyHistory](#)<sup>117</sup> is a great extension written by Mike Fourie for his MSDN article [Extending Visual Studio Team Explorer 2012](#).<sup>118</sup> This is a very useful example of how to extend Team Explorer. With this extension, we have one place to see all of our history, such as work items, shelve sets, change sets, and recent solutions. Many of us use this tool to visualize all of their history in one place.

## Team rooms

[TFS Team Rooms](#)<sup>119</sup> provides access to team rooms inside Visual Studio. The user remains focused within the integrated development environment (IDE) while collaborating within one or more team rooms.

## News

[News](#)<sup>120</sup> delivers Visual Studio Online and other news and updates inside Visual Studio, again keeping the users focused within their IDE.

## Yammer

Yammer is a private social network that brings together all of our teams for conversations, surveys, content, and data in one place. Through Yammer, we stay connected to our community and information and collaborate with team members and other engineers. We access Yammer using a web browser, a mobile device, or Outlook.

---

<sup>115</sup> <http://tfsagilepoker.com>

<sup>116</sup> <http://apps.microsoft.com/windows/app/team-planning-poker/5afb3e32-750b-47c5-8e36-36d9ca7965a3>

<sup>117</sup> <https://myhistory.codeplex.com>

<sup>118</sup> <http://msdn.microsoft.com/en-us/magazine/dn201751.aspx>

<sup>119</sup> <http://visualstudiogallery.msdn.microsoft.com/c1bf5e4f-5436-465d-87da-09b2f15ff061>

<sup>120</sup> <http://visualstudiogallery.msdn.microsoft.com/6c4d14c3-e66f-40d0-8d77-ff7883a40f6a>

## Flight Plan Status Board

An innovative visualization project developed by the ALM community. We use the [Flight Plan Status Board](#)<sup>121</sup> to visualize the progress and status of our projects.

## Controlled Vocabulary

[Controlled Vocabulary](#)<sup>122</sup> is a tool we use to organize our emails for subsequent retrieval. This project provides an Outlook client to support the adoption of a controlled vocabulary.

We use it for subject indexing by adding predefined keywords to the email subject headings. This enables easy searching and quick recognition of email associated with projects.

## Adapting to reality

### One maintainable environment . . . simplicity rules

#### Visual Studio Online

While we evaluated and tested the use of the TFS Integration Tools to migrate all our projects from on-premises servers to the cloud (see [Migrating from an On-Premises Team Foundation Server to Team Foundation Service Preview Using the TFS Integration Tools](#)<sup>123</sup> for details), we made two strategic choices. We decided to start at ground zero in terms of work items and to manually check in the “latest” code base as needed. We have no auditing or operational requirements that require us to maintain extensive history. For the very infrequent visits to the past, our backups have proven more than adequate.

Although the planning guide and the planning poster would typically nudge us toward the cloud-based Team Foundation Service home, we had a different motivation—to be among the first users of [Visual Studio Online](#),<sup>124</sup> committing ourselves to the solution. The quick reference planning guide shown in Figure 97 was used to plan our migration and determine our TFS structure. The ✓ and ⊙ symbols show our decision path.

---

<sup>121</sup> <https://vsarflightplan.codeplex.com/>

<sup>122</sup> <http://mikfourie.github.io/ControlledVocabulary/>

<sup>123</sup> <http://msdn.microsoft.com/en-us/magazine/jj130558.aspx>

<sup>124</sup> <http://www.visualstudio.com/products/visual-studio-online-overview-vs>

Do I need 1 [, 2,] or more?

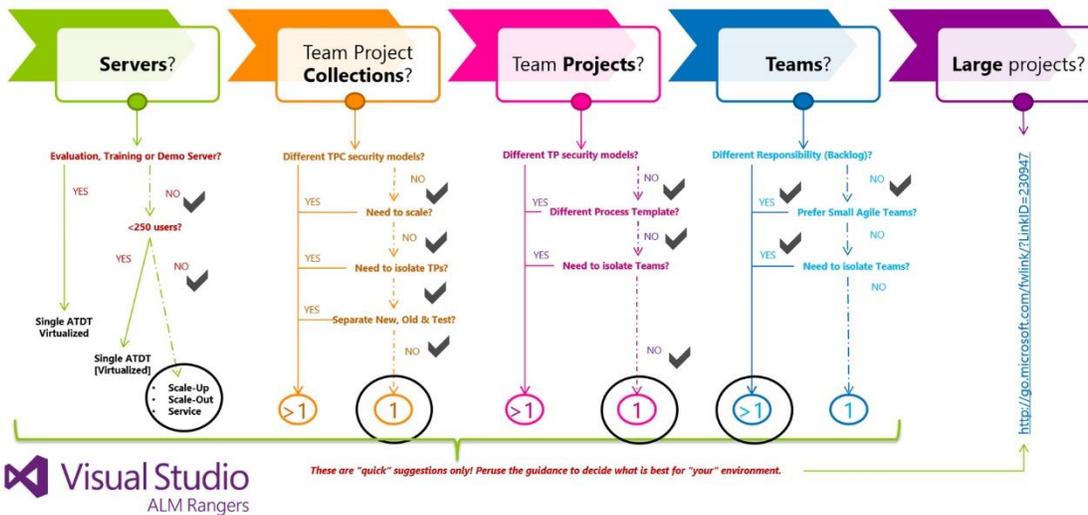


Figure 97. Planning guide: Guided toward Visual Studio Online.

## One (or few) team project collection(s)

We keep the number of team project collections as small as possible to minimize administration cost and, most importantly, to keep it simple for the user. Switching between team projects collections closes all queries, work items, documents, and solutions. This frequent switching can be a daunting, confusing, and unproductive experience.

Our decision to embrace Visual Studio Online enforced the one team project collection model because each service account is currently limited to one team project collection. This implies that we are enjoying the advantages of administrative and usage simplicity, as well as the ability to share artifacts among team projects.

In the future, we might analyze the "potential" need for more team project collections if the service introduces support for more than one collection per account. What we would like to change, when possible, is the team project collection name to make it more descriptive.

**NOTE** We live and thrive within one team project collection.

## One (or few) team project(s)

Planning of team projects is crucial, because once we have implemented a design strategy, it becomes immutable and very, very difficult and costly to change. Our objectives were to improve the user experience, centralize our backlog and bug management, standardize on one process template, and, most importantly, to dogfood and align our environment with our latest planning guidance.

Looking back over the past year, the move to a single team project has been interesting, sometimes challenging, but most definitely a good one. For those who have embraced the environment and the supporting process, the productivity gain through less context switching (remember, context switching is a waste) and better sharing, as well as the improved transparency across projects, is both evident and appreciated.

In reality, our team project collection currently contains nine team projects. We started with a one team project per project model, are now in one team project per product (Visual Studio ALM), and anticipate that the collection will have no more than three team projects in the future.

Are we really using only one team project? The answer is both yes and no (see Figure 98).

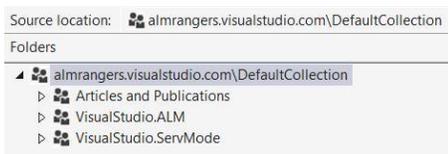


Figure 98. We actually have more than one team project (for a reason).

As shown, we have three team projects. VisualStudio.ALM is our production team project, housing 99 percent of our artifacts and 100 percent of all active projects. The Articles and Publication and the VisualStudio.ServMode team projects are archive team projects, which contain a backup of our publications and our inactive projects.

**NOTE** We have one operational team project and a “few” archive team projects.

## Multiple teams lighting up

The introduction of Team Foundation Server teams allows us to design an environment that is more representative of and conducive to a team environment. Working through the Team Foundation Server Planning Guide, we quickly realized that we needed more than one team, each owning an area path and iteration path. As part of our dogfooding, we are encouraging teams to remain relatively small, typically five members, but up to a maximum of ten, to promote agility and reduce administrative overhead on the Project Lead and Scrum Masters. Teams that outgrow these limits usually become tough to monitor within our challenging environment (ALM Guidance: [Visual Studio ALM Rangers—Reflections on Virtual Teams](#))<sup>125</sup> and are best broken into smaller feature teams.

We have sliced up our team project into projects, typically relating 1:1 to those seen on the [ALM Ranger solutions catalog](#).<sup>126</sup> One team represents each project. In some cases, for example with our quick-response solutions, we have an umbrella project that contains multiple related but separate feature projects. We use the area path as the dividing line between feature teams, but treat them as one project, with one joint goal, tracking, and backlog.

**NOTE** We have many teams and in some cases feature teams.

## One (shared) sprint

In [ALM Rangers dogfooding journal of the Team Foundation Service](#), we introduced our monthly cadence in July 2012. Used by all our project teams, it created a predictable rhythm for the community and our engineering process. It allowed our teams to discuss past, current, and future sprints and milestones with a common language.

### **Cadence + Synchronization = Predictability!**

We broke up the year into 12 monthly sprints, each of which we optionally split into two one-half month sprints. We started with the new definition of the iteration path on July 1, 2012, which is the reason why the monthly sprints are numbered 1–12 for monthly (and 1.1, 1.2, 2.1, 2.2, and so on) during FY13 and 13–24 in FY14. (See Figure 99.) The teams use monthly, half-monthly, or a combination thereof. This allows each team to choose their delivery cadence. The typical pattern is a half-month planning sprint, followed by three to four monthly sprints. Each sprint starts with achievable sprint objectives and ends with a sprint deliverable. Teams with no deliverable or heartbeat instill no confidence and are red flagged.

<sup>125</sup> <http://msdn.microsoft.com/en-ca/magazine/hh394152.aspx>

<sup>126</sup> <http://aka.ms/vsarsolutions>

FY13											
JUL 1	AUG 2	SEP 3	OCT 4	NOV 5	DEC 6	JAN 7	FEB 8	MAR 9	APR 10	MAY 11	JUN 12
FY14											
JUL 13	AUG 14	SEP 15	OCT 16	NOV 17	DEC 18	JAN 19	FEB 20	MAR 21	APR 22	MAY 23	JUN 24
FY15											
JUL 25	AUG 26	SEP 27	OCT 28	NOV 29	DEC 30	JAN 31	FEB 32	MAR 33	APR 34	MAY 35	JUN 36

Figure 99. Shared (common) sprints starting S1 on July 1, 2012.

We managed to create a controlled environment, cadence and synchronization and reduce unpredictable events. Our less-than-perfect software engineering planet, populated by geographically distributed teams, made up of part-time volunteers, was predictable. However, we did not align our iterations with the broader engineering cadence of three weeks, as shown in the hypothetical example in Figure 100.

**!(Cadence + Synchronization) = Unpredictable => Misaligned Teams**

FY15															
JUL 25	AUG 26	SEP 27	OCT 28	NOV 29	DEC 30	JAN 31	FEB 32	MAR 33	APR 34	MAY 35	JUN 36				
	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85

Figure 100. Misaligned cadence.

Regular, stakeholder-wide integration and synchronization were challenging and often confusing. Stakeholders using the three-week cadence and associated milestones were understandably confused with the language “Project X will ship during sprint 30, which is either the end of sprint 76 or start of sprint 77.” Huh?

Therefore, on August 25, after a very brief deliberation among Scrum Masters, we changed our future iterations from monthly to three weeks. Now we have aligned our iterations with all the other software engineering processes in terms of the rhythm, synchronization, and resultant predictability (Figure 101).

**Cadence + Synchronization = Predictability!**

FY15															
JUL 25	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN				
	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
Other teams	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85

Figure 101. Aligned cadence.

Regular stakeholder-wide integration and synchronization should now see clouds of confusion replaced with smiles of predictability. We cannot wait to report that “Project X will ship during sprint 77” at our next synchronization event. Simple! Cadence + Synchronization = Predictability! => Aligned Teams!

## GEM

**Meaningful names!**

Just as a common vocabulary adds clarity, aligning iteration names with the rest of the organization makes it easier. The screen shot in Figure 102, for example, uses Financial Years as the main node, with children nodes named according to the divisional sprint names S1-Sn. We added a bit of additional context, initially the start date and more recently the week numbers.

Iterations	Start Date	End Date	
VisualStudio.ALM	2012-07-01	2014-06-30	Backlog iteration for this team
Projects	2012-07-01	2014-06-30	
FY13	2012-07-01	2013-06-30	
FY14	2013-07-01	2014-06-30	
Copy Editing	2014-06-01	2015-06-30	
FY15	2015-07-01	2016-06-30	
Sprint 25 (201407)	2014-07-01	2014-07-31	
Sprint 26 (201408)	2014-08-01	2014-08-31	
Sprint 71 (35-37)	2014-08-25	2014-09-14	
Sprint 72 (38-40)	2014-09-15	2014-10-05	
Sprint 73 (41-43)	2014-10-06	2014-10-26	
Sprint 74 (44-46)	2014-10-27	2014-11-16	
Sprint 75 (47-49)	2014-11-17	2014-12-07	
Sprint 76 (50-52)	2014-12-08	2014-12-28	
Sprint 77 (1-3)	2014-12-29	2015-01-18	
Sprint 78 (4-6)	2015-01-19	2015-02-08	
Sprint 79 (7-9)	2015-02-09	2015-03-01	
Sprint 80 (10-12)	2015-03-02	2015-03-22	
Sprint 81 (13-15)	2015-03-23	2015-04-12	
Sprint 82 (16-18)	2015-04-13	2015-05-03	

Figure 102. Visual Studio Online iteration/sprint view.

Area path == team

We primarily use the area path for the team per the default setting of creating a new team, although teams are free to create subareas for additional use.

Personas are the glue

The [Supporting Guidance and Whitepapers](#)<sup>127</sup> defines all the personas and their roles that we encounter or that our solutions target. But before we talk about our team personas, be sure to look back at Table 11, which shows a matrix that compares the roles and the suitability of overlapping them.

Product Owner

The Product Owner (PO) is the face of the product to the leadership chain and the development team, makes the tough decisions on scope and engages other stakeholders as needed.

As shown in the Figure 103, the PO is the interface to the product group team or teams. The PO facilitates all communication with the product group, thus shielding the product group and establishing 1:1 communication links when and as appropriate.

The PO prioritizes and owns the backlog, especially the Feature/Epic PBIs. The PO coordinates communication between product and development teams, driving the product knowledge and alignment with product groups. The PO creates product awareness and passionately promotes the solution and acts as the subject matter expert with his liaison with the product teams.

<sup>127</sup> <http://vsarguidance.codeplex.com/>

## Project Lead

The Project Lead (PL), also known as Dev Lead, enables development teams. The PL loves transparency, dislikes information hiding, and focuses on fine-tuning the team and collaboration with other stakeholders.

As shown in Figure 103, the PL is the interface to the development team(s). The PL facilitates all communication with the development team, thus shielding the team and establishing 1:1 communication links when and as appropriate.

**NOTE** A PL can optionally act as a Scrum Master, contributor, and reviewer as well, in which case this team member's responsibilities and estimated effort will change.

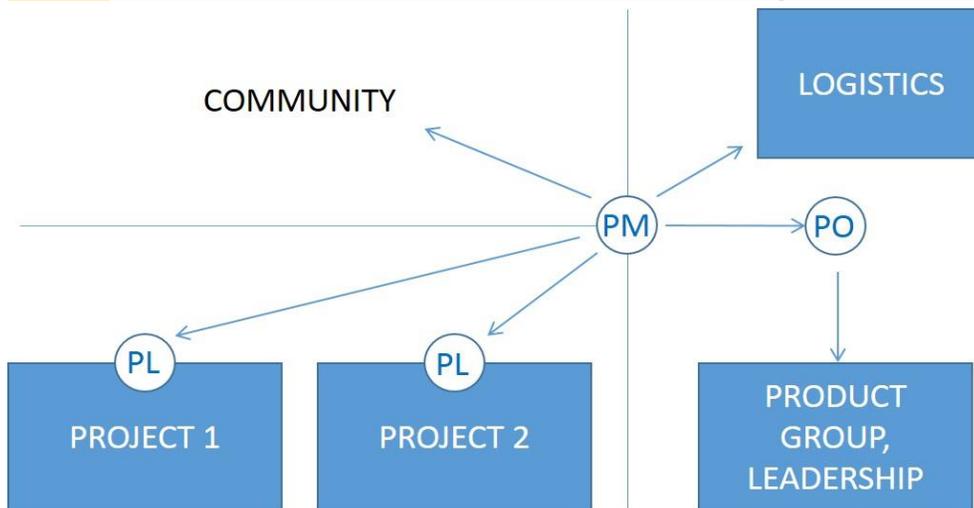


Figure 103. Roles and responsibilities for Product Owner, Project Lead, and Program Manager.

## Program Manager

The Program Manager (PM) works with stakeholders, customers, and development teams, focusing on team orchestration and keeping the team and backlog from stagnating. The blog posts [Persona within the ALM Rangers](http://blogs.msdn.com/b/willy-peter_schaub/archive/2014/08/22/fun-of-pm-1-persona-within-the-alm-rangers.aspx),<sup>128</sup> [Energizing the team](http://blogs.msdn.com/b/willy-peter_schaub/archive/2014/08/27/fun-of-pm-2-energi-s-z-ing-the-team.aspx),<sup>129</sup> [Program Management—Thinking about PM != PM and Visual Studio ALM Rangers \(Part 2\)](http://blogs.msdn.com/b/willy-peter_schaub/archive/2011/10/25/program-management-thinking-about-pm-pm-and-visual-studio-alm-rangers-part-2.aspx),<sup>130</sup> and [Program Management – Are some of the ALM Rangers Symbiotic PM's?](http://blogs.msdn.com/b/willy-peter_schaub/archive/2011/11/15/program-management-are-some-of-the-alm-rangers-symbiotic-pm-s.aspx)<sup>131</sup> explore the definition of a PM within Microsoft and the slight variation within the ALM Rangers. The PM lives between the PO and PL, with a strong connection to the community and end users. In addition, the PM assists with the backlog, especially with aligning customer expectations and scope, and coordinates communication between stakeholders. As with other roles, we expect Program Managers to create product awareness and passionately promote the solution.

## Contributors

The contributor, also known as developer, builds the product. Contributors maintain their assigned tasks and keep them up to date on the backlog. Our process is to select and work on one task at a time to keep the backlog free for others to assign themselves work. When a contributor assigns more than one task to himself or herself, we assume they need to work on both tasks because there is some dependency. Otherwise, we are creating waste in the form of undone work.

<sup>128</sup> [http://blogs.msdn.com/b/willy-peter\\_schaub/archive/2014/08/22/fun-of-pm-1-persona-within-the-alm-rangers.aspx](http://blogs.msdn.com/b/willy-peter_schaub/archive/2014/08/22/fun-of-pm-1-persona-within-the-alm-rangers.aspx)

<sup>129</sup> [http://blogs.msdn.com/b/willy-peter\\_schaub/archive/2014/08/27/fun-of-pm-2-energi-s-z-ing-the-team.aspx](http://blogs.msdn.com/b/willy-peter_schaub/archive/2014/08/27/fun-of-pm-2-energi-s-z-ing-the-team.aspx)

<sup>130</sup> [http://blogs.msdn.com/b/willy-peter\\_schaub/archive/2011/10/25/program-management-thinking-about-pm-pm-and-visual-studio-alm-rangers-part-2.aspx](http://blogs.msdn.com/b/willy-peter_schaub/archive/2011/10/25/program-management-thinking-about-pm-pm-and-visual-studio-alm-rangers-part-2.aspx)

<sup>131</sup> [http://blogs.msdn.com/b/willy-peter\\_schaub/archive/2011/11/15/program-management-are-some-of-the-alm-rangers-symbiotic-pm-s.aspx](http://blogs.msdn.com/b/willy-peter_schaub/archive/2011/11/15/program-management-are-some-of-the-alm-rangers-symbiotic-pm-s.aspx)

## Reviewers

Reviewers, also known as testers, validate the functionality and quality of the product. As with other roles, reviewers keep their tasks up to date, identify impediments early, and push the team to achieve the highest quality bar.

## Mentors

The mentor, also called peer buddy, is an experienced subject matter expert, acting as a mentor to his or her peers, in particular to the Project Lead and backup Project Leads but also to engineers in training.

**NOTE** The article [The Difference Between a Mentor and a Consultant](#)<sup>132</sup> gives an interesting perspective of a mentor. The article states: "A mentor should be motivated by nothing more than the desire to help. Second, a mentor's role is not to advise you but rather to give you a different way of thinking."

## Scrum Master

The Scrum Master (SM) guides, mentors, and coaches the Project Leads and teams as needed by using Scrum, Extreme Programming, SAFe, and Kanban as enablers. The Scrum Master is responsible for removing impediments and facilitating communications, and, as a team player, facilitates, mentors, and guides the team. We strive to share information transparently and ensure team status is visible and known at all times to everyone.

The Scrum Master supports the Product Owner and Project Leads in an effort to facilitate their involvement and minimize their overhead. The Scrum Master continuously improves the teams and the engineering practices and tools.

This appendix brought you an overview of the popular Agile frameworks and our adaptations for part-time and geographically distributed teams. We shared with you how we adapted, our use of the Microsoft Visual Studio ALM stack and other Microsoft solutions such as Yammer and Lync, and our use of third-party tools.

Before we end this appendix, we give you a final case study for one of our projects.

---

<sup>132</sup> [http://www.inc.com/magazine/201207/norm-brodsky/difference-between-mentor-and-consultant.html?utm\\_source=feedburner&utm\\_medium=feed&utm\\_campaign=Feed%3A+inc%2FHeadlines+%28Inc.com+Headlines%29](http://www.inc.com/magazine/201207/norm-brodsky/difference-between-mentor-and-consultant.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+inc%2FHeadlines+%28Inc.com+Headlines%29)

## Case study: ALM Readiness Treasure Map . . . walking the plank

Managing a virtual team distributed around the globe and establishing a team identity is a complex challenge. We introduce the ALM Readiness Treasure Map team because they have managed to create a cohesive and effective team scattered across Brazil, Canada, India, South Africa, the United Kingdom, and the United States (Figure 104 and Figure 105).



Figure 104. Captain [Robert Bernstein](#)<sup>133</sup> introduced an exciting team spirit and identity.

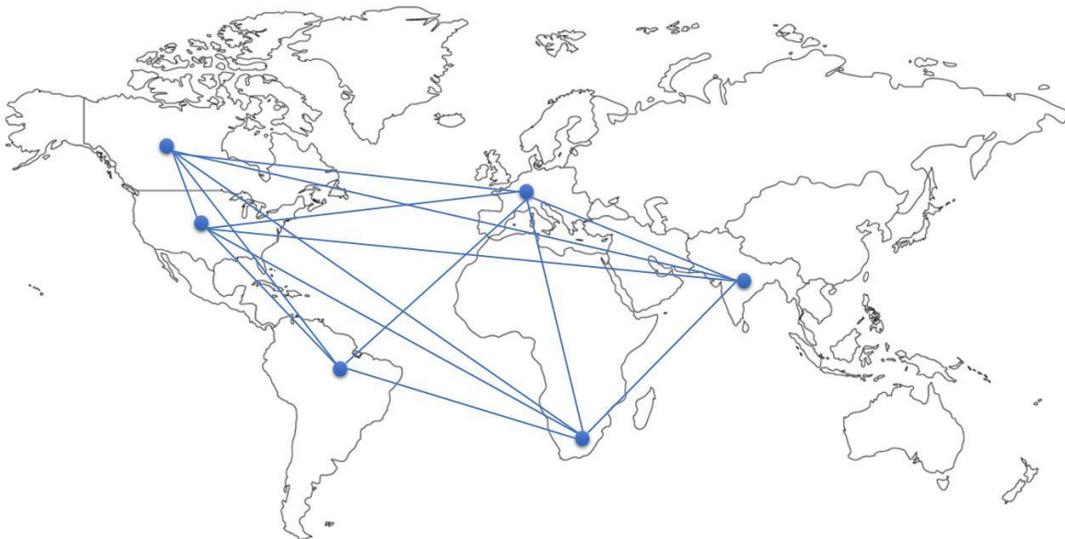


Figure 105. Distributed ALM Readiness Treasure Map team.

### Background information

The team's original motivation was to provide a master catalog (treasure map) of the available ALM Readiness content to guide and track anyone through the process of becoming proficient in ALM practices. The design and development of the treasure map intended to dogfood new concepts and technologies while delivering a quality reference solution. Find more information at [TOC—ALM Readiness Treasure Map](#).<sup>134</sup>

<sup>133</sup> [http://blogs.msdn.com/b/willy-peter\\_schaub/archive/2012/03/04/introducing-the-visual-studio-alm-rangers-robert-bernstein.aspx](http://blogs.msdn.com/b/willy-peter_schaub/archive/2012/03/04/introducing-the-visual-studio-alm-rangers-robert-bernstein.aspx)

<sup>134</sup> <http://aka.ms/vsartmaptoc>



# Afterword: We are definitely not dysfunctional!

Context: "Our final sprint is blocked and potentially delayed again."

We need to fine-tune our value stream from start to finish, automating everything and raising events as early as possible. Does this ring a bell? Just like a factory line or a release pipeline, we strive to automate the entire creation process. Delivering business value is not just about raw speed, it is about agility, detecting and resolving impediments early, and a willingness to take a risk. "Do and ask for forgiveness later" is advice we hear and give often.

Focusing our final sprint on quality and planning, we ensure that Product Owners (PO), Program Managers (PMs), and selected Project Leads (PLs) sit together, discussing what needs attention and looking for wait times. Figure 107 highlights the often problematic tasks in our assembly line.

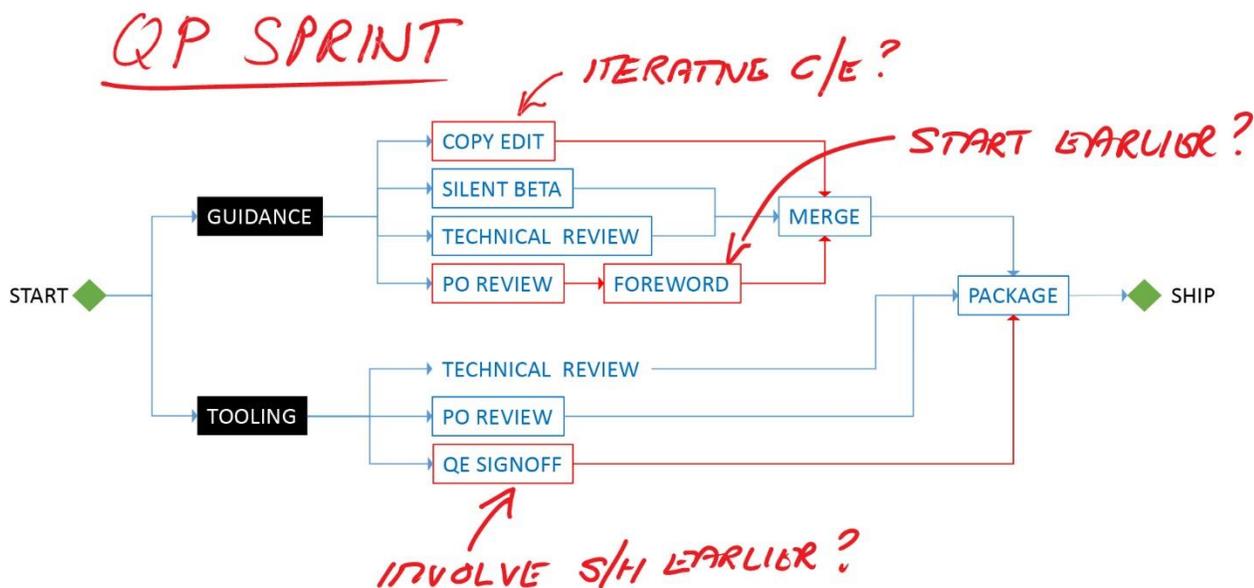


Figure 107. The last sprint, focused on quality and planning, shows bottlenecks.

Snippets of the dialogue from one of these meetings:

- "Do you agree this resembles our [Config as Code for DevOps and ALM practitioners](https://vsardevops.codeplex.com/)<sup>137</sup> project? Willy asks.
- Keith ponders the diagram. "Yes, and we should start the copyediting and quality essentials earlier, to ensure we do not create bottlenecks in the last sprint."
- "Anything else we could improve to make the Product Owner more effective?"
- "The foreword was one thing that slipped my mind and created a substantial level of 'Angst.' I recommend that Product Owners be reminded of these core deliverables during each sprint to keep them on their radar."
- "What went well for you, Keith?"

<sup>137</sup> <https://vsardevops.codeplex.com/>

- "The team, status, and backlog transparency and unrestricted access to the potentially shippable increments allowed me to proactively review and give candid feedback."
- "Great, candid and actionable feedback," Willy says with excitement.

What this brief epilogue and case study demonstrates is that even our framework, developed and refined during years of dogfooding, is not perfect. Remember: reflection, continuous improvement, and a willingness to take risks and embrace change are instrumental to our success.

## Influenced by giants

Our framework is based on a number of trusted, established, and proven technologies and frameworks, such as Lean thinking, Agile, MSF, Scrum and the Scaled Agile Framework (SAFe) (Figure 108). In addition, we stand on the shoulders of giants from the ALM community and a mantra of continuous inspection and adaptation.

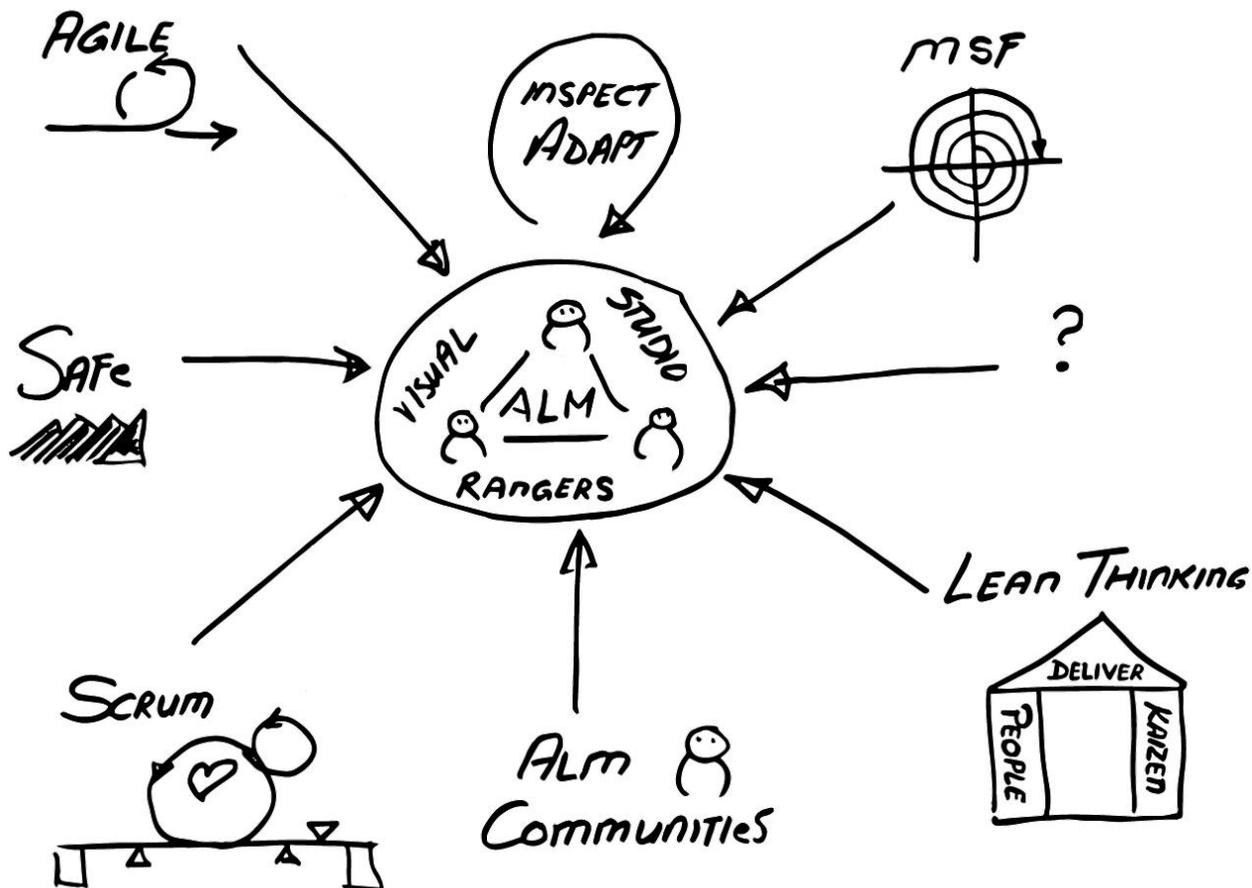


Figure 108. Influenced by giants.

There is no silver bullet, no framework that suits them all. or no one individual who has all the experience and drive to enable communities such as the Visual Studio ALM Rangers to become effective overnight. It is a continuous journey that requires tenacity, passion, experience, and a willingness to uncover impediments, realize that we are not perfect, adapt, and improve.

# Ruck == Loose Scrum == Scrum != Dysfunctional

The final myth we can dispel is that we are rebellious and dysfunctional anti-Scrum activists. We have done what Scrum encourages us to do . . . *inspect, adapt, and improve* (Figure 109).

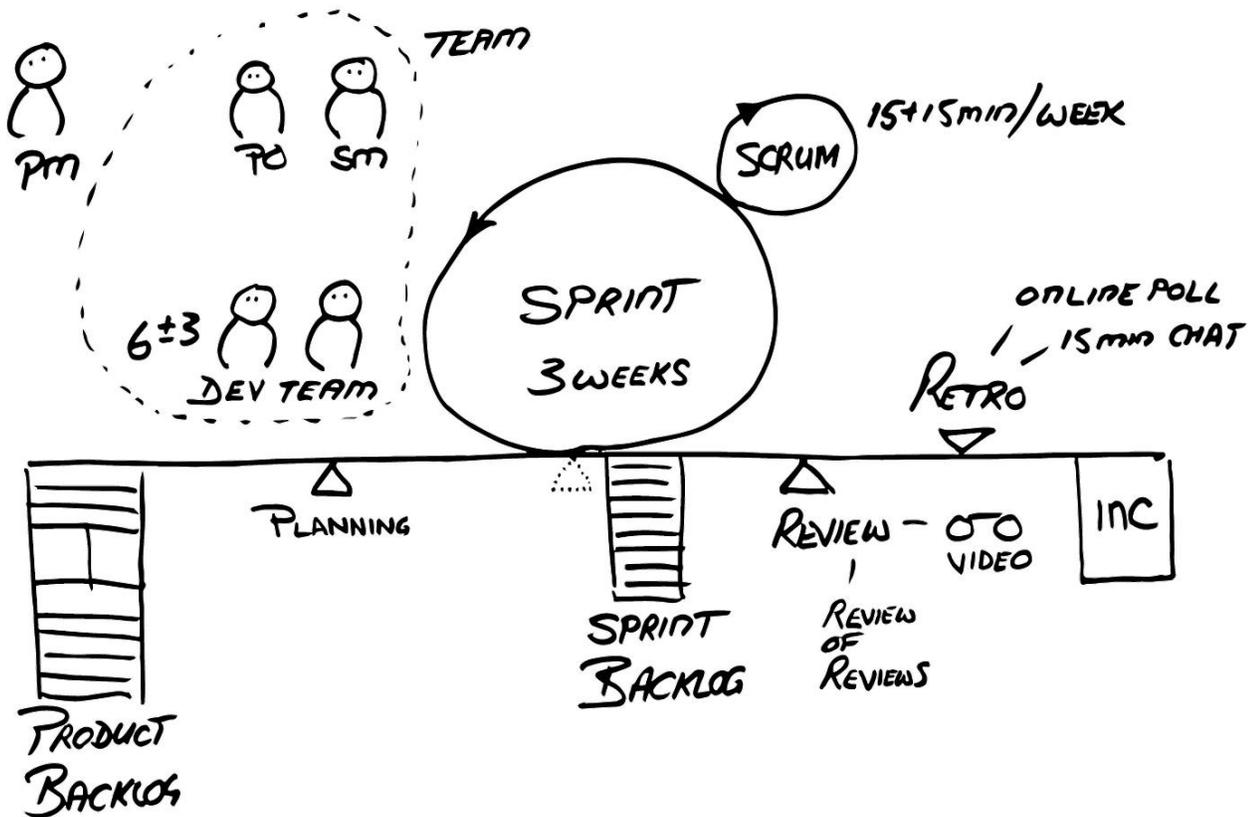


Figure 109. Ruck is "loose Scrum" and hence Scrum.

Looking at this illustration, we recognize the foundations of Scrum and our adaptation to make our geographically dispersed, part-time, and volunteer-driven community more effective. Yes, we introduced the Program Manager (PM) as a pivotal and supporting stakeholder, we have weekly instead of daily scrums, and we have reviews of reviews for better transparency and cross-team collaboration, retrospectives based on online surveys, and a quick one-minute chat. We have adapted our framework and no, we are not pure Scrum. However, we are effective and are continuously inspecting and adapting . . . not something that we would find in a dysfunctional team.

Thank you for your interest in our dogfooding, your candid feedback, and for bearing with us up to this page. We are all incredibly proud of what we, as a community of passionate, motivated, experienced, geographically distributed, and part-time volunteers have achieved. We urge you to embrace reflection, innovation, change, and risk. You, too, will look back and say:

*"Wow, we really did it"*

# What's Next?

It's up to you now . . .

We have entered an era where we continuously have to reflect on ourselves and innovate and evolve as rapidly as user requirements are changing and software solutions are evolving. We need to ensure that we maintain the value to the user, or we face rapid extinction.

Summarizing our learnings and recommendations in this book was the fun, brief, and easy part of a potentially long-term journey. We hope that we have given you actionable gems, initiated thinking around potential innovations in your environment, and that you will continue to support and encourage self-organized Agile teams. We encourage you to give us candid feedback and wish you success in your software adventures.

## Continuous innovation

For us, this is not the end of the road. As products such as [Visual Studio Online](#)<sup>138</sup> evolve, we will innovate, adapt, and improve our own frameworks. If the practical guidance summarized herein proves valuable, we may even meet for a revision of this publication.

## Further information

For further information, please visit Visual Studio [ALM Ranger Solutions Catalogue](#)<sup>139</sup> or email the [authors](#).<sup>140</sup>

---

<sup>138</sup> <http://www.visualstudio.com/products/what-is-visual-studio-online-vs>

<sup>139</sup> <http://aka.ms/vsarsolutions>

<sup>140</sup> [vsarAsm4Dvpt@visualstudio.onmicrosoft.com](mailto:vsarAsm4Dvpt@visualstudio.onmicrosoft.com)

# References

- Agile Manifesto. (2001a). *Manifesto for Agile Software Development*. Retrieved from agilemanifesto: <http://www.agilemanifesto.org/>
- Agile Manifesto. (2001b). *Principles behind the Agile Manifesto*. Retrieved from Agile Manifesto: <http://agilemanifesto.org/iso/en/principles.html>
- Boer, G., and S. Ferrell. (2013). *Agile Portfolio Management: Using TFS to support backlogs across multiple teams*. Retrieved from MSDN.
- Flinchbaugh, Jamie. (2014). Retrieved from A3 Problem Solving: <http://leanpub.com/a3problemsolving>
- Flinchbaugh, Jamie. (2012). *A3 Problem Solving*. Retrieved from A3 Problem Solving: Applying Lean Thinking: <https://leanpub.com/a3problemsolving>
- The Free Dictionary. (2014). *grooming*. Retrieved from The Free Dictionary: <http://www.thefreedictionary.com/grooming>
- Fuller, Thomas. (1650). Retrieved from The Phrase Finder: <http://www.phrases.org.uk/meanings/darkest-hour.html>
- Geoff, B. (2007). *Software Engineers on their way to Pluto*, ISBN: 978-0-620-38514-5. Johannesburg: BBD ([www.bbd.co.za](http://www.bbd.co.za)).
- Hinshelwood, Martin. (2014, 03 20). *What my father taught me about Agility Path (34 years before it was invented!)*. Retrieved from naked ALM: <http://nakedalm.com/what-my-father-taught-me-about-agility-path-34-years-before-it-was-invented/>
- Lange, S. (2012, 10 4). *VS/TFS 2012 Tidbits: TFS Feedback Management Behind the Scenes*. Retrieved from Steve Lange @ Work: <http://blogs.msdn.com/b/slange/archive/2012/10/04/vs-tfs-2012-tidbits-tfs-feedback-management-behind-the-scenes.aspx>
- Lean Enterprise Institute. (2009). *What is Lean?* Retrieved from Lean Enterprise Institute: <http://www.lean.org/WhatsLean/>
- Leffingwell, Dean. (2014a). *Scaled Agile Framework Big Picture*. Retrieved from Scaled Agile Framework: <http://www.scaledagileframework.com/>
- Leffingwell, Dean. (2014b). *wsjf*. Retrieved from Scaled Agile Framework: <http://scaledagileframework.com/wsjf/>
- Pereira, J. & W. P. Schaub. (2006). *.NET ENTERprise Solutions ... Interoperability for the connoisseur* (1 ed.). Johannesburg, Gauteng, South-Africa: BBD.
- Toyota. (2014). *Just-in-Time—Philosophy of complete elimination of waste*. Retrieved from Toyota: [http://www.toyota-global.com/company/vision\\_philosophy/toyota\\_production\\_system/just-in-time.html](http://www.toyota-global.com/company/vision_philosophy/toyota_production_system/just-in-time.html)
- Visual Studio ALM Rangers. (2014). *Better Unit Testing using Microsoft Fakes*. Retrieved from Visual Studio Test Tooling Guides: <http://aka.ms/treasure27>
- Wikipedia. (2014, 4 11). *Shortest job next*. Retrieved from Wikipedia: [http://en.wikipedia.org/wiki/Shortest\\_Job\\_First](http://en.wikipedia.org/wiki/Shortest_Job_First)
- Wikipedia. (2015, 03 11). *Fibonacci number*. Retrieved from Wikipedia: [http://en.wikipedia.org/wiki/Fibonacci\\_number](http://en.wikipedia.org/wiki/Fibonacci_number)
- Wikipedia. (2015, 02 17). *INVEST (mnemonic)*. Retrieved from Wikipedia: [http://en.wikipedia.org/wiki/INVEST\\_\(mnemonic\)](http://en.wikipedia.org/wiki/INVEST_(mnemonic))
- Wikipedia. (2015, 03 10). *Planning poker*. Retrieved from Wikipedia: [http://en.wikipedia.org/wiki/Planning\\_poker](http://en.wikipedia.org/wiki/Planning_poker)
- Wikipedia. (2015, 03 11). *SMART Criteria*. Retrieved from Wikipedia: [http://en.wikipedia.org/wiki/SMART\\_criteria](http://en.wikipedia.org/wiki/SMART_criteria)



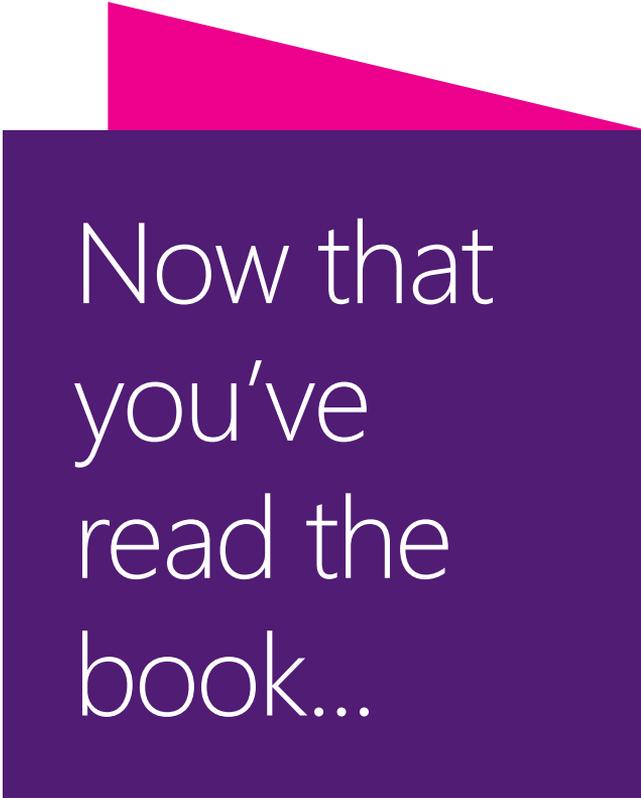
From technical overviews to drilldowns on special topics, get *free* ebooks from Microsoft Press at:

**[www.microsoftvirtualacademy.com/ebooks](http://www.microsoftvirtualacademy.com/ebooks)**

Download your free ebooks in PDF, EPUB, and/or Mobi for Kindle formats.

Look for other great resources at Microsoft Virtual Academy, where you can learn new skills and help advance your career with free Microsoft training delivered by experts.

**Microsoft Press**



Now that  
you've  
read the  
book...

Tell us what you think!

Was it useful?

Did it teach you what you wanted to learn?

Was there room for improvement?

**Let us know at <http://aka.ms/tellpress>**

Your feedback goes directly to the staff at Microsoft Press,  
and we read every one of your responses. Thanks in advance!

