Switching to Scrum

How to Implement Scrum in your Software Development Organization

1st Edition

William Patrick Swisher

October 2012

(v1.1.3 PDF)

Cover design by Marji Drost.
© 2012 William Patrick Swisher

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic, or mechanical including photocopying, recording, or by any information storage and retrieval system, without the written permission of the author, except where permitted by law.

http://SwitchingToScrum.com/

author.comments@gmail.com

Cover design by Marji Drost.

Printed in the United States of America.

ISBN: 1480025224
Introduction ................................................................. 8

Example Organization Description ................................. 10

Author Feedback ............................................................ 10

Analysis, Decisions and Messaging ................................. 11

Why Use Scrum? ............................................................... 11

Analysis and Establishing a Baseline ............................... 12

Conducting a Pre-Transformation Survey ........................ 12

Mapping the Product Delivery Workflow .......................... 14

Initial Transformation Activities ....................................... 15

Business Requirements under Scrum ............................... 22

Product Management Changes ....................................... 22

Product Line Roadmap .................................................. 23

Business Requirements Under Scrum ............................... 23

Decomposing Product Concepts ...................................... 24

Writing Backlog Items .................................................... 27

Backlog Item Scope ....................................................... 27

Backlog Item Components .............................................. 32

Construction of a Product Backlog ................................. 36

Defects in the Product Backlog ....................................... 42

Converting Existing BRD Requirements into Backlog Items 44

Initiating Scrum Development (Sprint Zero) ................. 48

Team Kickoff Meeting ................................................... 48

Definition of Done ........................................................ 51
Establishing Knowledge, Tools, Processes, Environments, Access and Meetings ................................................................. 53

The First Story Time Meeting ................................................................. 55

Scrum Development Continues (Sprint 1 to n) ......................... 57

Sprint Planning .................................................................................. 57
Daily Standup Meetings ................................................................. 64
Sprint Review Meetings ................................................................. 68
Sprint Retrospective Meetings ....................................................... 74
Relative Effort Estimates ............................................................... 77
Managing Defects ......................................................................... 86
Changing the Scope of the Current Sprint ........................................... 87

Inspection and Adaptation ............................................................... 89

Daily Standup Meeting ................................................................. 89

Burndown Chart Trends ................................................................ 90
Daily Standup Email Updates ......................................................... 92
Transformation Oversight Group (TOG) ........................................... 92
Common Transformation Problems ................................................. 97
Whole Organization Analysis .......................................................... 104
Scrum Resources Repository .......................................................... 106

Expansion ....................................................................................... 109

Starting Additional Scrum Teams ................................................ 109
Team Synchronization and Innovation ........................................... 112
Increased Disruption to the Organization ...................................... 113
Sprint Planning Meeting Agenda - Sample ..........................................................136
Sprint Review Meeting Agenda - Sample .........................................................137
Sprint Retrospective Meeting Agenda - Sample .................................................138
Story Time Meeting Agenda - Sample ..............................................................139
Scrum Masters’ / Product Owners' Club Agenda - Sample ..................140

Appendix C - Process / Flow Diagrams ......................................................141

Daily Standup Meeting - Process Diagram ....................................................141
Sprint Planning Meeting - Process Diagram ...................................................142
Sprint Review Meeting - Process Diagram .....................................................143
Sprint Retrospective Meeting - Process Diagram .........................................144
Story Time Meeting - Process Diagram ..........................................................145
Backlog Item States - Flow Diagram ...............................................................146
Introduction

Transforming a complex system such as a software development organization from one development methodology to another is neither trivial nor guaranteed once it begins. The transformation will require steadfast commitment from senior leadership and a willingness to put significant effort into setting it up for success, monitoring progress and making the necessary adjustments along the way.

Waterfall and Scrum methodologies are fundamentally different and the process of switching from one to another will be very disruptive not only to the software engineering department but the entire product delivery organization. Fortunately, because of the nature of Scrum, which produces tangible results early on, the perceived benefits will further drive acceptance and the organization soon experiences the “bandwagon effect” where more and more people realize the value of the new processes.

In the beginning there will be resistance both at the individual and departmental level and so an important part of making the transformation is to monitor the upstream and downstream effects of changing the Software Engineering department and continually make adjustments across the organization to effectively acclimate everyone with the new methodology.

This book was written based on real world experience of guiding software companies through the transformation. The approach and experiences described in this text come directly from working with organizations that successfully made the transformation.

The target audience for this book is senior engineering leadership, such as a CIO or VP of Engineering who are in the position to set the strategic goal of changing software development methodologies. It was envisioned that the processes and artifacts described in this text could be used by managers who directly supervise teams and could influence the transformation process at the direct contributor level.
The topics and layout of the book will take the reader through the transformation process as it would be executed. Starting first with making key decisions and the messaging for the organization setting the tone for the entire effort. Then some analysis to establish a baseline and identify shortcomings for later evaluation of success. Preparing artifact templates, designating the size and composition of teams and deciding about sprint cycles and calendars will finish off the initial preparation phase.

Then we turn our attention to preparing Product Management to create business requirements in the format necessary to supply the Scrum Teams with work. This too is disruptive, but will ultimately bring the two departments of Product Management and Software Engineering closer, improving how they work together.

When the first Scrum Team starts up, it will be important to establish the proper infrastructure and expectations and ensure the team understands what they’ll be working on and how. Next, all the essential rituals necessary to build new working software, address defects, and managing change during a development sprint will be explained.

Continual monitoring of progress at the team and organizational level will be covered with special focus identifying issues and innovation and dealing with commonly experienced problems.

Success with the first Scrum Team will then lead to starting up the second, third and so on. Dealing with coordination and shared learnings will be important as will how the rest of the organization will deal with the increased throughput of product delivery.

Measuring success after the transformation process has been underway will be an important step to determine if the organization as a whole has benefited from the change. How to determine the results as well as signs of success at the team and overall organization will be covered.

Finally, sample artifacts to help kickstart the transformation process are presented including essential scrum artifacts, meeting agendas and process / flow diagrams to help educate and explain.
Example Organization Description

Examples throughout the text are all based on a fictitious software development company so that they all relate to one set of circumstances. The reader is encouraged to read about this company in the following text in order to have the context to better understand the examples.

Apptastic, Inc is a small startup company with about 50 employees. It has the basic functional groups of a professional organization (Human Resources, Marketing, Information Technology, Software Engineering, Product / Project Management, Accounting, etc) but because of its small size, a number of people wear many hats and can pitch in to help where needed.

The Software Engineering team however is comprised of specialists in either programming or quality assurance with each group reporting to a separate Engineering Manager. Software projects are completed using the Waterfall methodology where the Product Managers define products and write up business requirements and pass them to Engineering for implementation.

Their upcoming product, Weatherly, should be popular with jet setting executives and sales people as it provides timely weather condition updates based on the user’s calendar entries. A recent market analysis indicated that the Weatherly application will have some serious competition and also needs to work on multiple phone platforms. Beating the competitors would require Apptastic to overhaul the product delivery process and make the organization as efficient as possible. The CIO, Mr. Hawking decided to shift the engineering organization over to Scrum with the goal of doing so within the next quarter in order to give the company a fighting chance of success.

Author Feedback

The reader is encouraged to send feedback about this book so it can be incorporated into the next edition. There are two ways to do so:

- Email author.comments@gmail.com; indicate what you liked about the book and provide suggestions on how it could be improved.
- Visit http://SwitchingToScrum.com/ and submit comments through the website.

All feedback is valuable and greatly appreciated.
Chapter 2 - Analysis, Decisions, and Messaging

Analysis, Decisions and Messaging

Making the transformation from Waterfall to Scrum isn’t all that easy but certainly not impossible. What’s needed most is a dedicated effort to push through the challenges that lay ahead. In this chapter, we will cover the fundamental analyses, decisions and messaging necessary to start the transformation successfully.

Why Use Scrum?

There are a number of software development methodologies available each with their own benefits which may suit any particular organization better than Scrum. However, Scrum is becoming widely practiced and with more software developers and technical management familiar with the methodology, it has become easier to implement since the concepts aren’t as foreign as they once were.

Scrum offers significant benefits over Waterfall, chiefly among them is visibility. With the Daily Standup meetings to coordinate work between team members and identify impediments, issues that would otherwise block progress are exposed and addressed. A team no longer has the option of waiting as they must complete work within the sprint timeframe which drives the resolution of issues.

Demonstrated progress at predictable intervals is also highly favored as completing work boosts team morale and shows management tangible results early and often. Under Waterfall, the beginning of a project is focused on understanding requirements and designing with little code being written. From the perspective of senior management, this ties up costly resources whose primary output is preparation to deliver, not the actual deliverables themselves. With Scrum, since the team focuses on the most valuable parts of a project first, the business can see results which may in turn inspire new ideas and refinements to the overall project.

With Scrum’s design of iterative development cycles, it supports the business’s need to respond quickly to marketplace changes and so long as the Product Backlog reflects those needs, the development team will work on the most valuable work in any given sprint. Additionally, because the
sprint cycles tend to be short periods of time, if priorities change in the middle of the sprint the business doesn’t need to wait long until the next cycle to address the most pressing needs.

A more cohesive sense of teamwork is characteristic of Scrum because the group of Developers, which includes everyone necessary to deliver completed software at the end of a sprint, select the work they will take on within the sprint and strive towards that goal. This self-determination and reliance on each other to complete the work creates collaboration and ultimately high performance teams.

Lastly, the Scrum methodology itself is very simple. There are few formal rules and the artifacts and rituals are straightforward and easy to understand. The most challenging part of Scrum is the mental shift required from working under Waterfall and really begin to work together as a team to deliver completed software each sprint. It doesn’t come easy for everyone but with enough time and attention to introspection and adaptation it can be successful.

Analysis and Establishing a Baseline

Now that the decision has been made to move forward with the transformation, some analysis work is recommended in order to establish a baseline of how the organization operates before any changes. These efforts will identify current shortcomings and help to measure success after some time operating under the new methodology.

Conducting a Pre-Transformation Survey

Using a survey is a quick and effective method for capturing peoples’ assessment of the organization and how it functions from their perspective. Since the transformation will affect many aspects of the organization, directly or indirectly, the survey should be designed for general distribution and not just a specific group like Software Engineering or Product Management.

There are two goals for these surveys, the first being to identify specific shortcomings that plague the organization now in order to make sure they are addressed as the transformation progresses and secondly, to establish metrics that can be used for comparison some time after the transformation started.
Survey Content

The purpose of this survey is to assess how the organization views the various part of the product development process. It should focus on different aspects of the process, not just Software Engineering and ideally be given to everyone directly involved in the product delivery chain from product ideation, definition, development, deployment, client implementation, and product support. Since collecting specific examples of shortcomings is also important the survey should include freeform text input fields for any respondents' comments. The following are suggested topics and they can be used or modified as necessary for different organizations.

- **General** - Capture the various roles of the respondents so targeted solutions can be applied where needed.
- **Visibility** - How well do the participants feel they know the overall strategy of the company, various product lines and the progress being made on active projects?
- **Predictability** - How well are products being delivered within expected timeframes and contain the necessary features and fixes?
- **Clarity** - How well have product requirements been defined and how can people get answers when development issues arise?
- **Communication** - How effective is communication across the organization and coordination between teams?
- **Quality** - What’s the perception of quality delivered for product releases?
- **Sustainability** - Are people working at the right pace or putting in additional hours just to keep up?
- **Productivity** - Do the tools, processes, and resources of the organization support the product delivery process?

After the survey has been administered and the data collected, some analysis can be done to look for shortcomings within the organization. These can then be given special attention throughout the transformation process.
Mapping the Product Delivery Workflow

The next step in analyzing the organization before the transformation is started is to create a *product delivery workflow*. By mapping this out visually it makes it easier to identify the current shortcomings as discovered from the survey and investigation.

Ideally the workflow should include all significant steps in the creation and delivery of a product. The easiest way to do this is to identify the departments and break out how they contribute to building the product. Each department does and how their part flows into the product. The following example is based on the company Apptastic where it is primarily the Product Management department that creates new products ideas, writes requirements and pitches the idea to Senior Management who decides to allocate resources to the project. Once software engineering resources have been designated, the Project Management group takes control and drives the project through handoff to Operations who deploys the new product for customer use.

While it may seem logical to stop the product delivery workflow when the product is delivered, that’s not really the end of it and having a complete picture is important because a change in development methodology will affect the entire product lifecycle.

After the product delivery workflow is complete, the next step is to collect and document the shortcomings for two purposes. The first being to see which naturally disappear once the new development methodology is in place. The second is to make sure that those that don’t initially disappear during the transformation are not forgotten and are actually addressed.
Identifying Workflow Shortcomings

The pre-transformation survey should have exposed some shortcomings and certainly many people will have identified such issues over time. They should be collected into a checklist for periodic review and to see if the transformation addresses them. If not, the new development methodology can be tweaked to provide a resolution.

<table>
<thead>
<tr>
<th>Functional Area:</th>
<th>Shortcomings Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Management</td>
<td>Business requirements documents are often handed over to Development with parts missing or very vague.</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>The QA Engineers often feel like they’re lost when code is turned over for testing because they’re rarely in the loop when code is being designed.</td>
</tr>
<tr>
<td>Operations</td>
<td>Code frequently doesn’t have the necessary logging or mechanisms where monitoring and alerts can be set up and used to check the health of the product.</td>
</tr>
<tr>
<td>Technical Support</td>
<td>A lot of feedback comes from customers through Technical Support but there’s no channel for this feedback to make it into new product versions.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

This list of shortcomings will provide good starter discussions for when the Transformation Oversight Group, discussed later in Chapter 6, convenes and begins looking at how the transformation is progressing and what issues should be addressed.

Initial Transformation Activities

Before the transformation begins there are a number of initial activities required to establish the first Scrum Team and how they will operate before an initial start date can be selected.

Training

Within any given software development organization there will be a wide range of experience with Scrum. To ensure everyone has at least the same basic level of understanding, those people who will be assigned to Scrum teams should attend training sessions. There are many Scrum fundamentals classes available offsite at any given time but ideally
having the trainer onsite to educate large groups at a time is very effective and ensures the participants hear the same terminology and explanations.

Developers and Managers can benefit from a general Scrum training course whereas potential Product Owners, typically Product Managers, should also receive training specific to the Product Owner role. This training won’t be inexpensive but the benefit of having everyone trained at the same time and way will minimize the problem of everyone wanting to follow Scrum the way they’re most familiar with.

Artifacts

Chances are that an organization that has been operating under a Waterfall development methodology won’t have the various Scrum related artifacts ready to use. Examples of Product Backlogs and Sprint Backlogs are readily available on the Internet but may be far more complex than needed by an organization new to the process. It is better to define simple versions that suit the immediate needs and expand them as the teams learn.

Once a Product Backlog template has been created, the first Product Owner will need to populate it with backlog items. They may feel overwhelmed to have to create an entire set of requirements in the new format and should be reminded that only a handful will be needed for the first sprint and that more can be written after the Scrum Team starts working. Chapter 3 covers creating a Product Backlog and writing backlog items, including converting existing Business Requirements Document content into usable material for the Scrum Team.

Messaging

One of the most important factors in making the transformation successful is how it is messaged to the entire organization. The message should be along the lines of:

- We’ve looked at how we can improve product delivery, respond to a rapidly changing marketplace and address our customer needs.
- We’ve decided to switch our product development methodology to Scrum with a target of having [number] of teams completing their first sprint by the end of the [time period].
Chapter 2 - Analysis, Decisions, and Messaging

- To facilitate this transformation we have arranged for training for all people directly involved in the development of products and hired experienced Scrum Masters to ensure we are adhering to the standards and follow best practices.

- The transformation will not be easy; it will take hard work for all of us and Senior Management is fully behind this change and confident in our ability to be successful.

- It is completely expected that there will be bumps in the road. As long as we’re making progress in identifying issues and making improvements we are sure to complete the transformation and come out a stronger and more resilient product delivery organization.

Despite the message being heard by everyone in the organization, there will be people that resist the change. To diffuse any growing opposition, provide a channel through which issues, grievances, and questions can be addressed such as a primary contact (e.g. PMO Director), email address (e.g. Scrum-Suggestions@YourCo.com), or set up an anonymous suggestion box for comments. Take all issues seriously as it will lend credence to the iterative nature of Scrum for introspection and adaptation.

For any proposed radical diversions from the newly established practice, make it clear that they will have to be evaluated by an executive steering committee. By setting this bar very high it will reduce the number of frivolous suggestions.

Lastly on messaging, it is vital that all Engineering Managers are of one voice in supporting the transformation. If there are any cracks, the dissenters will seek them out prolonging the transformation process.

Size and Composition of Scrum Teams

The size and composition of a Scrum Team will depend entirely on the nature of the work they will be doing. There are however some guidelines worth keeping in mind.

The ideal team size is around five to seven people; any more than that and the amount of communication overhead between team members may involve too much effort to keep everyone informed. Additionally, any meetings such as the Daily Standup, which will be covered in Chapter 5,
runs the risk of becoming too long to provide the quick collaboration and coordination necessary between members. If at all possible, if the team members can be collocated, that is, all sitting near each other, this arrangement can have a dramatic effect on how well they communicate during sprints.

Regarding the composition of the team, this will depend on the skill sets necessary for the anticipated work. Assuming that the long term strategy is to have multiple teams with each focused on a product line or specific high profile product, then selecting each resource is a matter of considering some of the following factors:

- **Product Familiarity** - How familiar is the resource with the existing code base and the potential technical direction it will take?
- **Company / Industry Experience** - How familiar is the resource with the company’s history and industry experience?
- **Career Experience Level** - At what level is the resource’s professional experience?
- **Leadership Qualities** - How well could the resource take the lead or mentor the more junior members of a team?
- **Scrum Experience** - How much experience does the resource have with Scrum?
- **Coding / Testing Work Ratio** - Given the nature of the products, what is the right ratio of coding to testing skills that will keep the team balanced enough such that people don’t run out of work each sprint?
- **Secondary Skill Sets** - What other skills besides coding and testing will be required just often enough to justify permanent presence on the team even if they will only spend a sub-set of their overall working time on team projects?
- **Personality** - How will candidates react to, and work with, other specific individuals?

Once team starts working together they should be monitored and if necessary, personnel adjustments can be made early to fine tune the right mix.
Chapter 2 - Analysis, Decisions, and Messaging

Sprint Cycle Days and Length

When an organization is first switching to Scrum the natural tendency is to align sprints with the calendar week, that is, starting on a Monday and ending on a Friday. It seems logical given that the workweek also follows this pattern but there’s a better choice.

Because it’s fairly common that employees take Mondays and Fridays off and some holidays also fall on those days, the result will be a number of Sprint Planning and Sprint Reviews being missed by team members. Assuming a two week sprint, the following diagram shows what it would look like. A ‘B’ indicates the beginning of a sprint and an ‘E’ designates the end.

<table>
<thead>
<tr>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>T</td>
<td>W</td>
<td>R</td>
</tr>
<tr>
<td>B</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

A good alternative sprint cycle is either Wednesday to Tuesday or Thursday to Wednesday as illustrated below.

<table>
<thead>
<tr>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>T</td>
<td>W</td>
<td>R</td>
</tr>
<tr>
<td>...</td>
<td>E</td>
<td>B</td>
<td>...</td>
</tr>
</tbody>
</table>

With this cycle, Sprint Planning begins mid-week and in the remaining days of that first calendar week, most planning is done and the team has time to start executing work. By the time the weekend arrives the team has a solid understanding of what they need to accomplish and have built up momentum.

Then in the next calendar week there’s five contiguous days (see above black box) where everyone can focus on accomplishing sprint work. The remainder of the sprint spills over to the next calendar week and the team can wrap up work and get ready for the Sprint Review. It may seem like there’s little difference between a Monday to Friday sprint cycle or the alternatives but there’s a great benefit to shifting the sprint cycle and giving the team five solid days of minimal meetings and distractions.
In the examples above, a two-week sprint cycle was shown. This could be extended to a three or even four week cycle which would give even more contiguous working days:

<table>
<thead>
<tr>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>M T W R F</td>
<td>M T W R F</td>
<td>M T W R F</td>
<td>M T W R F</td>
</tr>
<tr>
<td>... E B ...</td>
<td>... ...</td>
<td>... ...</td>
<td>... E B ...</td>
</tr>
</tbody>
</table>

However, keep in mind that with a longer sprint cycle, Sprint Planning and Sprint Reviews will take more time and if business needs change mid-sprint, there will be a longer wait until the team finishes current sprint work and is able to respond to the latest business priority.

In general it’s best to start with the shortest reasonable sprint cycle and that can be determined by looking at what standard activities will be needed to deliver completed work each sprint and how long they will take. For example, if completing regression testing for most code updates take a few days, then a two week sprint may not be long enough to get productive work done in time for testing. Keep in mind though that the processes of today may not be the processes of tomorrow especially as the teams focus on completing their first couple of sprints and find efficiencies to minimize testing effort.

There will always be a struggle against the new timelines the team has to work within; be mindful of their struggles and look for opportunities to improve their productivity so they don’t feel like there’s no hope of completing work within each sprint.

**Selecting the First Project Under Scrum**

When deciding what project will be the first run under the Scrum methodology there’s really only two choices, one of lower business priority or one of high business priority. It’s easy to think that by taking on the lesser priority it’s a good way to try out the new development process without causing any harm. This seems reasonable but it also means that when the Scrum Team has impediments the organizational drive to clear them will be absent and the transformation process will lose momentum and fail.
It is always best to select a project for which there is broad support and interest in seeing it succeed. Of course this means more eyes will be watching the progress, and results, and any failure may be quickly attributed to the Scrum process but the benefits outweigh the risks because the team will have far more support in completing the transformation.
Business Requirements under Scrum

Before the first Scrum Team convenes and starts development under the new methodology, there must be business requirements ready with which to work. Because of the nature of Scrum, previously defined requirements won’t be in the best format for the new process. Some conversion of existing requirements and writing of new requirements will be necessary by the Product Owner.

Product Management Changes

In many software development organizations Product Managers wield great influence in the activities and priorities of the engineering teams. They may not have Engineers as direct reports but because Product Managers represents the needs of the business, they may direct where resources and effort are spent.

Product Managers will have plenty of ideas ranging from initial-concepts to requirements-ready at any given time and often make use of Software Engineering resources to research, prototype, and develop those ideas. Engineers may find themselves assigned to, or involved with, more than one project at a time. Under the Scrum methodology however, the ideal is to have a Scrum Team focus solely on one project at a time which minimizes context switching and therefore increases their time delivering business value. Additionally, because they’re working as a team to design, build, test and deliver, there is a higher throughput of completed work.

The switch to the Scrum development methodology will require the Product Owners to get their ideas organized and ready for the Scrum Team ahead of time. Since tapping Software Engineers for various product ideas all at the same time isn’t efficient, they will need to refine and prioritize their ideas according to the value those ideas bring to the business. The Scrum Team can be an integral part of the process of defining business value utilizing their experience in building products. Any Product Owner would be wise to share their business goals with the entire team and be open to feedback about how to achieve them.
Product Line Roadmap

Product Owners are typically responsible for a Product Line which could consist of multiple versions of a single product or variations of a product created for different market segments. In any case, creating a Product Line Roadmap that represents the high level strategy is an essential part of being able to define what the Scrum Team will focus on and in what order. A visual representation of a Roadmap usually shows the products mapped on a calendar:

The primary feature components may be included in the diagram or simply listed nearby so the viewer can understand what’s included in each product version.

Notice that left side of the calendar has time increments in months whereas the right side is in quarters and could even be in years. This is because while the Product Owner knows what he wants long term, he has more visibility and more details around what could happen in the short term. If the Scrum Team existed already and they had some measure of Velocity, which is the rate at which they deliver business value each sprint, the Product Owner could use this data to more accurately map out where products are likely to occur on the calendar. In the beginning of the transformation from Waterfall to Scrum however, the Roadmap shows what the Product Owner desires and he periodically refines it as business strategy change. It should be shared with both management and the Scrum Team on a regular basis to convey the overall business strategy.

Business Requirements Under Scrum

In the world of Waterfall, product requirements are defined in a Business Requirements Document (BRD). These documents are designed to represent everything that the new product should include and often
contains minute details. There is no universal standard for the contents of a BRD but they typically have some high level explanation of the product, justification for its creation, and a breakdown of the specific functionality or characteristics required for the end result.

Creating a BRD isn’t a trivial exercise because a Product Manager has to capture the entire set of requirements and it can be very challenging to think of everything needed up front. Of course, as the project gets underway, changes will be needed for any number of reasons and those changes have to be incorporated into the BRD so there’s a record of what the final result is supposed to be. Additionally, the changes will have to be conveyed to the project team who may have to re-think significant amounts of work already planned, in progress or completed.

Under the Scrum methodology, there is far more flexibility in defining product requirements as it is an ongoing process. Because not all requirements are needed up front before the Scrum Team can begin to analyze and plan their first sprint, they can get started sooner. One of the key differences with Scrum is the expression of requirements in terms of business value versus specific elements of functionality. A successful Product Owner will work closely with the Scrum Team, sharing the business strategy including specific characteristics required and solicit input for execution. This is quite different from the often complained about experience of Product Manager “throwing requirements over the wall” and coming back into the picture when the project is nearing completion. Under Scrum, the Product Owners are much more engaged and work with the team to define the right-sized units of business value they can deliver within a sprint. The end result of this collaboration with the team is the writing of optimally constructed and sized backlog items.

**Decomposing Product Concepts**

Before we get into the specific structure of a backlog item and the best way to write them, one of the challenges a new Product Owner will face is how to convert a product concept into backlog items that are of the right size and composition to be completed within a sprint.

Returning to the *Weatherly* project example, an ideal first step is for the Product Owner to solidify the overall product concept in their minds so they can easily explain it to the Developers. This exercise can be thought of as writing an ‘elevator pitch’ where you have to convey what the
product is, what it’ll do, who will use it, and why it will be of value to customers. For example:

“Weatherly is a smart phone application that provides the user with timely weather information, clothing suggestions, and late-breaking alerts based on travel destinations found in their calendar so they are never caught unprepared for local weather conditions.”

For those readers familiar with Scrum already, you may notice the format looks a lot like a user story and that is no coincidence since the format is excellent for conveying the business value in plain language. User stories will be covered in more detail later in this chapter.

The next step is to define the features as the customer would experience them and create a list. This is where the process gets tricky. The Product Owner should avoid listing technical requirements and instead describe the features in terms of the value they provide to the customer. It may help to think about how the product would be marketed or the advertisement copy written to highlight features. Using the Weatherly example, the feature list would be:

- Provides current and forecasted weather conditions for the destination.
- Makes suggestions regarding clothing to bring.
- Automatically identifies travel destinations on the user’s calendar.
- Sends alerts regarding changes in weather just before the trip.

The first three describe discrete, single benefits for the user but the fourth one regarding alerting triggers questions about the kind of alerts and how they’d be delivered. Because of this, it should be broken down further. In Scrum parlance, this would be called an Epic, as in, containing more than one user story and are perfectly fine to include the Product Backlog with the understanding that it will need to be decomposed into more discrete concepts before the Developers can work on it. After breaking it down further, four separate benefits were identified:

- User configurable alerting options for method of delivery and timing.
- Alerts via voice message.
- Alerts via email.
• Alerts via text message.

These seem like the right level of granularity as they now describe discrete features. Once all of the features listed have been evaluated and if necessary, split up, the entire list should be ordered according to the business value they provide. Even if the Product Owner cannot imagine declaring the product ready for users until all of the listed features were included, the list should be sorted. Doing so is relatively easy if they ask themselves the question: “If I could only have one feature delivered, which would it be?”. The Product Owner may have to consider the needs of the market, return on investment, political considerations, and their own desires but ultimately one will be chosen. The question is asked repeatedly until the list is assembled in business priority order.

1. Automatically identifies travel destinations on the user’s calendar.
2. Provides current and forecasted weather conditions for the destination.
3. Makes suggestions regarding clothing to bring.
5. Alerts via email.
6. Alerts via text message.
7. Alerts via voice message.

Ordering the features is important because it defines the order of the Product Backlog which in turn represents the order that the Developers will complete the work. If project resources were cut unexpectedly the work delivered up to that point would be the most valuable and potentially enough to launch the product with.

The final step in breaking down a product concept into business value is to look at the ordered list of backlog items and decide at which item would there be sufficient business value delivered to bring the product to market. For example:

1. Automatically identifies travel destinations on the user’s calendar.
2. Provides current and forecasted weather conditions for the destination.
3. Makes suggestions regarding clothing to bring.
5. Alerts via email.

*(Everything above this line is in Weatherly iOS v1.0)*

6. Alerts via text message.
7. Alerts via voice message.

Everything above line six is absolutely essential to call it a first release and everything below it is optional. This list of essential features is typically referred to as the Minimum Marketable Features (MMF). It may be that the Product Owner wants all the features completed ultimately but deciding which ones are absolutely essential is important because changes and additions will occur and each change should evaluated to determine if it belongs within the MMF. As the list of backlog items change over time, the Product Owner should continually maintain or “groom” the Product Backlog so it always represents business priorities.

**Writing Backlog Items**

Once the features have been identified, listed and sorted according to business priority, the next step is to write the backlog items. Each one of the previously defined product features can be turned into one or more backlog items. Exactly how to write backlog items with the right details and scope is refined overtime by the Product Owner working with the team. When the Developers understand the business needs, they can use their expertise to help the Product Owner refine backlog items and create manageable sprint work.

**Backlog Item Scope**

There are basically four sizes, or levels of scope, that a backlog item can represent when describing business value.

1. **Complete Feature**

The ideal scope for a backlog item is to represent a feature in it’s entirety, that means once done, it is complete and ready for customer use. This does not include enhancements that may be needed later, only the functionality and characteristics needed for a v1.0 release.
Using the Weatherly product as an example, a backlog item could represent all functionality and product behaviors necessary to access a user’s travel destinations from the calendar, retrieve the weather data from the weather service database, and display the information within the user interface.

When delivering the complete functionality of a single feature may be too much scope for backlog item to be finished within a sprint, the next option is to define slices of functionality.

2. Slices of Functionality

When a backlog item cannot possibly include everything for a given feature, the next level of scope downward is to define thin slices of functionality that represent an end-to-end experience. It may take several of these slices to complete a feature.

Returning to the Weatherly example, if the application required the user to enter a username and password, a thin slice of functionality would cover the user experience of entering the information on the user interface, clicking a button to submit it, all the backend work to validate the account and any display necessary to the user letting them know it worked.

Writing backlog items for something that can focuses on interaction with a user is relatively easy. The more challenging ones deal with backend processes where describing an experience isn’t as straightforward. Feature #2 for the Weatherly application describes the core function of the product: retrieving weather information for a given destination.

2. Provides current and forecasted weather conditions for the destination.

For this feature, the phone application has to connect through the Internet to the services that provide weather information. Diagramed out, the process flow looks like the following, starting with the Phone App submitting a request on the left side:
Chapter 3 - Business Requirements under Scrum

The thinnest possible slice of functionality that traverses the entire process would be:

- Allow the phone application to submit one data element of a weather conditions request (e.g. destination city) to the App Gateway.
- Log the application request in the Transaction Logging Database.
- Pass the data element to the Lookup Engine which reads the database for a match.
- Log the result in the Transaction Logging Database.
- Send a response to the phone application with the current weather conditions (temperature, humidity, cloud cover) only.

Diagramed out, the thin slice of functionality touches every part of the process:

The final functionality that completes a lookup for weather information will have much more data sent in the request and a complete response that the user interface will display to the user. This thin slice of functionality covers just a fraction of the desired behavior but at least demonstrates how the various components of the process will work together. Of course, splitting a feature’s functionality across backlog items in this way may require some re-work by the Developers because what
they create to satisfy one backlog item may not be the best design for a completed feature. The following diagram illustrates this point:

The overlapping areas represent changes that will be necessary between delivery of slices of completed work. The App Gateway and Lookup Engine will need to handle more data elements and the Weather Database’s schema may have to be redesigned or at least expanded with each new layer of functionality.

This may seem wasteful but there’s a lot of value in doing thin slices. First, the Product Owner gets to experience what the user will ultimately see and that may inspire new ideas. Second, any technical or process issues that could hinder the product are exposed and then can be addressed earlier in the development process. Lastly, by delivering product functionality that is potentially ready for customer use, the Product Owner can decide to stop adding functionality and ship earlier than expected.

When slices of functionality aren’t small enough to be completed within a sprint, the scope can define functional components.

3. Functional Components

Depending on the product, it may be very difficult to complete an end-to-end slice of functionality within a sprint. There are many factors including sprint length, experience of the Scrum Team, complexity of the systems involved, being able to set up prerequisite environments, et cetera. The next best option is to concentrate on completing the work on one step of the process at a time, usually starting at the beginning of the process. The work would then include:

- Allow the phone application to submit all data elements of a weather conditions request to the App Gateway.
This won’t results in potentially shippable work but at least meaningful progress is being made.

If backlog items are focused on functional components, and these components are being built across sprints, the team will need to verify that previously completed components are working with the latest ones in the current sprint. This testing effort should be considered and included in each subsequent sprint to ensure that once they reach the end, the entire feature works as expected.

When working on a functional component still isn’t possible, focusing effort on foundational work is a good option for setting up the right conditions so that future backlog items can deliver complete business value.

4. Foundational Work

Backlog items of this scope, may include research, designing, database schema definition, or even setting up environments for future effort. It is the furtherest away from directly delivering business value but may be completely necessary for some products.

Keep in mind though that under Scrum, the goal is to deliver potentially shippable work each sprint so focusing the scope on Complete Features or Slices of Functionality is best. When it is necessary to reduce the scope to Functional Components or Foundational Work, the Product Owner should make it clear to the team that the work completed should support returning to the more complete levels of scope as soon as possible.
Backlog Item Components

Now that we have defined how to break up business needs into manageable work, it is time to explore what is included in a backlog item.

Well-constructed backlog items have four main components:

- **User Story** - A narrative description explaining who will use it, what is desired and why it is needed.
- **Acceptance Criteria** - The specific characteristics or behaviors that will be needed in order for the Product Owner to accept the backlog item as complete.
- **Supporting Artifacts** - Any documents or sources of information necessary to deliver what the backlog item describes.
- **Questions & Answers** - A listing of questions and answers that define what the Developers needed to know in order to implement the backlog item.

User Stories

*User stories* are a key component of a backlog item because they provide context for the business value desired. They are typically written in a formulaic way defining for whom the functionality is intended, what exactly is wanted, and how it will be used. The typical format follows this pattern:

As a `<role>`,
I want to `<action [or] experience>`,
so I can `<explanation of why>`.

By presenting the business value in the context of a role and their desire to accomplish something, the requirement is easily understood and establishes a tangible goal for the Developers. The user story should answer the questions of *who, what,* & *why*. Even if the Developers have a complete set of acceptance criteria and sufficient details, by knowing the context and business value desired it will enable them to suggest improvements or alternatives that meet the need.

Not all functionality needs to be described in terms of an end user’s experience; some can be described from the perspective of the system or
business owner if it helps to explain the business value desired. The role is just part of the user story and helps provide an explanation of who might benefit from the business value being requested. Following are some examples of user stories based on the Weatherly product described previously:

- As a user, I want to have current weather conditions for my travel destination so I can get a sense of what it’ll be like for my trip.
- As the system administrator, I want to have application requests logged in the Transaction Logging Database so I can run reports showing traffic activity.
- As the Lookup Engine, I want to process all requests as quickly as possible so the user has a positive experience.

The last one, written from the perspective of the Lookup Engine, could also have been written from the perspective of the user:

- As the user, I want my transactions processed as quickly as possible so I feel like the application is responsive.

When deciding which role or perspective will work best in a user story, focus on who the immediate beneficiary of the functionality will be.

The key to writing good user stories is to make the business value or end goal very clear and understandable to anyone reading it. In each sample above the business value is clear but there aren’t enough details for the developer to begin coding. Those details are included in the acceptance criteria.

Acceptance Criteria

Acceptance criteria describe the precise conditions under which the backlog item will be accepted by the Product Owner at the end of the sprint. Where the user story defines the who, what, & why, the acceptance criteria can answer where, how and when. They are typically listed as single lines representing a discrete and testable deliverable. To avoid any ambiguity, direct commanding language must be used. The following examples correspond to the first user story example in the last section:
• The username data element must accept between two and 30 alpha characters and not contain spaces or escape characters.
• All submitted data elements must be converted to lowercase before processing and storage.
• Responses must include a results code indicating success or the type of error occurring. (See Results-Code-Table_v3 file stored on network project folder for entire list.)
• Phone application requests along with the data elements submitted must be stored in the Transaction Logging database.
• Responses must include a transaction ID generated from the database when the request was logged.

If a single backlog item’s list of acceptance criteria grows too long, the backlog item itself may be too large and splitting it into two or more based on logical divisions should be considered. What constitutes “too large” will be based on the feedback from the Scrum Team. Additionally, a single acceptance criterion may represent significant work and be turned into a new backlog item. When the Scrum Team participates in Story Time meetings, explained in Chapter 4, they will preview upcoming backlog items and provide the Product Owner with feedback that may result in changes.

The reader may have noticed that the third acceptance criteria referenced a file stored on the network. This is an example of a Supporting Artifact that contains far more information than is ideal for including directly in the backlog item and is therefore called out specifically.

Supporting Artifacts

Backlog items will sometimes need a lot of details in order to be completed and while much can be put into the acceptance criteria, there comes a point that having those details within the backlog item becomes troublesome to update and reference. There’s no hard and fast rule around what should be included but in general if Developers need to study many details in order to plan their implementation strategy, those details are best stored in a separate document. For example a few business rules listed in the acceptance criteria are ok:
• The Lookup Engine must return accuracy ratings according to the following:
  - > 50% <= 70%: ‘Likely’
  - > 70% <= 90%: ‘Very likely’
  - > 90%: ‘Almost a Certainty’

If the list of percentages and accuracy ratings had been longer, storing them in a separate spreadsheet on the network makes more sense. There it could be reviewed and updated easily and the acceptance criteria need only reference the source:

• The Lookup Engine must return accuracy ratings according to the rules in the ‘percentage_accuracy-rating_rules.xls’ in the project directory.

There’s one thing missing from the last acceptance criterion. When a supporting artifact is referenced, there should always be a version indicator so when a backlog item is being coded or tested the Developers know exactly which file is the correct one to use. This version indicator should be included in the filename for quick reference (e.g. ‘percentage_accuracy-rating_rules_v6.xls’ and within the file to be doubly sure the right information is being used and mistakes are avoided:

Ideally, supporting artifacts should also be stored along with source code or with the test case repository for future reference of how the project was completed.

Questions & Answers

This last essential component of a backlog item is one that can be added after the Scrum Team has begun to think about it during a planning meeting or during implementation. Even with the most straightforward and complete user stories and acceptance criteria, a Scrum Team will
have some questions about backlog items. These questions and their authoritative answers should be captured within the backlog item as a permanent record. When the answer clarifies an acceptance criterion it is best to update that criterion but it is also fine to each list question and the answer separately. The following questions are examples of what Developers might ask about a specific backlog item after having started implementing the work:

- **Are you sure that you’d like to start with the destination city data element versus the date which may be helpful for lookups?** => Yes, while both will eventually be needed, using the destination city is better for demonstrating to senior management in the early stages of this project.

- **If the Weather Database returns more than 20 matches the Lookup Engine could send too many choices to properly display on the user interface; do you want the amount returned to be capped?** => Yes, have the database results sorted by city and only use the top 20 records.

The list of questions & answers may grow over time but if there seems to be continuous confusion or too many details overlooked, the Product Owner can seek the team’s advice about best way to obtain the business value the backlog item is asking for. The Developers, with their more technical understanding, may be in a better position to explain how to reach the goal.

**Construction of a Product Backlog**

A Product Backlog is the collection of backlog items the Product Owner has created to represent the needs of the business. It is prioritized according to the value each item brings to the business and can frequently be reordered as business needs change.
Although specialized tools exist for creating and maintaining a Product Backlog, using a basic spreadsheet program is quite effective. In addition to the essential information that comprises a backlog item (User Story, Acceptance Criteria, and Questions & Answers), other columns can be used for designating a product release: Priority, Backlog Item Title, Story Points, Sprint and Status. The additional columns are used to organize, prioritize, inform, and keep track of progress during the sprint.

### Release

A Product Backlog could have hundreds of Backlog Items pertaining to multiple products and versions. The Release column provides a label by which to organize and sort a set of backlog items. As a Product Owner defines product releases or adds new backlog items, having a quick way of differentiating it from the entire set is convenient.

### Priority

The Priority column indicates the order in which backlog items should be worked on according to the business value they provide. It exists primarily to sort the backlog items so that come sprint planning time, they are in the correct order. As a Product Owner adds new items, using the Priority value is a quick way to designate the approximate position and can be refined so that the top of the Product Backlog always represents the most valuable work at
any given time. In some tools specifically designed for Scrum development, there is no Priority indicator since Backlog Items can be dragged and dropped into the correct priority position in the list.

**Title**

Writing a short *Title* for each backlog item creates a convenient way to identify the backlog items and the value they describe. Titles should be as short as possible and convey the desired business value. They still cannot include all the details and should not substitute properly written user stories and acceptance criteria. When backlog items represent defects, indicating that within the Title will help the Product Owner track and prioritize defects quickly (e.g. “DEFECT: Report Calculation Error”).

**User Story & Acceptance Criteria**

The *User Story & Acceptance Criteria* columns comprise the core information of the backlog item and really should be kept close together. This is because immediately after reading the user story, the Developers will want to know the specifics of what should be delivered. It is good practice to number each acceptance criteria and have spacing in between them for easier reading since the list could grow long and include many details. The Scrum Team will appreciate the Product Owner’s efforts to make the Product Backlog easy to read.

**Story Points**

After a new Scrum Team has completed a couple of sprints, they should start providing *Story Points* estimates as a way of judging how much
effort is required for each backlog item. Story points can be a valuable measure during Sprint Planning to assess if the team has taken on the right amount of work within the sprint. It can also be used to determine a team’s Velocity, or the amount of work they can do in each sprint and then used to predict how many sprints it will take to complete the set of backlog items comprising a product release.

**Sprint**

The *Sprint* column is for indicating the Sprint the backlog item is associated with. This value would be set during Sprint Planning when a team selects work for the sprint and serves as an indicator of when work was completed. Once a backlog item’s status is Completed or Accepted, the value should not change.

**Status**

The *Status* column will be updated as the backlog item goes from being initially created (New), to ready for selection for a sprint (Ready), to actively being worked on (In Progress), to completed according to the acceptance criteria and the Definition of Done (Completed), to reviewed and accepted by the Product Owner (Accepted). See Appendix C, “Backlog Item States - Flow Diagram” to see when a backlog item’s status changes.

**Questions & Answers / Notes**

The last column is for *Questions & Answers / Notes* and while the primary purpose is to collect questions from the team and the Product Owner’s official responses, it can be used for any notes necessary to aid in
understanding or keeping track of information. When questions and answers are recorded, it is best to write them clearly and follow a consistent format. Numbering each pair and using **bold** text to highlight the question portion will acclimate the reader to the pattern and aid understanding. The Product Owner should look for unanswered questions periodically and provide updates as open issues may be blocking the team’s understanding of the backlog item and willingness to select it for a sprint.

To view the complete Product Backlog example, see Appendix A, “Product Backlog Sample”.

**Sorting the Product Backlog**

Once the Product Backlog has been populated with a number of backlog items, for a product release they should be sorted according to business priority. If the list of items was fairly short, around a couple dozen, this wouldn’t take much effort but in the case where there are many more, a multi-pass sorting technique can make the process efficient.

The multi-pass sorting technique involves making three passes through the unordered Product Backlog with each pass refining a subset of the entire list. By doing multiple passes the sorting exercise is far more manageable than dealing with the entire list. This is by no means the last time the Product Backlog will be sorted as that will be occurring periodically as new backlog items are added and business needs change.
1st Pass

With the first pass, scan the entire Product Backlog looking for backlog items that are high priority, or belong to the first product release, or are associated with some goal or theme the Product Owner wants to focus on. For example, if the Product Owner wanted to demonstrate how clients would interface with the product, look for items that describe business value around designing the request protocol, or the XML file containing the data elements. As these backlog items are found, group them at the top of the Product Backlog without regard for the perfect order.

2nd Pass

In the second pass, focus only on the items brought to the top from the first pass. This should be a much smaller set and easier to put into the final sort order according to business value and the Product Owner’s goals. When comparing any two backlog items that are very different in function, characteristic or area of the product, determining their relative importance can be obvious. For example, designing the client request protocol may be more valuable to have done first because then it could be shared with prospective clients for feedback. In that case, place those backlog items above others that have to do with the XML file for sending the data elements described in the request protocol. After the more obvious choices are done, ordering backlog items where the differences are more subtle becomes challenging and it’s fine to have two backlog items of nearly the same importance next to each other. Referring back to the high level goals will always provide guidance for the best placement.
3rd Pass

In the third pass, review the remainder of the Product Backlog for any missed items or those that would be complimentary to the ones already ordered. Drop them into place in the sorted part of the backlog.

Within the sorted list, the difference between #6 & #7 may be so minor that they could easily flip-flop. There’s little value in getting the order of these cases just right, right now. It is much more important to make sure that #6 truly is more important than #10 or #25 because it so much closer to the top of the Product Backlog and could be selected for the next sprint. Just before each Sprint Planning meeting, the backlog items that could flip-flop may be reordered by the Product Owner as the Developers considers what could potentially be brought into the sprint.

At this point, the top of the Product Backlog is ordered and ready for Sprint Planning. However, depending on the effort level of each backlog item and the capacity of the Scrum Team, it may not be enough to fill a sprint. The Product Owner should continue to order the Product Backlog until there seems to be a sufficient amount of items available for a couple of sprints. How many will be unknown in the early days of working with Scrum but the Product Owner will quickly get a sense of what the team can do.

Defects in the Product Backlog

Most of the time backlog items will describe enhancements to a product but they can also describe defects. There’s a very clear distinction between the two where defects describe something failing to work as originally defined versus additional functionality. It may be the case that a Product Owner will view something missing as a defect but in actuality, it should be an enhancement request because the original requirements weren’t defined well enough in the first place. This may seem like nit-picking but a defect represents a failure on the part of the developers and if the alleged defect really wasn’t their doing then they shouldn’t take the hit for it. If a team repeatedly encounters this problem, where backlog
items aren’t defined well enough and disagreements regarding what is a defect ensue, it’s a sign that backlog items and their associated acceptance criteria aren’t complete enough.

Writing Defects

When defects are included in the Product Backlog as backlog items, their format can be different from items requesting enhancements. In place of the user story, a clear statement of what is failing is needed. These can be written in a technical manner such as:

The BillingTransaction table’s ReturnCode field isn’t being updated with code 902 for failed debit card transactions.

Or they can be translated into something that indicates the effect on business value:

When debit card transactions are failing, the database isn’t being updated with the proper codes.

In the later example, the meaning is clear to the Product Owner and provides some clue as to what the implications are which in turn makes prioritization easier. In this case, the reports generated from the database will be missing debit card transaction failures and that could have an impact on business decisions. When written in the style indicating the impact on business value, the details within the defect should include the former and more technical explanation to aid the Developers in fixing the problem. The defect details should always include:

**Problem Description** - A concise description of the problem that will aid the Developers in isolating the issue:

The BillingTransaction table, ReturnCode field, isn’t being updated with code 902 for debit card transactions failures. This will result in transaction reports missing records.

**Replication Steps** - The exact steps necessary for someone else to experience the problem including any information needed to set up the test:
Expected Results - A concise description of the expected results:

The BillingTransaction table, ReturnCode field should have a transaction code of 902 for the purchase attempt that failed.

Actual Results - Another concise description, this time of the actual results:

The BillingTransaction table, ReturnCode field did not have a transaction code of 902 for the purchase attempt that failed.

It may seem like a lot of details with redundant information but this level of detail provides the Developers with a clear description of the problem, how to replicate it, and what the failure was. Clarity is especially important when when working with resources in other locations or drastically different timezones where picking up a phone to ask questions may not be an option.

Defects are further enhanced by including screenshots with markings to indicate problems and expected results where appropriate. The example above may not benefit from a screenshot but a defect where pictures misaligned in a web page would. See Appendix A, “Backlog Item (Defect) - Sample” for a complete version of Backlog Item describing a Defect.

Converting Existing BRD Requirements into Backlog Items

At the time of an organization’s transformation to Scrum, there will likely be product ideas defined in Business Requirement Documents. These documents can be repurposed into backlog items and collected into a Product Backlog. Much of the same information required is already defined and all that’s needed is to repurpose the information in terms of business value.

This may be challenging for the Product Manager, now the Product Owner, when it was they themselves who wrote the original Business Requirements Document. They will have to back away from all the
details already defined and see the product holistically almost from an outsider’s point of view. Only then can they jump back in and slice up the product into discrete units of business value.

Using the *Weatherly* product again as an example, this product was originally conceived of and documented when the organization was using Waterfall to develop products. A full Business Requirements Document (BRD) was written and includes all the details the Product Manager thought of at the time.

The first step in the conversion is to review any material that describes the project including background, history, business analysis, goals and high level requirements. This will provide the basic understanding of the value the product provides and set the stage for identifying slices of specific business value that can be described as backlog items.

In the case of *Weatherly*, the main benefit is to provide weather forecasts for travel destinations stored in a user’s calendar. Based on this the business value described in backlog items should be focused on delivering pieces of that functionality to them.

Converting a Business Requirements Document (BRD) is relatively straightforward once you start looking for ways to group the specific requirements into units of business value. This may require pulling pieces from various sections to create the definition of something that fulfills a business need.

Using a section of the *Weatherly* BRD regarding “Logging & Reporting” the very top of the section indirectly indicates the business value regarding reporting:

```
Logging & Reporting
```

When users use the product, a record of their transactions will be kept for technical support analysis, business reporting, and billing purposes.

It is specifying that logging will be needed for a series of purposes in a dry factual kind of way. To convert this into units of business value it needs some context of who will use it and why. At least three User Stories can be derived from the above section:
• As a Technical Support Technician, I want all app transactions logged so that I can troubleshoot issues regarding usage of the product.
• As the VP of Product Management, I want all app transactions logged so that I can run reports on activity and be informed about how users are using the product.
• As the Accounting department, we want all app transactions logged so that we can create summary and detailed revenue reports.

The reader will notice that all three roles want basically the same thing, *all app transactions logged*, for some purpose tailored to their needs. The acceptance criteria for each backlog item would then detail the specific requirements needed in order to satisfy those needs. Taking the first user story above regarding the Technical Support Technician it could have the following acceptance criteria:

- The following data elements must be logged for app requests: product-code, request-type, request-ID, time-stamp, username, destination-city, travel-dates
- The following data elements must be logged for app responses: request-ID, time-stamp, response-value, error-code
- The database schema as defined in Appendix F, “Weather Database Schema, version 3.5” must be used.

Each of the above acceptance criteria were derived from requirement directive statements found in the BRD:

<table>
<thead>
<tr>
<th>ID:</th>
<th>Requirement Directive Statements:</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2 2</td>
<td>The project shall log the following data elements in the Transaction Logging database for <strong>app requests</strong>:</td>
</tr>
<tr>
<td></td>
<td>• product-code, request-type, request-ID, time-stamp, username, destination-city, travel-dates</td>
</tr>
<tr>
<td>R2 3</td>
<td>The project shall log the following data elements in the Transaction Logging database for <strong>app responses</strong>:</td>
</tr>
<tr>
<td></td>
<td>• request-ID, time-stamp, response-value, error-code</td>
</tr>
</tbody>
</table>
Chapter 3 - Business Requirements under Scrum

The second user story regarding the Account Manager who wants to keep tabs on client usage of the product might include the following in the acceptance criteria:

- The report must use the format as specified in Appendix G, “Weekly Transaction Report Sample, version 3”.

This too was derived from a requirement directive statement in the BRD:

<table>
<thead>
<tr>
<th>ID:</th>
<th>Requirement Directive Statements:</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2 4</td>
<td>The product shall use the database schema as defined in appendix F, “Master Database Schema, version 3.5”.</td>
</tr>
</tbody>
</table>

As must be obvious through these examples, the process of defining backlog items isn’t a matter of copying content from the BRD to the Product Backlog. The challenge is teasing out the business value from the lists of requirements and assembling backlog items that provide the right context and specifics necessary to judge them as done. The process of identifying business value continues until the Product Owner feels they have captured all the original BRD material as backlog items. A complete example of the BRD section dissected in these examples is available in Appendix A, “Weatherly - Business Requirements Document - Sample”.
Now that a Scrum Team has been identified, had some classroom instruction on Scrum development and the Product Backlog is ready for their use, the team can start working under the new methodology. Since working under Scrum is new to the organization, a Sprint Zero is warranted. Arguably any activities in a Sprint Zero could be done as part of the first true sprint (Sprint 1) but giving the team a little time to get acclimated to the new process can be beneficial. Typically a Sprint Zero will be half as long as the normal sprint length but that depends on what is to be accomplished. For this text, it is assumed that standard sprints will be two weeks in length and therefore Sprint Zero only one week.

There are very specific objectives for Sprint Zero and they focus on getting the Scrum Team ready to operate under the new methodology. This involves not only familiarizing the team with new processes but identifying what they need to work and getting them familiar with the products they will be focusing on.

This is a critical time for the team to understand and get on board and it is important that they be given the best possible explanations and opportunities to ask questions. Some will adapt quickly to the change and others won’t; great care must be taken to get everyone up to speed.

Team Kickoff Meeting

The purpose of any kickoff meeting is to bring team members together to be introduced to the purpose of their forming and the high level goals of the effort. In the case of the Sprint Zero kickoff meeting, they are also getting a first look at how they will operate under the new methodology.

The following sections explain the essential topics that an effective Sprint Zero kickoff meeting should have. Of course more can be added depending on the needs of the organization or project but what is represented here are the fundamentals that should be covered in order to start the process off right.
Welcome Message

Any meeting should start with a proper welcome message, explanation of why it has been organized and the expected outcome. This meeting is no exception. Assuming the newly assembled team is more or less familiar with each other and may have some experience with Scrum software development, it’s still a good idea to level set expectations.

Transformation Goal

Some team members will be excited to move to Scrum, others will have plenty of questions and perhaps uneasiness about the whole process. It should be expected that they’re wondering why this transformation is being undertaken and why they personally are involved.

Address their questions directly by explaining the reasons identified by the management team and what they hope to gain by switching methodologies. Cite statistics about the current process and those of how switching to Scrum provided specific benefits for other organizations.

As for the makeup of the team itself, similarly explain how the team was chosen and why. Scrum development emphasizes transparency so this is the best time to make it clear how the organization arrived at the decisions they made.

Next, explain that the transformation will require a new way of thinking about defining products, executing the work, tracking progress, and keeping the business informed. The transformation will not only include the immediate Scrum Team but many aspects of the organization as changing Software Engineering will have ripple affects across the entire system.

Ritual, Processes, and Artifacts Introduction

Immediately after explaining the transformation goal is the optimal time to introduce the team to some of the key rituals, processes, and artifacts. Going through each one in detail is not the goal at this point and could consume hours. The objective is to bridge the gap in their understanding of the existing Waterfall methodology to the new Scrum methodology. This can be accomplished with a simple table showing the old, the new, and the purpose of the new one. It doesn’t have to be a complete list but should include items from the old methodology that are most
recognizable to give everyone a sense of how things will change. For example:

<table>
<thead>
<tr>
<th>Waterfall:</th>
<th>Scrum:</th>
<th>Purpose:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekly Status</td>
<td>Daily Standup</td>
<td>For the Scrum Team to keep each other aware of progress and address issues.</td>
</tr>
<tr>
<td>Meetings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execution Planning</td>
<td>Sprint Planning</td>
<td>For defining the work within the next development sprint.</td>
</tr>
<tr>
<td>Project Delivery</td>
<td>Sprint Reviews</td>
<td>For demonstrating work to the Product Owner.</td>
</tr>
<tr>
<td>Lessons Learned</td>
<td>Sprint Retrospective</td>
<td>For assessing how the team did in the last development sprint and identifying necessary changes.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Challenges and Guidance**

It should be made clear to the team that no one expects everything to go smoothly and that there will likely be serious challenges. The key to success is to be open and honest about what the problems are and try to find solutions. If influential Senior Management is behind the initiative, make it clear they too expect issues to arise but that they have confidence that the team will resolve them. The transformation will be disruptive not only for the immediate Scrum Team but for other parts of the organization as well. Incremental improvements will be the norm as everyone gets used to the new methodology.

**Continual Inspection and Adaptation**

Built into the Scrum methodology is the process of inspection and adaptation where Scrum Teams conduct Sprint Retrospective meetings. Besides this however, improvements can be identified and implemented at any time and will need to be, lest they become impediments to success. When blockages to progress are encountered, ad hoc meetings will occur to characterize the problem, list potential solutions, and select the best that makes the most sense at the time. It will often be better to keep moving forward with a solution that seems the best choice versus taking too long to analyze and define the perfect solution. Losing momentum in the early stages of the transformation can cause it to halt all together as people will naturally want to return to the methods to which they were accustomed to.
Chapter 4 - Initiating Scrum Development (Sprint Zero)

Sprint Zero

Now that the team has an understanding of the adventure in which they will participate in, they should be introduced to the specific objectives of Sprint Zero. At a minimum this will include:

- Participating in a **Product Overview** provided by the Product Owner to help them understand the big picture and context of why they will be working on the first project.
- Creating the **Definition of Done** to guide the Scrum Team in knowing when they have delivered business value each sprint.
- Identifying **Tools, Processes, Environments and Access** they’ll need to operate within the new methodology.
- Attending the first **Story Time Meeting** where they’ll be introduced to the Product Backlog and start to understand how the Product Owner has defined slices of business value for them to deliver.

**Audience Questions**

This will have been a lot of information for most people so opening it up for questions is an important part of the meeting. No doubt there will be process questions and if they can be answered quickly they should, however for more in depth topics, separate and detailed discussions around the mechanics of how things will work can be scheduled for later. Many of these can be addressed when the first time a new ritual, process, or artifact is presented. If too much is explained at one time, most people won’t retain it, leading to confusion, frustration and resistance. It’s better to give high-level explanations now and conduct process training just in time before it’s needed.

**Definition of Done**

The Scrum Team’s goal each sprint is to deliver work that is “potentially shippable”, that is, business value that could be deployed to production or otherwise delivered to customers if the Product Owner so chooses. The Definition of Done establishes the standard by which they can judge if they have completed their work to meet the goal.
Ideally the Definition of Done would be created by the Scrum Team themselves so they can understand its meaning fully. Less ideal is that management defines it for them and they simply have to obey. The optimal solution for any given organization may be some hybrid of the two approaches where the team and the functional managers collaborate on the definition. Defining a set of principals may be an unnatural exercise for most people so starting off with a rough idea may help start the discussion.

The structure of the Definition of Done has no standard but it should clearly define the accepted practice of delivering quality and complete software to customer. For example, backlog items in the current sprint are Done when:

- All acceptance criteria associated with a backlog item are met.
- All defects associated with backlog items have been resolved or deferred by the Product Owner.
- All coding and testing standards have been followed.
- All documentation associated with coding or testing activities has been written.
- The backlog item is ready to be demonstrated to the Product Owner at the upcoming Sprint Review meeting.

And when the Product Owner intends to ship the software at the end of the sprint the following are needed:

- All product support handoff (Technical Support, Customer Service, etc) meetings / processes have been completed.
- All deployment handoff (Operations) meetings / processes have been completed.

At the end of a sprint, if a backlog item is not considered Done, then the team does not get credit for it during that sprint and is not demonstrated to the Product Owner. Unfinished backlog items can be finished off in the next sprint assuming the Product Owner prioritizes the work high enough to be selected during Sprint Planning.

Starting with a decent Definition of Done is critical to the Scrum methodology and it can be refined over time with each Sprint Retrospective meeting where the team reviews the events and
performance within the sprint. It should be stored in a location easily accessible by the team when needed for reference. See Appendix A, “Definition of Done - Sample” for a complete example.

Establishing Knowledge, Tools, Processes, Environments, Access and Meetings

Every organization is different in terms of the changes necessary to support the transformation to Scrum. The most fundamental change common to many will be the acceleration of the development cycle of design, coding, testing and deployment. This one change can be very disruptive but there are other considerations as well. Since no two organizations are the same, the following section is designed to prompt thinking about what might need to change. The evaluation of necessary changes will need to be done by more than just the Scrum Team.

Once the Scrum Team has been introduced to the Product Line Roadmap and initial product versions to be worked, this should help frame up what will be needed day-to-day to execute the work. The more they can identify and set up in Sprint Zero the more actual business value they can deliver in Sprint 1.

Knowledge

- **Did all members on the Scrum Team attend one of the Scrum training sessions?** If not, they may need a brief lesson on the basics of Scrum and a high level introduction to the processes so they’re not trying to catch up in Sprint 1.

- **Are there aspects of the first product (codebase, data, testing methods, etc) with which the team members aren’t familiar?** If yes, schedule training meetings with subject matter experts and invite everyone from the Scrum Team to attend so they can all hear the same information.

Tools

- **Will testing the product require new test data?** Testing will be part of every sprint and obtaining a standard set of test data before Sprint 1 will save creation time during sprints.

- **Is there an automated testing capability?** Again, because testing work will be included in every sprint, the more testing that can be automatically run and reduce manual effort the better.
• Will the team need new tools to build the software? The product could need different integrated developer environments or compilation tools to build applications.

• Will the current methods of communication and coordination work in an accelerated environment? Working under Scrum will accelerate the design, development, testing, and deployment of code so the old methods of communicating status or intentions may need to be upgraded for the faster pace.

Processes
• Will the first product require a new build process or modifications to the existing one? If so, the entire team should be trained on the new procedures.

• Will the new product require input or deliverables from outside Software Engineering? If so, establish protocols for working with external resources and integrating with their process to ensure they don’t become impediments.

Environments
• Will the first product require new code branches? A special code branch may be needed to support working under Scrum.

• Are testing environments readily available? In Waterfall development testing comes much later in the process but under Scrum the full suite of environments, data, and tools must be available within sprint 1.

Access
• Does every team member have access to all the necessary network drives, code repositories, and tools? The Scrum Team may have been assembled from various groups and therefore not already set up with the right access to everything they need to complete sprint work.

• Can the Scrum Team access the Product Backlog and Product Owner generated artifacts? The Scrum Team will need to view the Product Backlog and artifacts during Sprint Planning and Story Time meetings. They will also need to occasionally view these materials to make sure they are building according to the user stories, acceptance criteria, and referenced files.
Chapter 4 - Initiating Scrum Development (Sprint Zero)

Meetings

- **Have reoccurring meetings for the standard rituals been set up?**
  This includes the Daily Standup, Sprint Planning, Sprint Review, Sprint Retrospective, and Story Time meetings. Having these meetings at the same time and location helps establish a pattern and sets up consistency. Have Scrum Team members move other meetings to accommodate the optimal times. See Appendix B for sample meeting agendas.

In general the more preparation work that can be accomplished during Sprint Zero the better. Not everything will be covered by the beginning of Sprint 1 so it should be expected that some time will have to be allocated to finishing up.

**The First Story Time Meeting**

Near the end of Sprint Zero, after the Product Owner has shared the Product Line Roadmap and explained more about the first product to be worked on, the Scrum Team should have their first **Story Time Meeting**. The name of this meeting is admittedly a bit silly but it conveys the purpose perfectly. It is an opportunity for the Scrum Team to look at the top backlog items in the Product Backlog and begin to understand how the Product Owner has defined the discrete units of business value.

This meeting is not another presentation from the Product Owner to the team, but rather a working session where the entire Scrum Team, Developers, Scrum Master, and Product Owner engage in conversation about what each backlog item is asking for. This is where the term **Story Time** comes in as the Product Owner tells the story of what they’re looking for and how they sliced up the overall product into units of business value. If the backlog items are written well, they already include user stories, acceptance criteria, and references to necessary artifacts but the Product Owner can expand on each and answer any questions the team has.

During Story Time Meetings, the Developers...

- **seek to understand what the Product Owner is looking for.**
  This should include a review of the backlog item as written and any context or clarifying explanations from the Product Owner. Questions whose answers might be useful for implementation
should be recorded for future reference. Questions that indicate missing information or poor wording should be used by the Product Owner to update the backlog item.

- **make suggestions regarding backlog item objectives.** The team will likely be more familiar with the technology, processes and limitations associated with the backlog item and can provide feedback on how to restructure the backlog item, or split and create new ones to achieve the business value desired.

- **inform the Product Owner about dependencies or technical constraints that might influence the priority of the backlog item.** The Product Backlog will be sorted according to business priority but it may make more sense to rearrange the order slightly to ensure prerequisite work is completed first.

- **provide relative effort estimates for each backlog item.** When the team has been trained on relative sizing techniques, they should assign effort estimates to each backlog item. Two common forms of this are Story Points or T-Shirt Sizes. (How to train the Scrum Team on relative effort estimates with T-Shirt Sizes will be covered in Chapter 5.)

Participating in the first Story Time Meeting prior to Sprint 1 Planning will be very helpful since during the planning meeting the team will also be learning about the planning process and having a decent understanding of the backlog items ahead of time will make the meeting go smoother. See Appendix B for a “Story Time Meeting Agenda - Sample”.
Scrum Development Continues (Sprint 1 to n)

With Sprint Zero over and the team as ready as they’ll ever be, the first full development sprint can begin. In Sprint Zero much activity should have revolved around readying the team in terms of how they’ll operate under the new methodology but not everything planned may have been setup or occurred. Any remaining effort can continue in Sprint 1 with an understanding that the capacity for producing business value will be diminished. This is the overhead of starting a new team and will be less of an issue in subsequent sprints.

A key concept in Scrum development is the idea of establishing a rhythm of activity at a sustainable pace within each sprint. Some rituals may seem nonessential or burdensome but they all serve a purpose and over time fade into the background as established process for the delivery of business value.

The Scrum Team may find the transformation troublesome with all the new terminology, processes, and expectations but the single most challenging part of switching is the mental change and tendency for people to return to what they knew and may have practiced for years. If everyone keeps an open mind and is willing to learn and adapt, the transition can progress quickly.

Sprint Planning

The purpose of Sprint Planning is to accept work into the sprint that the Scrum Team feels they can commit to completing by the end of the sprint. This concept is important because the team is expected to deliver what they’ve chosen so they must have the ultimate say in what will be selected for the sprint.

The ideal environment for a Sprint Planning meeting is a conference room with whiteboard, markers, a projector and computer to display the Product Backlog and Sprint Backlog. The Scrum Master and Product Owner should sit at the end nearest the screen not because they are “the authorities”, which they are not, but because it will be easier for the
Developers to pay attention to the screen and engage with these two people if they're all in the same general location.

Meetings Preparation and Duration

The Product Owner should come to the meeting with the latest Product Backlog groomed and ready for sharing. Having the backlog items in the desired priority order is required and eliminating any extraneous information or distractions is recommended. Likewise, the Scrum Master should come to the meeting with the Sprint Backlog file prepared and ready to update with backlog items selected for the sprint. The Sprint Planning meeting is often time-boxed, that is, allocated a fixed amount of time to accomplish the objectives. The Scrum Master should be the timekeeper to make sure the team is making progress towards the goal of selecting the right amount of work into the sprint. The duration of a Sprint Planning meeting will depend on how long sprints are and what the team will need to accomplish the work. With a new team, the right amount of time may not be known but here are some general guidelines:

<table>
<thead>
<tr>
<th>Sprint Cycle:</th>
<th>Meeting Duration:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 week</td>
<td>1 hour</td>
</tr>
<tr>
<td>2 weeks</td>
<td>2 to 4 hours</td>
</tr>
<tr>
<td>3 weeks</td>
<td>3 to 6 hours</td>
</tr>
<tr>
<td>4 weeks</td>
<td>4 to 8 hours</td>
</tr>
</tbody>
</table>

These recommendations are based on the idea that the team accepts backlog items and breaks them down into tasks with owners and initial hours estimates. For a more experienced team, they may want to simply select the backlog items with the Product Owner and Scrum Master present and then have an in depth design meeting later to identify all tasks, owners, and initial hours estimates.

Sprint Capacity Assessment

When a team accepts work into a sprint, they are in fact committing to completing it so an important first step is to assess their capacity for sprint work. With any job there are the core functions a person performs most of the time and other various activities they are involved with throughout the workday. The sprint capacity assessment isn’t meant to detail every minute of each sprint day but to give the team a sense of how
available they’ll be for sprint work. After a number of sprints have passed, knowing how much work can be taken on becomes more based on a “gut feeling” rather than any calculations or measurements. The best way to discuss and visualize a team’s capacity is to show a simple matrix of team members by sprint days and to record the time they have available. The following chart is a typical example.

<table>
<thead>
<tr>
<th></th>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
<th>Day 6</th>
<th>Day 7</th>
<th>Day 8</th>
<th>Day 9</th>
<th>Day 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Louis P.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edwin H.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charles D.</td>
<td></td>
<td></td>
<td>PTO</td>
<td>PTO</td>
<td>PTO</td>
<td>PTO</td>
<td>PTO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carl S.</td>
<td></td>
<td>50%</td>
<td>50%</td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Isaac N.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Albert E. (PO)</td>
<td>PTO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marie C. (SM)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JRY</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Scrum Master can update this matrix ahead of time for any known dates and then ask the team during the meeting for any new information. Where a cell is left blank that should be assume to be totally available for the team minus the usual distractions or other non-sprint work activities. In the sample above Charles the Developer will be on vacation in the middle of the sprint, Carl the Developer will be on a special assignment not related to the team’s project, and both Albert the Product Owner and Marie the Scrum Master will miss a day for different reasons. The cells can contain whatever is needed so long as the team understands how their capacity for work will be affected. At the end of this assessment, everyone should be well aware of who is available, for how long, and how they can contribute to the delivery of sprint work.

Previous Sprint Lessons Learned

At the end of every sprint the team will participate in a Sprint Retrospective meeting to assess sprint execution and identify ways to improve for the next one. The details of meeting will be covered more later in this chapter. If Sprint Planning was occurring for Sprint 2 or later, the Scrum Team should review the key learnings from the retrospective meeting. Often many topics are discussed but there can be a few themes or salient points that are worth reviewing at the beginning of the current sprint. For example:
The team should not underestimate the effort involved for backlog items that will require unfamiliar or knowingly difficult technologies.

Backlog items dealing with user interfaces should be demonstrated to the Product Owner as early as possible in case there are cosmetic adjustments needed.

When backlog items are dependent on an external resource, the Scrum Master should invite the necessary people to join the Daily Standup meetings so they can be immediately available for questions.

It is best to show these on the projected screen with the Scrum Master talking through each one, providing emphasis and context where needed. By highlighting a few important learnings at the beginning of the sprint, it reinforces the change the team thought necessary and improves the chances of the advice being followed.

**Product Objectives Review**

By now the Scrum Team has seen the Product Line Roadmap, been given a tour of the first product’s main features and functionality, and talked about the top backlog items in a Story Time meeting. It helps to review what the short term objectives are one more time. Providing context, repeatedly, will help the team tremendously when accepting work into the sprint and they begin to plan work execution.

The Product Owner should display a list of the product’s business objectives and highlight any key features that contribute to it. After this the team is primed and ready to start the process of selecting sprint work.

**Selecting Sprint Work**

The process of selecting work into the sprint is straightforward and starts with the copying of the first backlog item from the top of the Product Backlog into the Sprint Backlog.
Then the Developers define tasks necessary to complete the backlog item, typically in the order in which they should be done but not always so. The task list should include all need work such as researching, training, designing, coding, testing, documenting, etc. The granularity of tasks is completely up to the Developers who may want a detailed list to track their own progress or more high level activities that they already know have many steps. How granular depends entirely on the individual but the right level tends to be what one might answer to the question of “What did you work on today?” Even if a hundred things were accomplished they wouldn’t be included in the answer, but instead a quick summary grouping similar parts of the effort would be given.

In the above example, the task list is short only to provide a general idea of format. In reality the list could be a dozen rows long for each backlog item and only include work necessary to complete the backlog item. The Scrum Master should guide the team in identifying tasks across disciplines (coding, testing, database administration) as well as any general needs such as infrastructure or coordination with other groups. When the frequency of new tasks identified dies down, the Scrum Master can call that one done and move on. To belabor the exercise will only build frustration and lose momentum. During sprint work execution, the team may, and often do, add new tasks as they are discovered.

Each task also needs to be assigned an Owner and Initial Hours Estimate. This can be done immediately after the one task is written, after all tasks for a backlog item have been identified, or when all backlog items and
tasks are logged. The preference is up to the team and by waiting to do them in small groups or all at once may be helpful in balancing out the work across the team.

The Owner is the person likely to be completing the work or responsible for seeing that it is done. Ownership of a particular task can change during the course of a sprint as the team rebalances work.

The Initial Hours Estimate can be a tricky value to provide. When asked for an estimate, most people, and this is especially true of software engineers, will try to be very accurate. Accuracy is unnecessary because this value is only used to gauge progress through the sprint and each day during the Daily Standup meeting, the owners of tasks will be asked for updates on the Remaining Hours to complete the work.

This new daily value can decrease, increase or stay the same from day to day and how it is used will be explained in more detail in the section about the Daily Standup meeting and Sprint Burndown Charts. If a Developer is getting stuck on providing an Initial Hours Estimate the Scrum Master should encourage a quick answer and offer that they can provide an update once they know more. This usually frees the Developer from the burden of providing a well thought out, accurate answer.

After each backlog item is decomposed into tasks, the Scrum Master should ask the team if they are formally selecting this backlog item into the sprint and look for team consensus. Then they should be asked if they have reached their capacity for the sprint or if they can take on more. It may not be necessary to ask this second question after only the first couple of backlog items based on their complexity but after a few more are included it should always be done. The assessment of capacity is very much a “gut feel” kind of measurement and may be wrong the first few sprints but as a team gains experience they’ll get better at knowing what they can take on and how busy each member will be.
Once the team has reached the point where they believe they cannot add any more backlog items the selection process is done. They should never be pushed into accepting more work than they think they can commit to completing.

**Sprint Execution Strategy**

Near the end of a Sprint Planning meeting where the team has selected what seems to be the right amount of work into the sprint, they should now define a Sprint Execution Strategy. This exercise is lightweight yet incredibly valuable for coordinating between team members.

This is where having a whiteboard and markers in the meeting comes into play. The Scrum Master should draw a simple timeline of the sprint on the board like so:

<table>
<thead>
<tr>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Mon</th>
<th>Tue</th>
</tr>
</thead>
</table>

Then, he should hand a couple of markers to team members, preferably at least one Coder and one Tester and ask them to map out when backlog item work would start, end, and any key events that affect the whole team like code merges or producing builds for testing. If the team usually defers to the Scrum Master for guidance and control, he should excuse himself from the room briefly to allow them to get started. A quick run to the restroom usually works well.

The team, comprised of experienced problem solvers, will easily map out a plan and the coordination between parties. It is typically a lively discussion with anyone able to grab a marker and participate. The end result will be a handful of events tied to sprint days and a satisfied look on the team’s faces.
Once the team is done the Scrum Master should capture the details and prepare a more formal version as a presentation slide and bring that to each Daily Standup meeting for reference. This technique of having the team map out their plan really helps with coordination and timing and will have them functioning more as a cohesive unit rather than independent roles (Programmer, Quality Assurance, Database Administrator, etc). Now that the sprint has been planned, the team can jump into executing the work and participate in the Daily Standup meetings which start up the next day.

**Daily Standup Meetings**

The Daily Standup meeting is the most frequently held meeting. It occurs every workday, except for the day of Sprint Planning, preferably at the same time and location with all members of the Scrum Team. The required participants of this meeting include the Developers, Product Owner, and Scrum Master. Other attendees outside of the team are optional and encouraged to observe, but not contribute during the first part of the meeting.

The purpose of the Daily Standup meeting is very clear, to allow the team to share progress, identify impediments, and obtain clarification or guidance from the Product Owner who represents the business in terms of what should be developed. The Scrum Master’s role during this meeting is to keep the team following the established process and learn about impediments they need to help clear.

In terms of scheduling meetings, it’s a good idea to block off 30 minutes of the team’s time, preferably first thing in the morning. The first 15 minutes should be reserved for each Developer to answer three questions:

1) What did you accomplish since the last Daily Standup meeting?
2) What will you work on before the next Daily Standup meeting?
3) Do you have any impediments or potential impediments that will block your progress?

This can start spontaneously or at the nudging of the Scrum Master and as each team member is done, the next one starts up. It is important to remember that this is not a status meeting where updates are provided to the Scrum Master or Product Owner but an information sharing meeting for the entire team. As an example, a Developer might say:
1) I finished coding the authentication function within the App Gateway so it can validate incoming service requests.

2) I’m going to provide Edwin with a demonstration of the authentication function and begin coding the Lookup Engine this afternoon.

3) I’m waiting on Helpdesk to give me permissions to the staging server and it has been two days already with no response.

When a team member has no impediments, they can simply say “No impediments.” and that should signal the next person to start up. When it’s difficult to remember what they want to say reading from notes is acceptable as long as they’re engaging with the group. Usually once a team is more experienced and in tune with each other’s work, answers to the three questions are more spontaneous and easily understood by everyone thus requiring less preparation and formal delivery.

There will be times when a team member wasn’t able to work on tasks associated with the project. In those cases it is perfectly acceptable to state that they didn’t accomplish any sprint work or won’t be able to focus on the project that day. This lack of progress is important information to the Scrum Master who will be sensitive to team members’ attention going elsewhere and impacting sprint work. They should investigate and/or mitigate whatever is causing the team member to be allocated elsewhere.

Those team members reporting nothing should not be ashamed of doing so when it is no fault of their own. If on the other hand, a team member isn’t making the progress they ought to be, it becomes clear to the team and the Scrum Master that something might be going wrong. The Scrum Master can investigate on the spot or discretely at another time but often being forced to repeatedly show no progress causes the team member to self-correct and get back on track.

As team members give updates, there will be questions or commentary among the team. This should not be stifled so long as it doesn’t interrupt the flow of the meeting. If a discussion topic seems to require in depth discussion or the conversation goes too long, the Scrum Master should ask to hold off until after they complete the ritual. Sometimes observers can contribute by answering a question or resolving an impediment and in that case they can jump in but again the Scrum Master should urge focus back to the ritual if the meeting is taking off in another direction.
After all team members have provided their updates the first part of the meeting is over and if there were no other issues to discuss the team disbands. If further discussion was needed, the team stays for the remaining allocated time. Once a team gets used to the flow of providing updates, the Daily Standup will often be less than 15 minutes.

New teams, even those familiar with Scrum, will take a couple of sprints to get used to providing updates and learning the natural flow of work between team members. The Scrum Master should coach whenever possible to help build a cohesive and collaborative team.

**Sprint Burndown Charts**

During Sprint Planning, the team created a Sprint Backlog file consisting of the backlog items accepted into the sprint and their associated tasks, owners, and initial hours estimate.

The Daily Standup meeting provides the perfect opportunity to give updates on the \textit{Remaining Hours} for each of the owned tasks.

These daily estimates represent what the task owner thinks they have left to complete the task. If the Developers find it challenging to provide an estimate, remind them that it only represents what they think based on what they know now and will be re-estimated at the next Daily Standup meeting. It’s perfectly acceptable that the value can go up, down, or remain the same. There should be no judgement regarding the number by the Scrum Master or others on the team but reaching zero can certainly be appreciated.
Chapter 5 - Scrum Development Continues (Sprint 1 to n)

There’s more than one way for Remaining Hours updates to be obtained; they could be done online before or after the meeting, told verbally during the meeting, or updated on a sprint backlog printout. This last technique is efficient and provides some visibility into what each team member has completed. To support this, the Scrum Master should lay printouts across a nearby table with pencils handy and the team members will find their owned tasks and make their updates.

The end result of capturing the team’s estimates is the Sprint Burndown Chart which provides a visual representation of the team’s progress. It is not meant to be a precise accounting of hours worked or hours left to do but gives the team some sense of where they’re at each day in the sprint. Following is an example of a sprint burndown chart where the team is about two thirds through the sprint.

Each day after the Daily Standup meeting, the Scrum Master updates the chart and either sends it via email or makes it available online for viewing. See Appendix A, “Sprint Daily Email Update - Sample” to see a typical message to the team.

Included in the chart is the Ideal Burndown Line (straight dotted line) which provides a visual guide starting from the highest point on the chart, typically the leftmost point, to zero on the last day of the sprint. This guideline gives the team a reference for where they ought to be in terms of work completed by each day in the sprint if they want to reach zero task hours. The typical team’s burndown line derived from Remaining Hours updates will generally bounce above and below the guideline from day to day but so long as it’s heading towards zero the team is making good progress. Chapter 6, “Inspection & Adaptation” will cover the Sprint Burndown Chart in more detail describing trending patterns and what action might be necessary for various conditions.
Sprint Review Meetings

The formal purpose of a Sprint Review meeting is to demonstrate completed backlog items to the Product Owner and gain their acceptance of the work. Informally it’s a time for the Developers to show off their work and contribution to the product. This is especially valuable when organization stakeholders attend the meeting and get to see not only the end result but some sense of the challenges overcome by the team and the professionalism they bring to building a product.

The Sprint Review meeting should ideally be scheduled for the latest part of the sprint which usually means the afternoon of the last sprint day. The duration of the meeting will depend on how long the sprint cycle is and how much work is to be demonstrated but it’s good practice to establish a reoccurring meeting with a fixed time and have the team get used to staying within the limits. Adjustments to the duration can be made after a few Sprint Review meetings have been held and a better timeframe is known.

The essential attendees of the Sprint Review are the Scrum Team members themselves. Others can, and frankly should, be invited so the team’s progress can be shown and feedback solicited. The meeting’s primary purpose however is for the Developers to demonstrate completed work to the Product Owner and the Scrum Master should be sure to keep the meeting on track and limit observer participation when needed.

Sprint Review Preparation

During a sprint, the Developers can be quite busy so it’s helpful for the Scrum Master to take on administrative work and minimize distractions for the team.

At the simplest, a Sprint Review can be nothing more than a demonstration of the work to the Product Owner without any special introductions or ceremonies. More established teams typically end up reducing the process to the essential elements once everyone is used to the practice. However, for newer teams it’s a good idea to include a little ceremony especially when observers not normally associated with the process are in attendance. An easy method to having some formality and ceremony is to create a presentation that explains and guides the process.
through the Sprint Review and serves as a historical artifact of the accomplishments during that sprint.

On the last day of the sprint, at the end of the Daily Standup meeting, the Scrum Master should ask the team for updates on:

- What backlog items they believe will be completed by the Sprint Review meeting.
- Who will demonstrate which backlog items.
- What the challenges were during the sprint.
- What factors contributed to not reaching zero task hours as illustrated by the Sprint Burndown Chart.

Once the Scrum Master has this information, he can build up the Sprint Review presentation slides while the team readies for the meeting later in the day. Following are a list of the essential slide topics to be included in the presentation. Additional slides and content can certainly be added based on what might be demonstrated or the organization’s needs.

*Title Slide* - Since this presentation file will serve as a historical record of the team’s performance the title slide should include the team name, sprint number and beginning/ending dates for the sprint. If the sprint work is focused solely on one product, including that makes sense as well.

![Apptastic, Inc]

<table>
<thead>
<tr>
<th>Team One</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint 9 Review</td>
</tr>
<tr>
<td>March 19-30, 2012</td>
</tr>
<tr>
<td>Weatherly iOS v1.0 Product</td>
</tr>
</tbody>
</table>

*Product Summary & Goals Slide* - This slide is primarily for any observer not familiar with the product and serves as a high level summary of the product and goals sought.
Switching to Scrum - William Patrick Swisher

Product Summary & Goals

Weatherly v1.0 - Providing users with up to date weather forecasts for travel destinations found in their calendar.

- Automatically identifies travel destinations on the user’s calendar.
- Provides current and forecasted weather conditions for the destination.
- ...

Sprint Resources & Capacity Slide - This slide represents a summary of the team resources and their capacity during the sprint. It is the same matrix shown during Sprint Planning and serves to convey just how much time the team had to work with.

Sprint Resources & Capacity

<table>
<thead>
<tr>
<th></th>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
<th>Day 6</th>
<th>Day 7</th>
<th>Day 8</th>
<th>Day 9</th>
<th>Day 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Louis P.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edwin H.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charles D.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTO</td>
</tr>
<tr>
<td>Carl S.</td>
<td>50%</td>
<td>50%</td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Isaac N.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Albert E. (PO)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTO</td>
</tr>
<tr>
<td>Marie C. (SM)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JRY</td>
</tr>
</tbody>
</table>

Sprint Burndown Chart & Factors Slide - The burndown chart tells an interesting story especially if the team had to overcome some significant increases in hours. Of course, not every sprint’s tasks will reach zero hours and it’s important to clarify why.
Sprint Burndown Chart & Factors

Factors that contributed to not reaching zero task hours:
- A last minute change to the reports came too late to be tested completely (3 hours).
- The final regression tests on the App Gateway weren’t finished because the testing automation tool crashed (2 hours).

Sprint Challenges Slide - As a testament to how the team deal with adversity during the sprint, this slide lists the challenges they had to overcome. It also serves as a conversation starter around problems that could be solved for future sprints thereby increasing the team’s efficiency and effectiveness.

Sprint Challenges

1. The team lost a member when he decided to retire after winning the lottery. The other team members picked up the extra work without having to drop any backlog items.
2. The automated build process continues to fail unexpectedly requiring manual intervention to complete.

Sprint Backlog Items Slide - This slide lists all the backlog items worked on during the sprint including those that were not completed.

Sprint Backlog Items

<table>
<thead>
<tr>
<th>Backlog Item:</th>
<th>Story Points:</th>
<th>Complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Current Weather Conditions:</strong> As a user, I want to ...</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Request Logging:</strong> As the system administrator, I want to ....</td>
<td>13</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Story Points Total / Accepted:</strong></td>
<td>18</td>
<td>(TBD)</td>
</tr>
</tbody>
</table>
Developer Demonstration Agenda Slide - Defining this for the team helps them plan who will demonstrate and how to get ready for it. Additionally it helps the Product Owner and any observers follow along with what they are viewing.

Developers Demonstrations

1. Louise P.
   - **Current Weather Conditions**: As a user, I want to have current weather conditions for my travel destination so I can get a sense of what it’ll be like for my trip.

2. Edwin H.
   - **Request Logging**: As the system administrator, I want to have application requests logged in the Transaction Logging Database so I can run reports showing traffic activity.

The Sprint Review presentation can include additional slides as the team thinks is needed. Remember that this is an opportunity to show just how much work was done and to show off presentation skills. Including diagrams, performance testing metrics, documentation or other interesting explanations of how the work was achieved can be valuable.

The last two slides (Sprint Backlog Items & Developers Demonstration Agenda) should be printed out and given to the Product Owner for reference. This help them follow along with the demonstrations and make notes regarding what they’re seeing.

Sequence of Events

Assuming the Sprint Review presentation is used, and again, this is recommended for newer teams, the meeting follows a prescribed sequence of events. Once the presentation reaches the Developer Demonstrations slide, the first Developer takes over the presentation, plugging into the projector or logging into the shared computer.

The Developer starts out by indicating which backlog item will be demonstrated, providing context or some explanation of how the work was done. It’s always helpful to observers not normally associated with the team and product to get a little background on what they’re about to see.
The backlog item is then demonstrated and if necessary, Product Owner questions are addressed. Observers can engage as well but the Scrum Master should intervene if they’re getting off topic and keep everyone focused on the primary purpose of the meeting. Time for new ideas or general discussion is best done after all the backlog items are demonstrated.

Once a backlog item is demonstrated, and sometimes these are done in small groups if they all relate to the same feature of function, the Scrum Master asks the Product Owner if there are any more questions. If not, one final question is asked of the Product Owner: Do you accept this backlog item?

The Product Owner should answer either “Yes” or “No” and if they’re not accepting the work, they should explain why. If the Product Owner had been participating in Daily Standup meeting and perhaps gotten previews of work throughout the sprint, an answer of “No.” shouldn’t occur too often. If it does however it will be disappointing to the Developers and the explanation of why should be done tactfully including any explanation of what would make it acceptable. Why the team failed to complete any work to the satisfaction of the Product Owner should be topic #1 in the upcoming Sprint Retrospective meeting in order to find the cause and how to avoid it in future sprints.

Assuming the Product Owner answers “Yes.”, then the entire team should have a round of applause for accomplishing the work. This may sound campy but Sprint Reviews should be a celebration of accomplished work and clapping is a nice touch.

After each backlog item is demonstrated, the next is shown until the entire set is finished. Then, the meeting ends with any outstanding questions or discussion topics.

Once a Scrum Team is more experienced and observers don’t often attend the Sprint Reviews, some of the ceremony can be dropped for efficiency but it is good practice for the Scrum Master to continue to create the presentation file for each sprint to keep a record of the team’s work.

Immediately after the Sprint Review meeting, the team goes into the Sprint Retrospective meeting to talk about how the sprint went and what could be improved.
Sprint Retrospective Meetings

An important tenet of Scrum is the idea of continuous review and improvement. The Sprint Retrospective meeting is designed specifically to bring the team together to review, discuss, and decide on how to improve for the next sprint.

The meeting should be held at the end of the sprint after the Sprint Review meeting when successes and failures are fresh in the team members’ minds. The length of time depends entirely on the team but starting out with an hour allocated ensures that some meaningful conversations will occur assuming it takes the team a little time to warm up. Once a team is more experienced, there won’t be as much need to guide the discussion as the team will be used to bringing up topics and the meeting can be much shorter.

Only the Scrum Team members are allowed to attend the Sprint Retrospective meeting because any outsiders, especially managers, will stifle discussions and some topics may be particularly embarrassing to individuals.

The Sprint Retrospective meeting could be a freeform discussion but it can be helpful to guide the process, especially for new teams. There are three essential topics for the team to discuss and the Scrum Master should guide the process.

Who did we appreciate during this sprint?

Most teams find this a little odd at first but it does provide value. First and foremost it’s an easy question to answer and warms the group up for discussions. Even the quietest of people can usually think of someone who helped during the sprint to make their job easier but if not, they shouldn’t be forced to. For example:

- **Jane Coder**, for training Joe Tester on the how to deploy gateway builds to the testing servers.
- **Dave Helpdesk**, for opening up permissions to the testing servers on a Saturday so we could kick off testing first thing Monday morning.
Naming people within the team (Jane Coder) or outside the team (Dave Helpdesk) is perfectly acceptable. When non-team members are named, the Scrum Master should make a point of informing them along with their manager to acknowledge their contribution to the team’s success. A public thank you goes a long way towards encouraging support again in the future. This list of appreciations isn’t typically long and if there are no other suggestions, then the Scrum Master should move on.

What went well during this sprint and how can we benefit from it?

At this point in the meeting, the team is warmed up and ready for the next question regarding what went right in the sprint. This is designed to help identify those practices or new ideas that occurred within the sprint and could be useful going forward. For example, if Jane Coder had written a script to clear out an old software builds and install the newest one and the software testers found it useful, this should be called out:

- The testing script that removed the old build and installs the new one could potentially save a couple of hours each sprint if it were modified to take command line parameters regarding the destination of the new build.

In the example above the tool was found to be useful and with some minor upgrades could be used by anyone on the team thus saving significant time. It could be suggested that the script be updated early in the next sprint and made available to the entire team. An Action Item should be recorded in the Sprint Retrospective summary to reflect the necessary work and ownership.

What didn’t go well during this sprint and how can we make it better?

Rarely will there be a sprint where everything went perfectly and nothing could be improved so this is the really important question to ask during the Sprint Retrospective meeting. The Scrum Master should encourage even the most trivial of issues since improving them for the next sprint will have an positive impact. Bigger issues are even better with significant impacts to team effectiveness. Following are two examples, big and small respectively:

- It is taking up to fifteen minutes to check in code each time causing Coder boxes to lock up rendering them unusable each time. It has been discovered that indexing function is running
Switching to Scrum - William Patrick Swisher

upon each code check-in causing the problem. We should change the run time to night to eliminate the problem.

- There’s no standard way we name our build files and it causes confusion with the testers requiring them to ask around for which files should be used. We should create a naming standard and use it for all future build files.

Anything could be improved including how a team communicates, processes they follow, or tools used and by exposing each problem and figuring out a solution improvements are made with each sprint. Over time the bigger issues are eliminated which helps not only improve efficiency but team morale as well because they can do their jobs with a minimum of distractions and frustrations.

If problem areas are associated with individuals great care should be taken to bring these up in a constructive manner. Someone will make a mistake occasionally and it is best if the root cause and a solution are identified versus any personal attacks on the person.

When a team finds it hard to identify topics to discuss, the Scrum Master should ask questions to illicit a response. This is especially useful when there was a significant failure and everyone is reluctant to bring it up. Since the only way to improve is to tackle the issue head on, the Scrum Master has to call out the obvious failure and encourage finding a solution.

Documenting Learnings

The Scrum Master, having led the conversations and sometimes asking provocative questions, should document the learnings of the team. This summary may be shared with the larger organization and therefore sent to the team first for review before opening it up to a wider audience.

Issues, especially those critical of people or sensitive topics should be written in a factual, non-emotional way. Leaving names out of the document is strongly suggested and only alluding to roles when absolutely necessary. For example, if Joe Tester had carelessly left the testing servers in an unusable state before going on vacation, the note in the retrospective summary could include something like: “When team members are expected to be out on vacation, they should spend some time familiarizing another team member with their environments and
work underway.” The discussion during the meeting will be lesson enough for Joe Tester and his name need not appear in print.

Sprint Retrospective meetings are very valuable to improving performance of the team and the Scrum Master should identify a few key learnings from each session to remind the team about during Sprint Planning meetings. See Appendix A, “Sprint Retrospective Summary - Sample” for a complete example summary.

Relative Effort Estimates

In Chapter 4, “Initiating Agile Development (Sprint Zero)”, the concept of the Story Time meeting was introduced. These meetings provide an opportunity for the team to understand what’s coming up next in the Product Backlog, give feedback to the Product Owner, and identify dependencies.

Relative effort estimates are a way of judging how much effort a backlog item will take to complete by what it involves and how it compares to other backlog items. Traditional estimates that produce some measure of time (hours, days, weeks, etc) can be wildly inaccurate depending on factors such as how new the team is, the complexity of the requirements, and familiarity with the technology needed to accomplish it. With relative effort estimates, the team can produce estimates that will enable the Product Owner to calculate when products will be complete enough for shipping.

Units of Measure

The key concept with relative effort estimates is that each backlog item’s estimate is relative to others. Whatever label is used to define the sizes, Story Points, T-Shirt Sizes, Potatoes, et cetera these labels are not associated with a unit of time. This is sometimes a challenging concept for software engineers because they are expected to be accurate and thorough in their daily job and over time work experience teaches them that the word “estimate” is often associated with a commitment to complete the work in the time they give. To produce relative effort
estimates each person will have to rely on gut-feel or instinct based on their experience and ignore the compelling need to be accurate.

Choosing which unit of measure is entirely arbitrary but sometimes new teams do better with more abstract units like T-Shirt Sizes versus Story Points. Ultimately they are the same, just a label for the size of effort. However because Story Points use numbers (1, 3, 5, 8, 13, 20..), it can be challenging to not associate those values with a standard measure of time like hours or days. For the rest of this section T-Shirt Sizes will be used and referred to as “size”.

Training the Team

Learning relative effort estimating requires the entire team’s participation because when they start estimating on backlog items, it will be important that they have all heard the same message and experienced the same calibration of what each size means.

Team training should take place in a conference room away from any distractions with at least an hour to learn the process and practice on real backlog items. This exercise may be a challenge for some so it’s not advisable to pick a time when everyone is tired or thinking about end of sprint activities. Ideally, pick sometime in the middle of a sprint where the big planning work is done, and they’re in an execution state of mind.

There are three distinct parts to training: (1) explanation of relative effort estimates, (2) practice on non-project sample backlog items, and (3) calibration on real backlog items. This format has proved to be effective at imparting the knowledge and acclimating the team with the new process.

Explanation of Process

The traditional estimate process calls for Developers to breakdown business requirements into tasks, identify dependencies, designate owners, and come up with durations. This can be a very time consuming process even for an experienced team.

Relative effort estimates involve assigning a size based on the relative effort involved to produce it as compared to other backlog items. This is a very important concept to convey and should be done so very clearly. Using a presentation slide deck is a good way to step the team though the explanation of relative effort estimates and that:
• They should not consider estimates to be indicators of hours, days, weeks or any other time period.
• They will not be held responsible for taking longer than the estimated value to deliver the backlog item because it is not a measure of time, only effort as compared to other backlog items.
• They should consider all aspects of delivering the backlog item in the estimate. This includes coding, testing, database, documentation, meetings, research, analysis; anything that it takes to declare a backlog item complete.

Additionally, to reduce the complexity of their decision making process, the number of estimate values can be limited to only five: extra-small (XS), small (S), medium (M), large (L), extra-large (XL). This set of sizes is representative of what you might find on clothing, hence the name of T-Shirt Sizes.

At this point the team can be given the set of card with sizes printed on them. Professionally made card decks can be found easily but in a pinch a sheet of poster-board that will fit in a printer can be used to make a number of sets for the team:

![Card Sizes](image)

Each Developer will need a set of cards and if there are extra during training, the Product Owner should participate to understand the process.

Now the team is ready for instructions on using the cards. The following steps should work and can be modified as the team determines variations that they like better.

1) As a group, the team reviews the next unsized backlog item. The Product Owner gives any context and answers Developer’s questions.
2) The Developers discuss how to complete the backlog item making sure they consider all aspects of what it takes to meet the
Definition of Done. They discuss how it compares to other backlog items’ estimated sizes.

3) When each team member thinks they are ready to select a size, they pull it out of the deck and hold it facedown so others cannot see.

4) When all team members are ready, the Scrum Master asks them to show their cards all at once. Keeping each card hidden is important because we don’t want one team member’s selection to influence others as any differences are important to address.

5) With all cards showing, the Scrum Master looks around the table and looks for low and high values. For example: S, M, M, L, M. Starting with the Developer with either the low or high card, the Scrum Master asks “Tell us why you selected that size.” Notice that there’s no judgement implied in that request because the size might be perfectly appropriate based on the person’s understanding. The Developers with the outliers explain their position and the team discusses their thinking.

6) When enough discussion has occurred and the team is ready to re-estimate, the Scrum Master calls for another round of cards be shown face up.

7) Again, lows and highs can be discussed or the team members may decide to adjust their estimates to match and reach group consensus.

8) The size estimate is then recorded for the backlog item and the team moves onto the next one.

Once the process has been explained, the team is now ready to practice what they’ve learned. There may still be questions about the steps so it may be helpful to try the process and then address any remaining questions.

Practice Estimation

Now that the team knows the basic concepts, has seen the sizing cards, and has gone over the process, it is time to practice in order to really understand how relative estimating works. While it is tempting to start with actual product backlog items, it is best to work with something everyone can relate to that represents concrete concepts versus the abstract ones associated with building software. The following example uses the project of remodeling a bathroom.
Chapter 5 - Scrum Development Continues (Sprint 1 to n)

Notice that the list is not in order nor does it have any details about each of the backlog items. This is on purpose because the team needs to learn how to engage with the Product Owner to clarify understanding. Assumptions lead to disappointment so in this training exercise, the Scrum Master acts as the Product Owner and the team asks questions; for example, the team might ask:

- **What kind of mirror? What size?** => It’s oval and about 2 feet wide and 3 feet tall with a single wire in the back that can be hung on a nail.
- **Does the vanity come with a faucet and plumbing built in?** => No, you’ll have to use the existing faucet and piping found on the old vanity.
- **Does the demolition include taking the debris to the local dump?** => No, there’s a dumpster sitting on the driveway that will be taken away when full.

As the team asks questions and discusses with each other they’re building a sense of the level of effort for each one. After some discussion, the team is asked to identify a small (S) backlog item. There probably won’t be a clear selection and some discussion and perhaps more questions will occur but ultimately the team will pick one. Using the sample set above, “hang mirror” is selected.

<table>
<thead>
<tr>
<th>Backlog Item</th>
<th>T-Shirt Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>hang mirror</td>
<td>S</td>
</tr>
</tbody>
</table>
With that backlog item in mind the team should estimate each of the remaining backlog items by comparing the relative effort of each. For example, is “install vanity” a little more work than “cleanup” or a lot more work. Through discussion and comparison, the team decides between medium (M) or large (L) and then the final size is chosen.

<table>
<thead>
<tr>
<th>Backlog Item</th>
<th>T-Shirt Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>hang mirror</td>
<td>S</td>
</tr>
<tr>
<td>install vanity</td>
<td>L</td>
</tr>
</tbody>
</table>

As the team progresses through the list there may be a point where they encounter a backlog item, “cleanup” when compared to “hang mirror” seems like a true small (S) size. In this case the team can reassign “hang mirror” to be extra-small (XS) and use small (S) for “cleanup”.

<table>
<thead>
<tr>
<th>Backlog Item</th>
<th>T-Shirt Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>hang mirror</td>
<td>S =&gt; XS</td>
</tr>
<tr>
<td>install vanity</td>
<td>L</td>
</tr>
<tr>
<td>cleanup</td>
<td>S</td>
</tr>
</tbody>
</table>

This adjustment is normal as the team starts reviewing the set of backlog and actually helps refine their understanding of the sizes and what they mean. Once the team seems to have the hang of the process, it is time to work on real backlog items.

**Calibration**

The last part of training turns the team’s attention to real backlog items so they start building up real and useful estimates. The best way to start is to select several recently completed backlog items ideally from the last sprint. These will be most fresh in mind and easiest to do. If the last several backlog items are homogeneous, that is, they are either defining the same thing with minor variations or are all of similar complexity, add a few more from even older sprints to add variety to the sample set. Ideally there should be at least three sizes assigned in this last part of training so when the team moves onto new backlog items there’s more size examples to work with.
Sizing real backlog items is exactly the same as with the bathroom remodeling example but now the team is working with items they are very familiar with so they might be a little more serious about it. If they appear to be over analyzing and getting bogged down in the minutiae they should be reminded that there is no perfect answer nor are they committing to any timeframe. These estimates should come about after a broad but not too deep analysis of all the factors a backlog item could involve.

As the team estimates, watch out for behaviors contrary to the goal of producing holistic sizes based on not only the entire effort necessary but how it compares to previously estimated items:

- **Limited Scope Considered** - It is natural for individuals to think of only their contribution to completing a backlog item but for this exercise they have to think of all the effort involved. If this seems to be happening, have the team members talk through their part to get it all out in the open. It might start with the Coders explaining what they’d build, then Database Administrators explaining what tables will be needed, then the Testers walk through what they’d need for testing and how to do it.

- **Making Isolated Estimates** - As the team defines the work necessary to complete the backlog item, they may jump directly to providing the estimate without thinking about how the item compares to others already estimated.

- **Creating Size Definitions** - Because this kind of estimating process is new and challenging individuals may start defining what each size means. For example, may equate a small (S) to two-man days or less. This seems reasonable but creating such definitions to cover all types of backlog items would require either very high level definitions that will have to be interpreted each time or very specific definitions that will become extensive and forever need updating with even more granular rules.

- **Not Adjusting Previous Estimates** - As a team continues their estimates there may be times when a size given earlier in the meeting no longer makes sense. Either the team remembers something relevant that would affect the size or a new item’s size seems like a better fit and the old estimate doesn’t seem right. It is ok to adjust within the estimation meeting but not advisable to
correct past estimates because over time, while there may be too-
low or too-high estimates, they will all average out.

- **Unknowns Causing Overestimation** - When a team member
doesn’t understand the backlog item or it involves something
they know little about the typical response is to overestimate.
When this occurs the Scrum Master should see if someone else if
familiar with the technology and can provide some training thus
reducing the estimate. If left unchecked this will skew the
estimates toward the high end because there will always be
things about new backlog items that will be unknown.

Once the team is done reviewing completed backlog items they should
move onto current sprint items and then completely new backlog items if
there’s time in the training session.

**The Next Estimation Session**

For the next estimation session the team will benefit from having a copy
of the previously estimated backlog items handy as well as a brief
description of the process they should follow. The Scrum Master can
provide this information and quick revisit of the process to ensure
everyone starts off together.

Story Time Meetings are the optimal time for relative effort estimating
because as they are learning about new backlog items they are gathering
enough information to provide estimates. There will be times however
when previously unseen backlog items are first shown to a team during
Sprint Planning. In this case the process of estimation should remain the
same and the team shouldn’t be allowed to break the item into tasks, then
estimate, because then they’re using two different methods between the
meetings.

Consistency in how backlog items are estimated is the most important
factor. A team could run high or low on average and that doesn’t matter
so long as they are consistent. After a few estimation sessions the teams
will get better at it and much faster relying on past work to judge future
work better. Instead of estimating a few per hour, the teams will be able
to knock off a dozen or more.
Using Relative Effort Estimates to Predict Release Dates

With each Story Time meeting, additional backlog items are assigned size estimates and once the Product Owner has enough to cover the backlog items they’ve designated for a product release, the probable release date can be calculated. This is done using the team’s Velocity, or average work accomplished each sprint divided into the remaining work in the Product Backlog.

Assuming the team is estimating in T-Shirt Sizes and those values are being recorded in the Product Backlog, the Product Owner can assign a value to each size in order to do the release date calculations:

The values chosen for each T-Shirt Size can be anything but using the first few numbers of the Fibonacci sequence is common (1, 2, 3, 5, 8). The important thing to remember is to be consistent. So assuming (XS=1, S=2, M=3, L=5, XL=8), a team’s sprint totals may look like this:

<table>
<thead>
<tr>
<th>Sprint #:</th>
<th>T-Shirt Sizes</th>
<th>Velocity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M,M,L,S</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>S,S,S,L,L</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>M,L,XL,S</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>XS,XS,L,M,M</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>M,M,L,L,XS</td>
<td>17</td>
</tr>
</tbody>
</table>

In the example above the team’s average Velocity is 15 and given the total value of all the backlog items needed in a product release, some quick math can determine how many sprints are left:

\[
\text{Total value of backlog items} / \text{Velocity} = \text{(# of sprints left)}
\]
This number of sprints is often quite accurate because it is based on actual team performance. Since a Product Backlog may change over time with backlog items being added or removed, the calculated release date can be recomputed anytime and certainly after each sprint when the latest Velocity is known.

Hidden Value of Estimation

Even if the Product Owner decides not to determine a team’s Velocity and use it to calculate a release date, the exercise of estimating engages the entire team to think about how to complete each backlog item and that alone is worth the effort of doing the estimates.

Managing Defects

Defects are a natural part of the software development process and every organization will have some established policy for managing them. Typically all defects are logged in a tracking system which has a defined workflow that ensures each one reaches some acceptable end state (e.g. Fixed, Not-a-Bug, Deferred, etc). Within the Scrum methodology formal testing is part of every sprint, not just after the bulk of coding is complete so there are some additional considerations when managing defects.

When defects are found and are directly associated with current sprint work they should be resolved before considering backlog items Complete. Some defects will need to be fixed without question but others may be minor enough that the team feels they can be deferred. In these cases, the Product Owner can be asked to render the final judgement and they will either choose that the defects be addressed within the current sprint, added to the Product Backlog, or dropped altogether. When defects are deferred to the Product Backlog, the Product Owner will prioritize them accordingly.

Conversely, when defects are not directly associated with current sprint work, there is no obligation to fix them within the sprint but the Product Owner can make the decision regarding how they are handled. If the defects are to be deferred, they are added to the Product Backlog and prioritized. If not, then the Product Owner should negotiate with the team to address the defect within the current sprint.
Defects will accumulate within the Product Backlog over time but so long as the Product Owner is prioritizing the backlog items, including defects, those that are important enough will rise to the top and be resolved.

Changing the Scope of the Current Sprint

Ideally, once a Scrum Team accepts work into a Sprint Backlog it can no longer change. If some new backlog item suddenly becomes the priority it should remain in the Product Backlog until the next Sprint Planning meeting. However nothing in business remains static for very long and there will be times a Product Owner wants to change the contents of a sprint. This can be done and involves the Product Owner and Scrum Team discussing the options and negotiating a resolution.

The best way to approach this is to extend the Daily Standup meeting to allow the Product Owner to present their request and allow the team to review, ask questions, and consider if it could be brought into the sprint. It is often the case that once sprint work starts, it is more involved to remove partially completed work than to finish it off so the Product Owner must defer to the Scrum Team’s judgement regarding what they can do. The team must be allowed to work through if they can accept the new backlog item because they have already committed to completing the work currently in the sprint.

Sometimes, the need to change the contents of a sprint comes from the Scrum Team itself. It may the be case that a backlog item, once planned to the task level, is discovered to be too much to complete within the sprint or there are dependencies required before the work can be done. In these cases, the process of negotiation is the same. After the Daily Standup meeting the case is made to the Product Owner to drop or alter some work within the sprint. The Product Owner could approve the request or not and the Scrum Team must carry on.

If during the course of a sprint it becomes clear to the Product Owner that the sprint work will not provide business value and there’s little point in completing the current sprint, the sprint can be abandoned and new sprint work planned to replace it. This is a very drastic measure and doesn’t come without costs as the Scrum Team will have to spend some time backing out half-completed work and of course go through the process of Sprint Planning all over again. The following list illustrates the options for addressing the need to work on backlog items not originally
planned within the current sprint. They are listed in increasing order according to the amount of disruption to the development effort.

1. Add the new backlog items to the current sprint with the full agreement of the Scrum Team because they feel they have capacity.

2. Reduce the acceptance criteria for current sprint backlog items thus opening up Scrum Team capacity.

3. Drop current sprint backlog items to open up capacity.

4. Take on the additional work even though it exceeds the planned Scrum Team capacity with the full agreement of the team.

5. Restart the sprint altogether with a new Sprint Planning meeting selecting the latest business priorities in the Product Backlog.

If in the course of the project both sides are being reasonable and flexible whenever they can be, this process works well and project work gets completed despite the changes to sprints.
Inspection and Adaptation

Success with Scrum depends on a dedicated effort to frequently review and improve all aspects of the new process. The nature of Scrum development with its short sprint cycles lends itself well to the need to inspect, analyze and modify process to be more efficient and effective. Even finding and improving a handful of relatively minor things can have dramatic effects on the entire process.

Some rituals such as the Daily Standup and Sprint Retrospective meetings provide much visibility into daily and sprint progress but there must be more than this. To successfully make the transformation, active efforts need be undertaken to find bottlenecks and failings and make the necessary adjustments across the entire organization.

Daily Standup Meeting

Back in Chapter 4, “Scrum Development Continues (Sprint 1 to n)”, the ritual of Daily Standup meetings was introduced and the text described when it occurs, who attends, the mechanics of answering three questions and updating hours estimates for owned tasks. The purpose was primarily for the Scrum Team to communicate and coordinate but much more can come from this meeting.

The Daily Standup meeting provides a window into how the Scrum transformation is doing. When the Developers provide their updates and express issues or frustrations they are exposing issues that could be localized to the team or something more systemic where adjustments to the larger organization are required.

Observers of the ritual should notice team frustrations and impediments. Of course the Scrum Master should try to eliminate anything blocking the team’s progress as soon as possible but some issues may be more global and will take a larger, more coordinated, effort to address it.

Recognizing systemic issues isn’t easy as everyone will still have the memory of how products were developed prior to the transformation starting. It can be challenging to realize a problem because people
sometimes just assume the way something has been done is simply the right way. An important part of the transformation process is to question everything and determine if something needs to be changed to support the Scrum Team’s goals of delivering potentially shippable product at the end of each sprint.

As problems are encountered that go against standard process or “the way it has always been”, the team should be encouraged to devise a solution, try it and then evaluate and modify as needed. Thinking “outside the box” is what’s required when the entire system has been disrupted by the transformation and the Scrum Master should recognize when this is needed and help facilitate.

**Burndown Chart Trends**

Progress within the sprint can be ascertained from the daily updates each team member provides. Additionally, the Sprint Burndown Chart created from each day’s remaining hours estimates can be quite informative.

The team’s progress on tasks, as measured by remaining hours, tends to decrease over the course of the sprint heading towards zero. There can be interesting patterns, some of which are cause for concern whereas others are not. In the following example the burndown line varies from the ideal burndown rate (dotted straight line) but is generally headed towards zero. It is normal to see this variation from day to day.

However, if the burndown line is headed upwards and a significant part of the sprint is over, then the Scrum Master should be discussing with the team the causes and potential solutions.
Chapter 6 - Inspection and Adaptation

If the upward trend was very early in the sprint, this may simply be a result of the team discovering more tasks or an increase in hours for existing tasks. The Scrum Master should watch the trend and if it continues to climb, the team should discuss how to redirect the line.

On the other hand, if the burndown line is headed to zero much quicker than the ideal burndown rate, the Product Owner should make sure the Product Backlog’s top items are ready for possible selection into the sprint as the team will be looking for work soon.

Sometimes, new teams find estimating the remaining hours for tasks challenging. They typically get used to it within a couple of sprints but if it continues to be a problem an alternative is to only record when assigned tasks are completed. Using the same spreadsheet table to generate the charts above, this can be represented with 1’s and 0’s. A ‘1’ indicates the task is still being worked on and a ‘0’, that it is completed.

Recording the information this way yields a similar sprint burndown chart but with a flat line on the left indicating that no tasks had yet been completed.
The benefit to recording remaining hours versus whether or not a task is complete is that mid-way through the sprint, the remaining hours for a specific team members can be compared to what their capacity is for the remainder of the sprint. For example, if the two testing resources on the team had a combined 40 hours of task work remaining, but only 24 hours of capacity, the Scrum Master can point this out to the team asking for solutions to rebalance the work.

**Daily Standup Email Updates**

Transparency is an important aspect of the transformation especially in the early stages because issues of all sizes and complexity will be exposed during the Daily Standup meeting and some may require significant changes to the organization’s standard process. Proving visibility into the team’s progress and challenges is highly recommended.

Each day after the Daily Standup meeting the Scrum Master should email not only the team with the latest burndown chart but any discussion points, challenges encountered and working solutions decided by the team. Recipients of this email should include Engineering Management the team reports to and any Senior Management who are direct stakeholders in the transformation. Exposing everyone to the trials and tribulations will help facilitate solutions. For a complete sample, see Appendix A, “Sprint Daily Email Update - Sample”.

As necessary, the Scrum Master can send Engineering Management notes on observations or challenges in order to open the dialog on solutions that require more fundamental changes the Scrum Team cannot possible make happen on their own. These are the kinds of topics that might be addressed in the Transformation Oversight Group.

**Transformation Oversight Group (TOG)**

A fundamental part of the Scrum methodology is continual inspection and adaptation typically occurring on a sprint by sprint basis. The Scrum
Team has a defined ritual for this, the Sprint Retrospective, where many improvements, big and small are identified and adopted. Their focus however, tends to be on the immediate process and needs in order for them to complete their work.

What’s also needed is a perspective from a higher level that transcends the Scrum Team and can also see how the bigger process picture is working. This group should meet on a fairly regular basis especially in the early stages of the transformation process and then taper off as appropriate.

**Composition of Group**

The core attendees of this meeting should include the Scrum Masters and the functional managers that represent some speciality like Software Engineering, Quality Assurance, Product & Product Management, et cetera. In general, these people should be attending the public rituals frequently like the Sprint Planning, Daily Standup, Sprint Review, and Story Time meetings to gauge how the process is working and their team members are doing. It’s no different than observing how their direct reports would be doing their jobs when not operating under the Scrum methodology. Their observations can be a valuable measure of how the transformation is progressing and what could be improved.

Additional people can be brought in when needing a different perspective or to float some ideas and discuss the implications. Individuals directly from the Scrum Team are certainly choices as are people in the organization that are affected by the output of the Scrum Team. The Transformation Oversight Group is all about finding the best way to operate in the new methodology. Anyone who is needed should be engaged.

**How Discussion Topics are Gathered**

There are two effective ways to identify topics for discussion, direct observation and one-on-one discussions with direct reports. When functional managers attend the public rituals they should be mindful of how their presence can have an effect on how the team members. This is especially true of more junior employees or for those people from cultures where authority carries more weight.
To minimize the effects of a managerial presence, the managers should respect the purpose of the meeting and make sure any contribution they make isn’t seen as authoritative or controlling. In Scrum, the team should be allowed to self organize and solve problems with their own authority within boundaries. That being said however, sometimes the manager has the answer or can help point the team in the right direction. Making progress is the goal and any distraction from that should be avoided. If the Scrum Master senses that a managers presence is causing a disruption they should address it, because it represents an impediment.

The second way to identify topics is for the functional managers, who presumably have one-on-one discussions with their direct reports, to simply ask about how the process is working for the employee. For example:

- Are you getting what you need to complete your work?
- Are the available tools effective?
- How’s your interaction with other team members?
- Are you getting the information you need?
- What causes frustration for you?
- What do you think we should do differently?

These are only a sample of the questions that could be asked but ultimately what the manager wants to know is: Is this working, and if not, why not?

**How Topics are Logged**

Since discussion topics, which can be either *issues to be addressed* or *innovations to be adopted* will be generated by multiple functional managers, a single shared repository with restricted access works well to store the topics for review by the group. The format can be anything really but a simple spreadsheet works well:
To best describe, organize and prioritize group discussion on the topics there’s a set of essential information to record:

- **Area Affected** - Either the Scrum ritual, artifact, or process the topic pertains to. This could include rituals such as the Sprint Planning meeting, Daily Standup meeting or artifacts such as the Product Backlog or Sprint Retrospective Summary or just be something covering the entire methodology.

- **Category** - Indicate a categorization of the type of topic; this could be a selected from a standard few (Process, Policy, Tools, or Personnel) or anything the person recording the topic wants to use.

- **Urgency** - Since the topics for discussion could either be an issue to be addressed or an innovation to be adopted, the best way to prioritize them is by how soon ought the group review and make decisions. A simple scale works best with ‘1’ representing the need to review it quickly and ‘5’ being something that can wait.

- **Issue / Innovation Description** - Describe the actual issue or innovation or the general observation. Whenever possible include the negative or positive impact occurring.

- **Action** - Once a strategy has been devised to address the topic, it should be logged and noted who owns execution.

Some standard sets of values have been suggested above but of course the Transformation Oversight Group could pick their own or have the fields be open to any values. Standardizing on a limited set makes it easier to focus on a set of topics at a time. See Appendix A,

Discussion and Action

The TOG is fundamentally a problem solving team that uses the logged topics for discussion but should also lookout for patterns of problems and anticipating needs. They should maintain the longterm vision of where the transformation will take them and that guides what issues and innovations to address.

At times it will be tempting to address an issue by defining a set of all encompassing rules but this doesn’t always work. The exercise of creating a set of rules can help define the boundaries in which the team should operate and instead of passing down the latest laws to the teams, reminding them about what the overall goals are with some specific constraints may be more effective. It can be very revealing to define the rules loosely and allow the Scrum Team to find the way that works for them. So long as they are operating within acceptable boundaries, it should be considered successful. For example, an edict could be issued regarding Sprint Planning stating that:

- Sprint Planning must be completed within four hours.
- In depth design should be avoided during Sprint Planning to avoid exceeding the time limit.
- The Scrum Master must ensure the team is making sufficient progress through the planning process to be finished within the time limit.

This could work but, what if a team is taking on a completely new product and requires far more discussion and learning to even know how to complete the backlog items? In this case the rules will be an impediment to preparing to work. If the rules above represent a desire for teams to finish planning and get to work as quickly as possible, then the message to the team could be softer such as:

- It is encouraged that Sprint Planning should be completed within the first day of the sprint. If the team feels the Sprint Planning meeting should be modified to select backlog items only and then have more in depth design discussions they should adjust as
Chapter 6 - Inspection and Adaptation

needed and update the Sprint Backlog with the list of tasks, owners, and initial hours estimates.

The second message conveys the overall meaning, that Sprint Planning should be completed within a day but in a softer way. As the team tries variations on Sprint Planning day, the optimal timeframe and structure will emerge and that can be the new standard. This message can be delivered through the Scrum Master on the planning day or through a more formal message sent from Engineering Management.

Deciding upon the right solution, which may be to let the team find their own solution in some cases, is the ultimate goal of the Transformation Oversight Group. Analysis, debate, and experimentation are the keys to success in guiding the transformation process.

Common Transformation Problems

Every organization’s experience through the transformation process will be different but some problems will commonly occur.

Results Expected Too Quickly

Scrum development is often billed as producing results quickly and being lightweight in process. This is true however making the switch from one established development process to another must take time. The transformation involves not only laying out a new set of processes but training people and dealing with changing their habits. To address this, expectations must be set that at least a couple of sprints will be completed before true results can be seen. Additionally keeping everyone informed about important milestones, such as coding and testing environments established, new process defined, initial software delivered et cetera, should give everyone a sense that the transformation is progressing.

Leadership Perceiving Development Output Has Slowed

It’s fairly common in software companies that the pool of software engineering resources are shared across projects. Individuals may not be coding on each project but they’re involved for research, design, and defect fixing. All this activity is typically claimed as active progress on each project but doesn’t necessarily translate into delivered software
products. Once a team switches to Scrum, their work within a sprint is clearly defined and while individuals could be tapped for other activities, they should be allowed to fully focus on the sprint work and have full engagement with the rest of the team. This may result in the perception that less work is being done but in fact, more complete work is being delivered. As the teams start delivering increments of business value consistently each sprint, the perception of output will change.

**Product Backlog Still Written like Business Requirement Documents**

Despite the Product Owners attending training and having sample backlog items to reference there can be a tendency to revert to writing backlog items as if they were the requirements statements. This is in part because they are used to the old methods but also because it seems easier. To rattle off a list of requirements takes little effort but often the list lacks focus on any one part of the overall product and will result in parts being skipped.

Writing proper backlog items, each with a user story and specific acceptance criteria may seem like less progress but because the item represents a discrete chunk of business value (e.g. a login screen) the author is able to focus on more of the details required to call it complete. Then, when the Scrum Team reviews the item, they can understand the overall goal and help fill in the acceptance criteria gaps that the Product Owner missed. A few well written and comprehensive backlog items will represent more work for the team than would a long list of disjointed requirements statements. Additionally it makes the team partners in the building of business value versus “order takers” cranking out code. As a Product Owner gets used to writing backlog items representing business needs, they tend to find it easier to define products at the business level versus a technical level because it mirrors their thinking about features and how to sell them to clients.

**Status Reports no Longer Reflect Development Process**

Every organization has some form of status reporting and the higher the reports go up the chain of command the more summarized and formal they usually become. The transformation to Scrum will bring about changes to how products are designed, coded, tested, and delivered and
Chapter 6 - Inspection and Adaptation

d this will have an effect on what can be, and should be, communicated in status reports.

Under the Waterfall methodology project status is often conveyed in terms of the estimated delivery date based on some technical analysis completed before coding was started and periodic reports on progress as compared to the initial timeline. As issues arise, they are explained with estimates on how they might affect the delivery date and when intermediate milestones aren’t met there’s always the questioning of how the project can be accelerated to catch up and meet the estimated or decreed delivery date.

Under Scrum, because the team focuses on delivering business value right away versus analyzing requirements and producing an estimate, the way progress is shown should be changed. Of course management will always need to know when a project will be done because there are expectations to be set and activities to coordinate. This can be easily accomplished using measured team performance from sprint to sprint along with relative effort estimates for the remainder of the Product Backlog. The essential parts of a report should include:

- **Product** - Include a unique name and version number to facilitate communication and list the primary goals or expected benefits of the work.

- **Calculated Delivery Date** - Using actual team performance each sprint (Velocity) and the remaining work in the product backlog for the release the number of sprints remaining can be calculated. The Product Owner can add additional time based on the activities needed to ship a product but should refrain from just padding for the sake of ensuring they make the date. The calculated delivery date should be the truest reflection of reality to keep it easy to create and build trust in the value provided.

- **Current Sprint Work** - List business value expected (features, functionality, or knowledge gained) from the current sprint and any recent accomplishments or challenges overcome by the team.

- **Next Sprint Work** - List the likely work to be taken into the next sprint based on the current prioritization of the product backlog. This can change of course based on business needs but should be stable enough to list in a few bullet points.
• **Issues & Risks** - List any issues currently affecting the team and their work and if available how they are being addressed. Additionally, listing risks with the probability of their occurrence and impact will alert the reader to any potential issues and spark discussions about how they can be mitigated.

The actual format for the report can vary but a simple spreadsheet works well and can be updated easily from week to week. See Appendix B, “Weekly Management Report - Sample” for a complete sample report.

<table>
<thead>
<tr>
<th>Product</th>
<th>Calculated Delivery Date</th>
<th>Current Sprint Work</th>
<th>Next Sprint Work</th>
<th>Issues / Concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>* Create interface to accept app requests.</td>
<td>* Create internal reports.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Create app request database.</td>
<td>Move reporting database into production.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Design reports.</td>
<td>* Hold training sessions for Customer Support.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Design &amp; create databases for logging.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Document database schema design.</td>
<td>April 20: Design discussions are continuing but there's a significant disagreement about the right architecture.</td>
<td></td>
</tr>
</tbody>
</table>

Weatherly Android v1.0.0 | Q2 2013 (est.) | April 20: There's a shortage of qualified Android programmers so the search area will be expanded.

This report information should provide a comprehensive summary of progress on the active ones and an indicator of upcoming products. Every organization is different and the design can be modified as needed. Calling out progress from sprint to sprint and showing that progress is being made early usually satisfies the report viewers and of course they can always attend any of the Sprint Review meetings to witness for themselves what teams deliver at the end of each sprint.

**Inconsistent Messaging Regarding Transformation from Management**

The transformation to Scrum will be very disruptive and sometimes the significant changes within an organization can be challenging for many people. The organization has to believe that there’s no turning back to the old ways and if there’s the slightest hint that some members of management aren’t quite on board with the goals, this could give nay-sayers and those having a hard time changing the opening they need to make their case.
Once the decision has been made to convert the organization every manager within the engineering organization must convey the same message; that there will be bumps, which is to be expected, but we will be making the transition and within the specified timeframe.

Team Members Challenges Operating Under Scrum

When any disruptive organizational change occurs each person can have their own unique reaction but they generally fall into broad categories. The following represent possible scenarios and how to address them.

- **Team members finding it challenging to overcome years of operating under Waterfall.** People can very much be creatures of habit and once they get used to one methodology, even with the slight variations seen across organizations, it is hard to do something very different. For those who are having a hard time for no specific reason other than it is so different, provide training courses with hands-on activities that teach the mechanics and how to think differently under Scrum.

- **Team members who are losing power or influence operating in the team environment.** In organizations there are sometimes those individuals who wield great influence and are seen as the leaders, facilitators and sometime heroes because of their ability to put out fires and deliver when help is needed most. When those individuals are suddenly demoted as it were to working as part of a team that is responsible for selecting and delivering work together it can seem to take away some of that prestige. It is best to channel their energy by asking them to mentor the less experienced on the team or to seek their opinion when facing team decisions where no strong preference is clear.

- **Team members not understanding the new methodology.** Offer additional training sessions with those individuals, covering the overall goals of each part and walking through the process. Team them up with someone more experienced for guidance and never make them feel like they’ve failed somehow because it’ll only drive them back to the old familiar ways of Waterfall.

- **Team members subverting the transformation process.** Individuals subverting the conversion effort may be subtle or outright and it’s very hard to address either way. When this occurs, have a talk with them on how they think the new
methodology is working and how they think it could be improved. Listening to concerns and taking action wherever possible may win them over.

- **Team members who had a bad experience before.** This one is especially challenging but the remedy is much the same as the previous case. Talking with those individuals is the best bet in understanding their objections in light of past experience and while no promises can be made that “this time will be better” invite them to utilize their negative experience to help guide the current transformation.

- **Personality conflicts when adjusting to the new methodology.** Such conflicts will probably arise when team members differ on how to accomplish some goal or perform some process. Both sides may have completely valid opinions with evidence to back them up but forward progress cannot be paralyzed by indecision. Establish a culture within teams of hearing all opinions, providing rebuttals, and the entire team voting to decide which one to do try. The chosen path for any given decision may be the wrong one but until something is tried and evaluated, it may never be known.

**Problems within a Sprint**

There’s any number of problems that new teams will encounter; the following represent fairly common occurrences during a transformation.

- **Team members running out of work.** Depending on the makeup of the team and the nature of the work there may be times when a single team member or a role (Coders, Testers, etc) may complete their work before the sprint is over. Of course since the entire team is responsible for delivering work according to the Definition of Done, everyone should pitch in to help where they can. Coders can help test to a degree, Testers can help prepare environments and so on but there will be limits to this. Each team should know that there are alternative tasks that go beyond sprint work but are valuable for the effectiveness of the team. For example, Coders can write unit tests to improve quality of code, peer code reviews can be held, or learning about environments and code bases necessary for future backlog items. Testers can invest time in automation, either learning about tools or creating
reusable test suites. They can also prepare for future backlog items by setting up environments or planning test strategies. There’s no shortage of work for a team, especially in the beginning of the transformation and any spare time can be utilized productively.

- **Testing effort consumes a high percentage of sprint time.** Sometimes the level of effort to code versus test is a high ratio, sometimes it’s the other way around. These conditions should be apparent during Sprint Planning meetings and planned for but if not, as was explained in the previous bullet point about having capacity, those who have time can apply themselves to valuable work that improves the effectiveness and efficiency of the team.

- **Team members focused solely on their traditional roles.** In the early stages of the transformation, this is fairly common with individuals trained and experienced in one speciality or another. There’s no single remedy for a person limiting the scope of what they do to match their profession but they should always be encouraged to step in to help their teammates where and when appropriate.

- **Waterfall process appearing within sprints.** The pattern of design, code, test, and ship can be so engrained in team member experience that it appears even within the short sprint cycle. It may not be noticed right away but after a few sprints if the pattern shows that design and coding occur in the first half and testing is concentrated in the second, the team is experiencing a “mini-Waterfall” pattern. It may seem to work but will break down when the coding part takes longer than expected and starts to squeeze the time left for testing. To address this, the coders and testers must work out intermediate handoffs of work so instead of: (code, code, code / test, test, test) the pattern is (code / test.. code / test.. code / test.. etc). The Scrum Master can encourage the team by watching for when backlog items become code-complete and calling attention to the work being ready for a handoff. This should establish a pattern that the team will pick up on and do themselves.

- **Scrum Teams and dependencies with external resources.** Some projects will have dependencies on resources external to the Scrum Team. This could include things like analyses of data that will shape how a product works, or new hardware requirements that have a long lead time. When a team plans a sprint they should identify these dependencies and factor that into the
decision of whether or not to accept specific backlog items into the sprint. During Story Time meetings, where the team looks ahead into the Product Backlog, these dependencies should also be identified and in both cases, the Scrum Master can help coordinate the deliverables to ensure the team’s progress is unimpeded.

- **Established process / tools from Waterfall era impeding Scrum era.** The transformation doesn’t change the entire organization from top to bottom all at once and so there will be some established process or tools the Scrum Team will be dependent on that worked well under Waterfall but won’t be effective or effective enough for the fast paced Scrum environment. Some of these incompatibilities may be foreseen before the transformation and every effort made to adjust them but often they won’t be known until the team experiences an impediment. As part of the ongoing effort to streamline the organization and adjust to working under agile, analyzing these impediments and making adjustments is crucial to success.

**Resistance Outside of Engineering**

Since the transformation focuses on software development much of the focus will be on that part of the organization. It will however have ripple effects across the organization and there may be resistance to changing outside of the software engineering group.

Resistance may be caused by any number of reasons but in general it comes down to people knew how to operate under the old system and that has all changed. People will want to revert or modify the process and under Scrum they cannot just be told ‘no’ or their opposition will grow. Establish conduits for their needs and grievances and address each respectfully. Show them how to accomplish what they want within the new system and include their feedback in improvements to gain their support and quell their feelings of being forced into the new system.

**Whole Organization Analysis**

From the beginning of the transformation process the Scrum Team has experienced the most change in process and roles. The areas of the organization tightly coupled with them will also have been affected and
compelled to adapt to the new paradigm. In the following diagram, representing Apptastic’s product delivery process, there are many groups involved. The ones within the dotted line rectangle indicate those that interact most with the Scrum Team and through the team’s transformation also have to adapt their internal process to support them.

The groups outside of the dotted line rectangle will surely change over time as products are conceived, designed, built and deployed. To help identify the necessary changes ahead of time, and before they are forced to, some analysis and streamlining can be prepared for in advance.

**Analyzing Upstream and Downstream Dependencies**

The simplest way to ferret out potential changes is to meet with leaders from each group and ask a question that makes them think about the impact to their area. For example: “We will be creating deployable software products far more frequently than before; how will this affect your group and how you operate?”

The answer to that question should spark an exploratory conversation about where the bottlenecks and blockers will be and how process can be streamlined to support the quicker delivery of products. For example, in talking with the Information Technology group, the issue of long lead times for ordering hardware will be a problem. The solution will be that the Scrum Team has to spend more time looking ahead in the Product Backlog identifying any hardware needs early.

In talking with the Client Setup group who takes over where the sales process left off to set up clients to use the products, they will need more training and a handoff process that is more comprehensive and efficient. In thinking about this further they may also need more personnel to handle the increased load.
The previous two examples focused on groups *downstream* of the Scrum Team but the groups *upstream* shouldn’t be left out. In the case of Marketing, their research and analysis of market trends and customer needs may have to be sped up to accommodate the increased need of Product Owners to define backlog items. Even groups like the Legal Team whose involvement with Software Engineering is minimal at most, may need to accelerate their contribution to end user license agreements or decisions about how to deal with competitors whose market territory the company will go after.

There’s no tried and true formula for flushing out all the areas that will change nor does doing so guarantee that the organization as a whole won’t suffer challenges. However, spending some time identifying and modifying areas that will be affected will smooth out the transformation effort across the whole organization.

**Scrum Resources Repository**

The transformation process is basically an education process and what better way to educate than to create a centralized internal website dedicated to the new development methodology and all the learnings acquired over time. An internal website doesn’t have to be fancy but should be informative and easily navigated. The exact content will vary by organization but the following are some helpful suggestions.

**Main Page**

The main page is the first part of the website a visitor will see and can serve as an index for the rest of the website. The most valuable topics should be only a click away from this page.

- **Frequently Asked Questions** - Create a FAQ list organized by topic with a hyperlinked table of contents for easy navigation.
- **Artifact Templates** - List the most commonly used artifacts in template form so only the most recent copy of the file is ever downloaded.
- **Sample Meeting Agendas** - Include a set of standard meeting agendas to be used when establishing meetings so everyone is following the same process.
- **Best Practices & Learnings** - As challenges are figured out, problems solved, standards agreed upon, these should be
recorded as the best practices and learnings. This becomes especially useful when starting up an additional Scrum Team who can benefit from all the experience gained.

- **Engineering Standards** - Chances are the organization has a defined set of standards regarding how code is written, testing is done, and other such rules. They don’t have to be added to the website but making sure there’s a link to the source provides a quick reference for anyone interested.

- **Codeline & Merging Policy** - As with the Engineering Standards, making the codeline and merging policy readily accessible will be of benefit from someone who’s looking for organizational standards.

- **Calendars** - Having a calendar of any key events in the coming months may be quite useful especially when there are multiple teams.

**Key Concept Specific Pages**

Some topics, such as Sprint Planning, or the Product Backlog are significant enough parts of Scrum that they could have a dedicated page that explain the purpose, the process, frequently asked questions, and links to templates for download. For example:

**Sprint Planning**

| Purpose | Downloadable Template 1  
|         | Downloadable Template 2  
|         | Downloadable Template 3  
| FAQ:    | (1) How do I ...  
|         | (2) What is the ...  
|         | (3) When do you ...  

Detailed treatment on such topics will help educate and synchronize everyone’s understanding of how to work within the new methodology.
Maintenance

With any public information website it will have to be maintained. Some of the content such as best practices and learnings or the frequently asked questions lists will need to be updated regularly to ensure visitors to the site get some value from them. Once the content is seen as stale, visitation will drop off and it may not have the benefits to the organization as it once did.
Expansion

Thus far this text has covered the transformation to Scrum with regards to only having one Scrum Team. In reality most organizations will have multiple development teams and therefore it is possible that they all switch over to Scrum. There may be some cases where it’s not ideal to change from Waterfall because of the nature of the work they do and each situation will have to be evaluated. Even if some development teams do not convert over to Scrum, they may adopt some of the practices that benefit them.

Starting Additional Scrum Teams

Returning to the example company of Apptastic, they have two main lines of business, Direct to Consumer (DTC) and Business to Business (B2B). Within the DTC line of business they have several smaller products but their upcoming flagship product, Weatherly, is predicted to bring in most of the revenue. Because the organization has enough resources to make three Scrum teams, they decided to do just that with one each dedicated to DTC, B2B and infrastructure projects that support all products.

The transformation process started with one team (“Team-One“) but the question is, when to start the second and third? It would make sense to allow Team-One to continue on their journey through all the issues and innovations and to let the entire organization settle down once again after the initial disruption in development methodology. However, it is actually better to continue the disruption with the addition of the second and third team because they will expose new issues and innovation that will ultimately have to be addressed to declare the transformation a success.

Assuming Team-Two will not wait for long, the question remains of when exactly. There is no precise indicator of when is the best starting point, but in general, once Team-One has completed a sprint and experienced all the rituals and used all the artifacts, Team-Two should start. This may seem too soon as issues are still being worked out but at this point there will be “working solutions” to most problems and when the second team starts
using those “working solutions” they are in effect testing the design and their experience will allow for refinement. When Team-Three starts soon after, they too will apply the latest “working solutions” and their experience will once again refine the established processes.

By starting multiple teams in this way, it will result in a lot of work for the teams and Engineering Management who will be instrumental in helping to define process under the new methodology. As bad as this seems it is much better to address all the issues early and settle on standard processes that work well early in the transformation. If the second team started much later all those initial “working solutions” would become standard procedure and much harder to change as they will be part of the organization’s culture. Additionally the rest of the organization would have to contend with dual standards for interfacing between Waterfall and Scrum teams, status reporting, and processes when working as resources external to the Scrum team.

**Sprint Calendar Recommendations**

Assuming each team was using a two week sprint cycle, it would be helpful to start Team-Two’s sprint halfway through the first team’s sprint, effectively establishing two sprint cycles with a Sprint Planning meeting starting every week. In the following table each team’s sprint begins (B) and ends (E) every two weeks.

<table>
<thead>
<tr>
<th>Team</th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M T W R F</td>
<td>M T W R F</td>
<td>M T W R F</td>
<td>M T W R F</td>
</tr>
<tr>
<td>One</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Two</td>
<td>B</td>
<td>...</td>
<td>...</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When adding a third team who will also be on a two week sprint cycle, instead of starting on the same day as Team-One, their sprint could start one day later.
Chapter 7 - Expansion

Using either Wednesday or Thursday as the start of a sprint still keeps Scrum rituals off of Mondays and Fridays which are the most frequent days for time off. Also, the Developers benefit from having the five contiguous days in the middle of the sprint for concentrating on their work.

Regarding the Daily Standup meeting which occurs each day except for Sprint Planning day, it is recommended they start as early as reasonable and because it is important that Engineering Management be able to attend these meetings to gauge process and address intra-team dependencies, the meetings should be staggered to allow for their participation.

In the above diagram, two meeting lengths are covered, one allocated 15 minutes and the other 30 minutes. Having the 30 minutes blocked off for all attendees can be helpful when more in depth discussions about impediments are needed. Of course if Engineering Management is attending all Daily Standup meetings they may end up missing the next team’s time slot if an important issue is raised in the current one.

Whatever timing is used for sprint length or rituals, it should be evaluated like everything else to find the optimal solution for the organization.
Team Synchronization and Innovation

The Scrum methodology emphasizes self-organization where teams determine for themselves how to achieve sprint work and deliver completed products. When there’s only one Scrum team, how they work with all the rituals, artifacts, roles, and processes is being closely monitored and adjusted. When the second and third team start up they will invariably stray from how the first team is operating. In the early stages of the transformation some care must be given to making sure the teams are relatively synchronized for the following reasons:

- By having more than one team striving for the same goal: to deliver complete products while operating within the same organizational processes and constraints; the issues encountered and resolved by one team can be avoided by another. The same is true for innovation where as one team develops something, others can benefit from their experience.

- When teams are left to evolve on their own, they may establish very different methods for interfacing with external resources, how progress is measured, and different understandings of common terminology. The rest of the organization will appreciate having the same interface and expectations when dealing with the different Scrum Teams.

- The people on each Scrum team should be at some point, interchangeable between teams as part of a periodic rotation strategy to cross train and build up redundancy. This will be most common for Developers but could include Scrum Masters and Product Owners as well.

However, while it is important to keep the teams synchronized there’s the danger of overdoing it with too many restrictive rules. Teams should not become clones of each other as allowing differences promotes innovation and that benefits all teams.

Finding the right balance between an all encompassing set of rules and anarchy isn’t easy. But what works well is to establish a limited set of hard rules and then provide guidelines that define the boundaries in which a team can operate. By doing so, the core rituals, artifacts, roles, and processes will remain fairly stable and similar across teams but the team will still have the leeway to innovate. Too many rules will stifle
problem solving and the teams will remain dependent on outside management.

Increased Disruption to the Organization

The first Scrum Team will cause a disruption to the entire product delivery process across the organization. With the addition of more teams the disruption will only intensify as more demands are made upstream to feed the team work and downstream to accommodate team needs as they all try to push products out the door.

Organizational policies and procedures, such as how products are deployed to the production environment or how code is branched and merged, will be strained further with the increase in release frequency.

The Transformation Oversight Group, in conjunction with the Scrum Teams will have to keep watch on how the increase in activity strains the overall system and make the appropriate adjustments.

Scrum Masters’ / Product Owners’ Club

Inspection and adaption is weaved into the fabric of the core Scrum rituals but such in depth analysis for the purposes of improving need not be limited to those meetings and processes. When there are multiple Scrum teams there are usually multiple Scrum Masters and Product Owners and the people who fill these two roles can benefit by learning from each other as they too encounter issues and innovate.

An effective way to share learnings is to have dedicated and reoccurring meetings for each role. For the sake of giving these a name, they can be called the “Scrum Masters’ Club” and “Product Owners’ Club”. Since this meeting is about sharing and learning, the basic agenda should include:

- Recent issues and how they were resolved.
- Recent innovation and their benefits.
- Walk through of how a ritual is performed, artifacts are used, or some processes used to integrate a team with the rest of the organization.
- General discussions about the role they have within Scrum.
Anything can be demonstrated or reviewed and the attendees could concentrate on one ritual or artifact at a time or the overall process flow of product delivery. Learning about each others’ approaches may provide a handful of minor improvements resulting in dramatic positive changes.

For a sample meeting agenda, see Appendix B, “Scrum Masters’ / Product Owners’ Club Agenda - Sample”.

Sharing Team Learnings

As Scrum Teams complete sprints and have Sprint Retrospectives they will learn much about how to improve their own performance and interact with the entire organization to deliver products.

Some of the learnings will be shared across teams through Scrum Masters, Product Owners and the Developers but what’s also needed is the creation of a central repository of knowledge built up over time and available to any teams looking to address issues or adopt innovation.

Not everything discussed in a Sprint Retrospective is a candidate for inclusion in this knowledge-base but those topics that can be relevant outside the team should be. The simplest way to capture this is to summarize a problem and the corresponding solution. This is the same for both issues and innovation. Including examples and step-by-step instructions will complete the shared information. For example:

<table>
<thead>
<tr>
<th>Title: Report review takes too long during Sprint Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: When a Backlog Item results in a detailed artifact such as a client-summary report, having the Product Owner review the details during the Sprint Review takes too long and interrupts the flow of the meeting.</td>
</tr>
<tr>
<td>Resolution: During the sprint, as soon as detailed deliverables are ready for review they should be sent to the Product Owner who can provide feedback or accept the work as complete. During Sprint Review meetings then, the work will quickly be accepted and recorded as such.</td>
</tr>
</tbody>
</table>

These individual write-ups can be collected on the Scrum Resource Repository internal website and made available to all Scrum team
members. Announcing the latest additions or setting up alerting will help drive visitors to the website.

Ultimately the goal is to capture and share knowledge so that every team can benefit from the experience of others.

**Scrum Adoption Outside of Engineering**

As the transformation achieves more and more successes, people from other parts of the organization will take an interest in learning about Scrum and how it might be useful within their functional areas.

Assuming the initial Scrum training sessions were limited to those persons directly involved with the Scrum teams, a good way to spread the knowledge is to hold internal classes on the basics of Scrum and how the methodology or parts of it would be useful outside of the Software Engineering group. After explaining how the Scrum artifacts and rituals are used to create software products, they can be associated with more general work processes as a way to make the connection for the audience. For example:

- If a team wanted to organize their projects and assignments, use a Product Backlog to collect them all and sort them by their relative priority. If the term *product* doesn’t seem appropriate, then call it an *Initiative Backlog*.
- If a team wanted to plan execution of work and designate owners, define a sprint (2-4 weeks) and hold a Sprint Planning meeting. Build up a Sprint Backlog by selecting work from the Initiative Backlog and identify tasks, owners, and initial hours estimates.
- If a team wanted to track progress during the sprint, conduct Daily Standup meetings. If progress on initiatives is too slow for meaningful daily updates, have the meeting three times a week instead of five.
- If a team wants to demonstrate accomplished work at the end of the sprint, hold a Sprint Review meeting and invite stakeholders or other interested parties to see the progress.
- If a team wants to measure performance and improve process, have Sprint Retrospective meetings at the end of each sprint to evaluate and define improvements.
For those group that express an interest in adopting some aspects of Scrum, the more experienced Scrum Masters can help define the right level of process in conjunction with each team.
Chapter 8 - Measuring Success

Measuring Success

The beginning of the transformation from Waterfall to Scrum is fairly obvious, starting with the first Scrum Team working under the new process. Knowing when the transformation can be considered finished isn’t easily defined because Scrum is an ongoing process of continual improvement and not a final state to which an organization arrives at.

However such ambiguous answers won’t suffice within the typical software development organization because someone, usually higher up in the chain of command, will want to know when the initiative has been completed. Since there’s no hard definition of being “transformed”, defining and reporting on a set of milestones will satiate the need to see progress and provide an indication of success. Every organization will have specific goals to be achieved but in general the following should work for most:

- Composition of the Scrum Team and their area of focus.
- Templates for Scrum artifacts created.
- Initial Product Backlogs written to support two or three sprints.
- Coding and testing environments established.
- First Scrum Team completes Sprint 1.
- Additional Scrum Teams complete Sprint 1.
- Delivery of the first product under Scrum.
- All products in the pipeline now prepped for or underway with Scrum.

Each of the above are significant achievements as they all involve quite a bit of work and the coordination of many people across the organization. In addition to the milestones, there are other ways to measure success using some of the work completed at the beginning of the transformation.
Survey Resampling

Prior to the start of the transformation a survey was conducted to measure how the organization was performing in delivering products. It touched upon key areas such as the visibility people had of the overall strategy, the predictability of product delivery, the clarity of requirements, communication across groups, quality of products delivered, the sustainability of the overall process and productivity in terms of tools, processes and resources.

The survey results should have indicated areas of concern, or shortcomings that could potentially be addressed by the transformation. After the organization has been operating under Scrum for some time reissuing the survey will give a measure of what has changed and if any problem still needs to be resolved. Re-surveying the organization cannot be done too early as it is best to wait until at least all Scrum Teams have completed a few sprints or a couple of products have been delivered under the new methodology.

The survey can be issued a third time when it seems the major problems are solved and working under the new methodology has become more or less routine. Again, the goal is to identify large systemic problems that aren’t necessarily visibility from one team’s perspective. These larger issues can be addressed by the Transformation Oversight Group discussed in Chapter 6, “Inspection & Adaptation” or at a higher executive level where larger scale organization change is needed.

Product Delivery Workflow Revisited

Part of the initial transformation activities involved creating a Product Delivery Workflow in order to identify any shortcomings across the organization and to get a sense of what areas will need attention to operate better under the new methodology.

After a couple of products have been delivered using Scrum, a second review and update of the overall Product Delivery Workflow will help identify areas that still need attention under the new process. Any issues remaining or new ones can be funneled to the Transformation Oversight Group to discuss and drive the appropriate adjustments.
Signs of Team Success

While much attention will be on how well the organization is delivering products under Scrum it is important to pay attention to the individual Scrum Teams and how well they are performing and adapting to the new process. Specific signs point to a team that has successfully made the transformation.

- Communication amongst the team members is frequent, meaningful and collaborative in nature. The mentality of handing work “over the fence” has largely disappeared with Coders and Testers collaborating on designs to understand how the functionality will be delivered and coordinating on code handoffs and testing work.

- Information and issues are openly shared between the Developers (Coders, Testers, etc) and the Product Owner. No more is “Product” or “The Business” talked about as if a different group since they are all on the same team and responsible for delivering work at the end of each sprint.

- The pattern of work handoff between members looks like (code/test.. code/test.. code/test) versus (code, code, code/test, test, test) and the time between the (code/test) handoff is very short as the two roles coordinate on what is being delivered and when.

- The traditional roles (Coder, Tester, DBA, etc) start to blur as each team member helps others out. It will start at first with one being asked to help another during crunch time and will eventually change to each person completing work that crosses roles lines. For example, a Coder may prepare a testing environment or generate test data as part of their development effort and the Tester can make use of these right away.

- Team interaction becomes friendlier, more comfortable and the energy level of the team increases as they start working in a more synchronized manner. Joking and friendly banter occur spontaneously with all having a good laugh.

- The team is resilient when there are unexpected changes in priority or urgent issues requiring team attention threaten to derail or invalidate completed work.

- The team is openly self critical and people are comfortable calling out problems and offering solutions without blame for specific people. When the root cause was a result of one person’s failings, the issues are brought up and resolved with little hurt feelings as
everyone realizes that mistakes can be made and the focus is on solving the problem, not blaming anyone.

Over time a team that performs well can start to streamline some rituals such as Sprint Reviews or Retrospectives because they are so in tune with each other’s activities. Retrospection and adaptation occur naturally and spontaneously and the team’s rate of evolution increases dramatically.

**Signs of Organizational Success**

Like with the individual Scrum Teams, the organization as a whole can experience changes that indicate a successful transformation.

- Visibility into product development strategy and current status has improved across the organization. With the creation, maintenance and sharing of the overall product roadmaps and a consistent way of reporting on progress from sprint to sprint, everyone can benefit from the transparency a common and simple development process provides.
- There’s less of a divide between Product Management and Engineering where they talk less about each other as separate entities and more like partners working together.
- The business is more responsive to changes in the marketplace, client needs and addressing defects in a timely manner. With a short sprint cycle, the latest needs (enhancements or issues) will only wait a short time in the queue versus the team having to finish projects with current resources to work on something new. In addition, all the frustration that generally goes with negotiating the allocation of resources and changing priorities is eliminated. This improves the efficiency of everyone involved.
- There’s more awareness and avoidance of context switching and churn which saps energy and slows productivity overall.
- Risks can be spotted further out enabling the organization to mitigate them sooner and come up with better options.

The transformation to Scrum is never really finished as the teams and organization as a whole continually make incremental improvements over time. Effectively the “new” development process becomes invisible, like the plumbing or electricity in your house where it just works and people no longer think about it. At this point Scrum has become the
established process and the transformation process could be considered complete.
Backlog Item (Enhancement) - Sample

This sample backlog item (enhancement) is shown in this format to illustrate the essential elements and doesn’t imply that all items should be written this way. The structure of the Product Backlog will dictate the final format.

<table>
<thead>
<tr>
<th>Title</th>
<th>Current Weather Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Story</td>
<td>As a user, I want to have current weather conditions for my travel destination so I can get a sense of what it’ll be like for my trip.</td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>(1) The username data element must accept between two and 30 alpha characters and not contain spaces or escape characters.</td>
</tr>
<tr>
<td></td>
<td>(2) All submitted data elements must be converted to lowercase before processing and storage.</td>
</tr>
<tr>
<td>Story Points</td>
<td>5</td>
</tr>
<tr>
<td>Q &amp; A / Notes</td>
<td>Are you sure that you’d like to start with the destination city data element versus the date which may be helpful for lookups? =&gt; Yes, while both will eventually be needed, using the destination city is better for demonstrating to senior management in the early stages of this project.</td>
</tr>
</tbody>
</table>
Backlog Item (Defect) - Sample

This sample backlog item (defect) is shown in this format to illustrate the essential elements and doesn’t imply that all items should be written this way. The structure of the Product Backlog will dictate the final format.

Title:

DEFECT: Failed transactions aren’t be logged as error 902.

User Story:

===Problem Description===
The BillingTransaction table, ReturnCode field, isn’t being updated with code 902 for transactions failures. This will result in missing transaction report records.

===Replication Steps===
1. Enable the Client Gateway service on QA-Box#3.
2. Using the test client data, run the entire data set through the Client Gateway.
3. Observe the BillingTransaction table, ReturnCode field for no record of the failed transaction.

===Expected Results===
The BillingTransaction table, ReturnCode field should have a transaction code of 902 for the purchase attempt that failed.

===Actual Results===
The BillingTransaction table, ReturnCode field did not have a transaction code of 902 for the purchase attempt that failed.

Story Points:

3

Acceptance Criteria:

N/A
## Product Backlog - Sample

<table>
<thead>
<tr>
<th>Priority</th>
<th>Title:</th>
<th>User Story</th>
<th>Acceptance Criteria:</th>
<th>Q&amp;A / Notes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Current Weather Conditions</td>
<td>As a user, I want to have current weather conditions for my travel destination so I can get a sense of what it’ll be like for my trip.</td>
<td>(1) The username data element must have 2 and 30 alpha-numeric characters and not contain special characters or escape characters. (2) All submitted data elements must be converted to lowercase before processing and storage.</td>
<td>Are you sure that you’d like to require data elements? The data elements may already be lowercase, but the user might prefer to keep them uppercase for legibility. Which may be helpful for lookup? Yes, while both will eventually be needed, using the destination city is better for demonstrating the senior management in the early stages of this project.</td>
</tr>
<tr>
<td>2</td>
<td>App Requests Logging</td>
<td>As the system administrator want to track application requests logged in the database so I can run reports showing traffic activity.</td>
<td>(1) All data elements sent by an application must be logged. (2) Logged data elements must be accompanied by a database-generated timestamp.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Processing Requests Quickly</td>
<td>As the system administrator need to process all requests as quickly as possible.</td>
<td>Note: As of March 1, the product owner still researching SLAs in order to determine what the right values are.</td>
<td></td>
</tr>
</tbody>
</table>

**Switching to Scrum - William Patrick Swisher**
Definition of Done - Sample

[team name]

Definition of Done

Ratified by the Scrum Team: [date]
Approved by Management: [date]

The Definition of Done specifies the criteria by which sprint backlog items are considered Done. It guides decisions and activities around accomplishing sprint work.

Sprint backlog items are considered “Done” when:

1. Sprint backlog items are completed as defined by their Acceptance Criteria;
2. AND all coding is created according to the Engineering Organization’s standards;
3. AND all testing is completed according to the Engineering Organization’s standards;
4. AND all defects of severity level 3 and above are fixed and verified;
5. AND all documentation is updated;
6. AND IF the sprint work will be deployed to Production at the end of the sprint, a handoff meeting is held to turn over the code to Operations.
7. AND IF the sprint work will be deployed to Production at the end of the sprint, a Deployment & Support Plan will be written by the team and approved by a Engineering Manager.
Switching to Scrum - William Patrick Swisher

Sprint Backlog - Sample

<table>
<thead>
<tr>
<th>Day</th>
<th>Remaining Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

### Backlog Item / Tasks:

<table>
<thead>
<tr>
<th>Task</th>
<th>Owner</th>
<th>Initial Hours</th>
<th>Days</th>
<th>Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Current Weather Conditions: As a user, I want to have current weather conditions for my travel destination so I can get a sense of what it'll be like for my trip.</td>
<td>Jim</td>
<td>12</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>(2) Request Logging: As the system administrator, I want to have application request logging so I can run reports showing traffic activity.</td>
<td>Joe</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Create request XLS package.</td>
<td>John</td>
<td>9</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Create request submission process.</td>
<td>Jane</td>
<td>7</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Add error handling of malformed data.</td>
<td>Jamie</td>
<td>8</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Create database schema for application</td>
<td>Janet</td>
<td>9</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Write application request report.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create scheduled job to run reports and send via email.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Appendix A - Sample Artifacts

### Transformation Oversight Group - Topic Backlog - Sample

<table>
<thead>
<tr>
<th>Area Affected</th>
<th>Category</th>
<th>Urgency (1-5)</th>
<th>Issue / Innovation Description</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daily Standup</td>
<td>Personnel</td>
<td>1</td>
<td>The SVP of Sales has been attending the Daily Standup and interrupting the flow of the meeting with questions about functions and estimates directly of the Developers. The Scrum Master has asked him to hold off but seem uncomfortable to do it more forcefully.</td>
<td>The Development Manager will ask the VP of Engineering to intervene and talk with the SVP of Sales.</td>
</tr>
<tr>
<td></td>
<td>Process</td>
<td>4</td>
<td>There are a lot of meetings during each week for Developers and they are complaining about not having enough time to complete sprint work.</td>
<td>The Scrum Master by adding titles for backlog items that summarize each in a few words. This is helping each person find their place quickly.</td>
</tr>
<tr>
<td></td>
<td>General</td>
<td></td>
<td>The Scrum Master has improved Team-One’s sprint backlog by adding titles for backlog items that summarize each in a few words.</td>
<td>The PMO Manager will review the design and have all four Scrum Masters adopt the style for each of their team’s next sprints.</td>
</tr>
<tr>
<td></td>
<td>Tools</td>
<td>2</td>
<td>The procedures around requesting a new build from Configuration Management are designed to work within a Waterfall environment. The process of completing a form, having it signed, submitting it, and waiting for the approval meeting every Tuesday &amp; Thursday is taking so long that it blocks deployment at the end of some sprints.</td>
<td>The Development and QA Manager will meet with the Configuration Manager to define a new build request process that doesn’t involve so many approvals and can be turned around within a day.</td>
</tr>
<tr>
<td></td>
<td>Policy</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Logging & Reporting**

When clients use the product, a record of their transactions will be kept for technical support analysis, business reporting, and billing purposes.

<table>
<thead>
<tr>
<th>ID:</th>
<th>Requirement Directive Statements:</th>
</tr>
</thead>
</table>
| R22 | The project shall log the following data elements in the Transaction Logging database for **app requests**:  
• product-code, request-type, request-ID, time-stamp, destination-city, travel-dates |
| R23 | The project shall log the following data elements in the Transaction Logging database for **app responses**:  
• request-ID, time-stamp, response-value, error-code |
| R24 | The product shall use the database schema as defined in appendix F, “Weather Database Schema, version 3.5”. |
| R25 | The product shall produce a monthly transaction report based on the design in Appendix H, “Monthly Report Sample, version 2”. |
| R26 | The product shall produce a weekly transaction report based on the design in Appendix G, “Weekly Transaction Report Sample, version 3”. |
SUBJ: Sprint 9 - Daily Burndown Chart - Day 7

BODY:

Hello Everyone,

Here’s the latest chart for today. We’re coming in sooner than expected overall but will have to keep an eye on the remaining hours for testing as they are significant.

![](image)

Discussion Points:

- Edwin mentioned he’s out of tasks for the sprint and will spend some time writing unit tests and then look at upcoming product backlog items to design implementation.
- Louis will follow up on the request to IT regarding permissions on the deployment staging server.

Thanks,

Marie C.
Sprint Retrospective Summary - Sample

Sprint Retrospective - Sprint 9

Who did we appreciate and why?

- Dave Helpdesk, for opening up permissions to the testing servers on a Saturday so we could kick off testing first thing Monday morning.

What went well during the Sprint?

<table>
<thead>
<tr>
<th>#</th>
<th>Description:</th>
<th>How can we keep benefiting from this?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The testing script that removed the old build and installs the new saved a couple of hours.</td>
<td>Have the script modified to make it more general for team usage and distribute to everyone who could use it.</td>
</tr>
</tbody>
</table>

| ... | ... | ... |

What didn’t go well during the Sprint and how can we make it better?

<table>
<thead>
<tr>
<th>#</th>
<th>Description:</th>
<th>How can we make it better?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>It is taking up to fifteen minutes to check in code each time causing coder machines to lock up each time.</td>
<td>Work with the Configuration Management group to change the indexing from after each checkin to be a nightly job.</td>
</tr>
</tbody>
</table>

| ... | ... | ... |

Action Items:

- Carl will update her build deployment script with command line parameters and share it with the team.
### Weekly Management Report - Sample

<table>
<thead>
<tr>
<th>Product</th>
<th>Current Sprint Work</th>
<th>Next Sprint Work</th>
<th>Issues / Concerns</th>
</tr>
</thead>
</table>
| Weatherly iOS v1.0.0 | • Design App Gateway file format.  
• Create interface to accept app requests.  
• Create app request database.  
• Design reports.  
• Design & create databases for logging.  
• Document database schema design.  
• Finalize App Gateway request processing.  
• Create internal reports.  
• Move reporting database into production.  
• Hold training sessions for Customer Support. |                                                                                   | April 20: The Product Owner who was defining this has become very involved in another product and may not have enough time to define a proper Product Backlog in time. |
| Website Update   | Q3 2012 (est.)                                                                       |                                                                                 |                                                                                 |
| Weatherly Android v1.0.0 | Q2 2013 (est.)                                                                       |                                                                                 |                                                                                 |

Calculated Delivery Date: June 31, 2012
# Pre-Transformation Survey - Sample

The following survey should take less than five minutes to complete. Comments about lower scoring questions are encouraged and appreciated. The scoring scale represents: (1) Strongly Disagree, (2) Disagree, (3) Neutral, (4) Agree, (5) Strongly Agree

## General

<table>
<thead>
<tr>
<th>What department do you belong to? (Development, Quality Assurance, Product Management, Senior Management, etc)</th>
</tr>
</thead>
</table>

## Visibility

| I understand the overall strategy for the company. | 1 2 3 4 5 |
|---|
| I understand the strategy for the products I’m working on. | 1 2 3 4 5 |
| I feel informed about the status of products actively being developed. | 1 2 3 4 5 |

## Predictability

| Product releases are delivered close to the time of when they are estimated to be done. | 1 2 3 4 5 |
|---|
| Defects are fixed and delivered quickly. | 1 2 3 4 5 |
| Delivered products contain the features needed. | 1 2 3 4 5 |
| The frequency of product delivery is acceptable. | 1 2 3 4 5 |

## Clarity

| Product requirements are clear enough to complete work. | 1 2 3 4 5 |
|---|
| I know who to ask regarding requirement questions. | 1 2 3 4 5 |
| The standard practices for my department known by everyone in the department. | 1 2 3 4 5 |
## Communication

<table>
<thead>
<tr>
<th>Description</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication across departments is good enough. (e.g. Development to Product Management)</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Communication between roles is good enough. (e.g. Coders to Testers)</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Communication between individuals is good enough. (e.g. Coder to Coder, Tester to Tester)</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Handoffs of work between teams works well.</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

## Quality

<table>
<thead>
<tr>
<th>Description</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivered products have sufficient quality.</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

## Sustainability

<table>
<thead>
<tr>
<th>Description</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>I have enough time to complete my work each day.</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>I can work at a sustainable pace to complete projects.</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

## Productivity

<table>
<thead>
<tr>
<th>Description</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>The tools I use enable me to work efficiently and effectively.</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>The policies and processes within my department enable me to work efficiently and effectively.</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>People from outside my department provide help when needed.</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

## Comments:

...
## Appendix B

### Two-Week Sprint Events Calendar - Example

<table>
<thead>
<tr>
<th>Days</th>
<th>Typical Daily Activities</th>
</tr>
</thead>
</table>
| Day 1 (W) | **Sprint Planning Meeting**  
Backlog Item Task Breakdown  
Task Work Begins |
| Day 2 (R) | Backlog Item Task Breakdown  
Task Work Continues  
QA Planning |
| Day 3 (F) | Backlog Item Task Breakdown  
Task Work Continues  
QA Planning  
QA Testing |
| Day 4 (M) | Task Work Continues  
QA Testing  
Defects Addressed |
| Day 5 (T) | Task Work Continues  
QA Testing  
Defects Addressed  
Optional Story Time Meeting |
| Day 6 (W) | Task Work Continues  
QA Testing  
Defects Addressed |
| Day 7 (R) | Task Work Continues  
QA Testing  
Defects Addressed |
| Day 8 (F) | Task Work Continues  
QA Testing  
Defects Addressed  
Optional Story Time Meeting |
| Day 9 (M) | Task Work Wrapped Up  
QA Testing Verifying Defects  
Sprint Review Preparation |
| Day 10 (T) | **Sprint Review Preparation**  
Sprint Review Meeting  
Sprint Retrospective Meeting  
Story Time Meeting |
Daily Standup Meeting Agenda - Sample

<table>
<thead>
<tr>
<th>SUBJ: Daily Standup Meeting</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODY:</td>
</tr>
<tr>
<td>Hello Everyone,</td>
</tr>
<tr>
<td>In this meeting, the Scrum Team will gather at the same location and time each day to share their progress and what they plan to work on, declare any impediments hindering their progress, and provide estimates for the remaining hours on owned tasks.</td>
</tr>
<tr>
<td>The <strong>first 15 minutes</strong> are dedicated to each team member answering the following questions:</td>
</tr>
<tr>
<td>1. What did you work on since the last Daily Standup meeting?</td>
</tr>
<tr>
<td>2. What will you work on before the next Daily Standup meeting?</td>
</tr>
<tr>
<td>3. What impediments are blocking or potentially blocking your progress?</td>
</tr>
<tr>
<td>The <strong>second 15 minutes</strong> are reserved for any discussion topics and if we have nothing to address, we’ll adjourn.</td>
</tr>
<tr>
<td>This is a fast-paced and focused meeting; please come prepared to provide your updates to the team.</td>
</tr>
</tbody>
</table>
Sprint Planning Meeting Agenda - Sample

<table>
<thead>
<tr>
<th>SUBJ: Sprint [#] Planning Meeting</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODY:</td>
</tr>
<tr>
<td>Hello Everyone,</td>
</tr>
</tbody>
</table>

In this meeting the Product Owner will present the ordered Product Backlog to the Scrum Team. The team will select backlog items to include in the upcoming sprint according to their assessed capacity during the sprint.

The end result will be a Sprint Backlog the team commits to completing by the end of the sprint according to the established Definition of Done.

Process:

1. **Sprint Capacity Assessment** - The Developers review their work capacity in the upcoming Sprint, taking into consideration, holidays, vacations, sharing of resources, production support issues, etc. This assessment helps frame the amount and type of work the team can commit to completing.

2. **Product Backlog Item Review & Commitment** - The Scrum Team starts at the top of the Product Backlog, reviewing each backlog item and discussing what’s involved to complete it, who the resources would be, any dependencies and asking clarifying questions of the Product Owner as needed. If the team feels they can commit to completing the backlog item, it is officially selected for the sprint. Then, the next backlog item is reviewed, discussed and committed to until the team feels they have reached their capacity for the sprint.

3. **Sprint Execution Strategy** - The Scrum Team will create a high level plan of key events in the sprint such as handoff of code for testing, merges, builds, etc for the purpose of coordinating work between team members.
Sprint Review Meeting Agenda - Sample

**SUBJ:** Sprint [#] Review Meeting

**BODY:**

Hello Everyone,

In this meeting, the Scrum Team will demonstrate completed backlog items to the Product Owner for evaluation according to the acceptance criteria. The Product Owner will then either *accept* or *reject* the work with any necessary feedback. The end result will be an official listing of backlog items completed and marked as such within the Product Backlog.

All observers are welcome but asked to allow the Sprint Review process to proceed with minimal interruption. Time for more extensive feedback and questions will be made available at the end.

**Process:**

1. **Sprint Capacity Summary** - The Scrum Master summarizes the capacity the team had during the sprint, noting any challenges that affected the performance of the team and completion of sprint work.

2. **Review of Backlog Items** - The Scrum Master reviews the list of backlog items accepted into the sprint along with their story point estimates. This includes indicating which backlog items were added/dropped from the sprint after the Sprint Planning Meeting concluded.

3. **Backlog Item Demonstrations** - Individual team members demonstrate completed backlog items, providing context where needed. The Scrum Master will ask the Product Owner if they *accept* or *reject* the work.

4. **Summary of Backlog Items Disposition** - The Scrum Master summarizes end results confirming with all parties which backlog items were *accepted* or *rejected* and why.
Hello Everyone,

In this meeting, the Scrum Team will review the events of the closing sprint and their performance, both individually and as a team.

The end result will be a document summarizing the discussion points that will be stored in the project archive and distributed to any interested stakeholders.

Only Scrum Team members are invited to this meeting to encourage open and honest discussions.

Process:

1. **Appreciations** - The team identifies who helped the team accomplish the sprint work and why. This can include people inside and outside the team.

2. **What went well?** - The team identifies what events, process, or people’s contribution helped the team accomplish their work. New standard procedures or process are agreed to based on what is discussed.

3. **What didn’t go well and how can we make it better?** - The team identifies what events, process, or people’s performance hindered the team during the sprint. New guidelines or process are agreed to based on what is discussed.

4. **Action Item Review** - The Scrum Master summarizes the specific action items and their owners identified during the discussion.
Story Time Meeting Agenda - Sample

SUBJ: Story Time Meeting

BODY:

Hello Everyone,

In this meeting, the Scrum Team will review upcoming backlog items and become familiar with what the Product Owner will want built.

Process:

1. The Product Owner will present and summarize individual backlog items one at a time starting with those with the highest business priority.
2. The Developers will seek to understand, asking questions, and providing feedback to improve how each backlog item is written and/or suggest alternative ideas. Clarifications or new details should be documented within the backlog item for future reference.
3. The Developers will identify any external resources or dependencies so they can be coordinated by the Scrum Master.
4. The Developers will provide relative effort estimates for each backlog item they feel they have enough understanding of.
5. The Developers will review as many as possible within the meeting time without compromising the learning process.
Hello Everyone,

This meeting is for the participants to share experiences, learn from each other and define best practices for their respective roles within the Scrum methodology.

**Agenda:**

1. Sharing of recent **issues**, their impact and resolution.
2. Sharing of recent **innovation** and the benefits.
3. Walk through of how rituals are performed, artifacts are used, or processes executed to identify improvements or synchronize approaches.
4. Open topics for discussion.
Daily Standup Meeting - Process Diagram

The Scrum Team meets in the same location and time each day. The Scrum Master welcomes everyone and signals the start of the meeting.

The Developers answer three questions:
(1) What did you work on since the last Daily Standup?  
(2) What will you work on before the next Daily Standup?  
(3) What impediments or potential impediments are inhibiting your progress?

The Product Owner is offered an opportunity to ask questions or provide updates.

Observers are offered an opportunity to ask questions or provide updates.

The Developers update the remaining hours estimates on owned tasks.
Sprint Planning Meeting - Process Diagram

The Scrum Team meets on the first day of the sprint, as early in the day as possible. The Product Owner shows the top backlog items from the Product Backlog which is ordered according to business needs.

The Developers review their work capacity for the sprint, taking into consideration: holidays, vacations, sharing of resources, product support issues, etc so they have an accurate sense of how much time they have available.

The Developers review, discuss, & clarify the top backlog item and when satisfied they can complete it within the sprint, officially select it for the sprint. Each backlog item is broken down into tasks, assigned owners, and given an initial effort estimate in hours.

After each backlog item is selected for the sprint, the Developers assess their remaining capacity and if available continue to add backlog items to the sprint.

The Developers create a high level timeline of the deliverables between team members, key events, and dependencies as a strategy of how the sprint work will be accomplished.
The Scrum Team assembles on the last day of the sprint as late in the day as possible. The Scrum Master hosts the meeting and if needed, explains to any stakeholders how the ritual works.

Depending on the level of explanation required, the Scrum Master presents slides explaining the release & sprint goals, resources capacity, the burndown chart along with reasons for not reaching zero hours, and the challenges the team encountered during the sprint.

The Developers demonstrate backlog items, one at a time or in groups identifying each so the Product Owner can follow along. After each is demonstrated, the Scrum Master asks if there are any questions. If not, then the Product Owner is asked “Do you accept this/these backlog item(s)?”

The Scrum Master summarizes the disposition of the backlog items, noting any pending decisions or action items for the Scrum Team.
Sprint Retrospective Meeting - Process Diagram

The Scrum Team meets on the last day of the sprint, immediately after the Sprint Review meeting. Only the Scrum Team members are allowed to attend as outsiders would dampen honest communication regarding team performance.

The Scrum Team identifies individuals, on or outside the team that helped accomplish work during the sprint. The Scrum Master shares the team’s gratitude with those individuals after the meeting.

The Scrum Team discusses new techniques, processes, or tools discovered that helped the team accomplish sprint work. Decisions regarding how to continue benefiting from the discoveries are made and agree to by the entire team.

The Scrum Team discusses issues of any type that inhibited sprint work and how to minimize or eliminate it from future sprints. The Scrum Master should ask questions to start discussions especially for topics the team is reluctant to discuss.

The Scrum Master records and shares any issues with outsiders that can help solve them. Additionally, any innovation that other teams may benefit from are shared to promote learning and accelerated transformation to Scrum.
Story Time Meeting - Process Diagram

The Scrum Team meets on the last day of the sprint, immediately after the Sprint Retrospective and when prudent, at other times during the sprint. The Product Owner brings the most recent Product Backlog ordered according to business value.

The Product Owner introduces upcoming backlog items, providing context and explanation of acceptance criteria.

The Scrum Team discusses each backlog item, asking questions to clarify understanding and provide feedback regarding the scope, dependencies, and design considerations. The Product Owner answers questions which are recorded in the backlog item's notes or integrated into the acceptance criteria.

The Scrum Team assigns relative effort estimates (T-Shirt Sizes, Story Points, etc) once they have an understanding of each backlog item.
Backlog Item States - Flow Diagram

Backlog items are initially marked as **New** when added to the Product Backlog and have not yet been reviewed by the Scrum Team. The items near the top of the Product Backlog should be complete enough for the team to review during Sprint Planning or Story Time meetings.

Backlog items are changed to the **Ready** state after the Scrum Team has the opportunity to review them, ask questions, make suggestions, and assign a relative effort estimate such as story points. The backlog item is now ready for selection into a sprint during a Sprint Planning meeting.

Backlog items are switched to the **In Progress** state once they have been selected into a sprint. This state is used regardless of if the item is being actively worked on or not.

Backlog items reach the **Completed** state when the Developers have completed all work in accordance with the Definition of Done and are ready to demonstrate it to the Product Owner.

Backlog items are marked as **Accepted** once the Product Owner reviews the work and acknowledges that it meets the acceptance criteria. If a backlog item is rejected by the Product Owner the state changes back to New or Ready depending on the clarification needed before it can be selected for another sprint.
SWITCHING TO SCRUM

Using Scrum for software product development is becoming increasingly popular and while the methodology is straightforward, the hardest part is making the procedural and cultural shift away from Waterfall, or traditional software development techniques. This book provides a guide on how to prepare an organization for the transformation to Scrum starting with making key decisions, messaging, preparing for the first team, adding teams, monitoring progress, and measuring success. Numerous sample templates and diagrams are included to jump start any organization’s efforts to make this shift successful!

Topics covered in this book include:

- Making key decisions and organizational messaging
- Utilizing existing business requirements for Scrum development
- Creating a Product Backlog, writing backlog items and managing defects
- Preparing for the first Scrum Team and project
- Initiating software development with Scrum
- Monitoring the transformation process and making adjustments
- Expanding the number of Scrum Teams
- Measuring transformation success
- Includes many templates & sample artifacts to kickstart the transformation

About the Author:

William Patrick Swisher has worked for over 15 years in various technology fields including Technical Support, Customer Service, Database Operations, Product Management, and of course Project Management. With a bachelor’s degree in Computer Science and Masters in Psychology, he is comfortable dealing with technology, systems and process but has the ability to understand how people fit into the equation and delights in helping others succeed.