S
C
R
U
M

Professional Scrum Developer Training

Optimizing Teams One Sprint at a Time

Microsoft® Visual Studio®

Accentient™

Scrum.org™

Accentient™

ACCENTIENT EDUCATION SERIES

Committed to training success

+1 (877) 710-0841 ■ www.accentient.com

Professional Scrum Developer Training

Course Number:	**PSD**
Version:	**2.1**
For software version:	**2010**

Images and Artwork

Acknowledgements

This course would not be possible without the help and input from the following people:

- Ken Schwaber
- Sam Guckenheimer
- Kelly Calvert
- Michael Vincent
- Dan Massey
- Matthew Hughes
- Jason Dean
- Adam Cogan

All trademarks referenced are the property of their respective owners

---

Disclaimer

Accentient

- A leader in Visual Studio ALM knowledge
- Has helped hundreds of clients understand, implement and leverage Team Foundation Server
- Has a close working relationship with Microsoft
- Has a staff consisting of Microsoft MVPs, Microsoft Regional Directors, Certified Professional Scrum Developers, Certified Scrum Professionals and Microsoft Certified Trainers

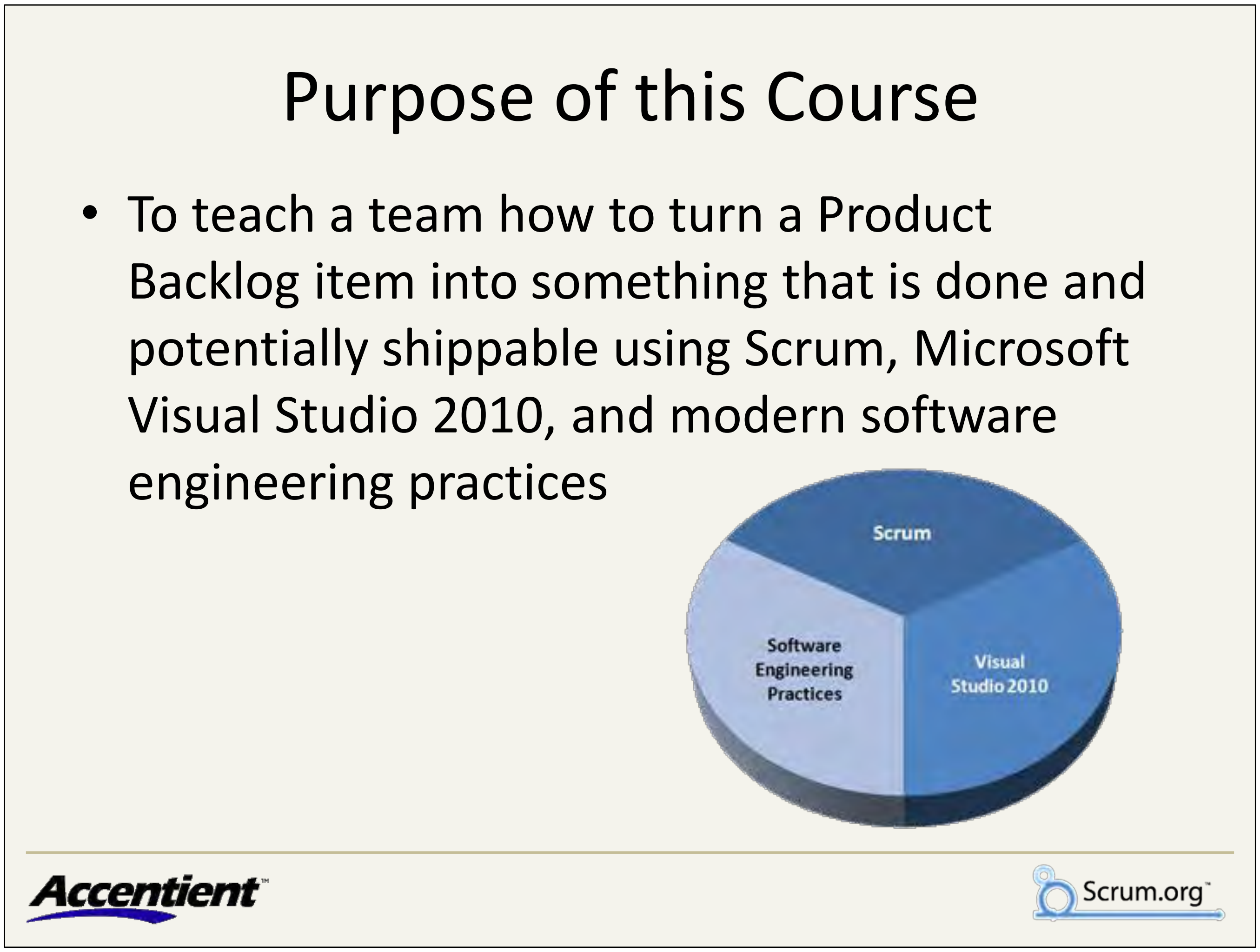
www.accentient.com

Richard Hundhausen

- President of Accentient
- Author of software development books
- Microsoft Regional Director
- Microsoft MVP (Visual Studio ALM)
- MCT, MCSD, MCDBA
- richard@accentient.com

Microsoft® Regional Director PROGRAM

Microsoft® MVP Most Valuable Professional



Purpose of this Course

- To teach a team how to turn a Product Backlog item into something that is done and potentially shippable using Scrum, Microsoft Visual Studio 2010, and modern software engineering practices

Scrum
Software Engineering Practices
Visual Studio 2010

Accentient™ Scrum.org™

How to get the most out of this course

For starters, you should attend the *entire* course, from beginning to end. There are no days or modules that are "okay to skip". You need to see the material in its entirety and participate with your team in all of the activities. The Professional Scrum Developer course is designed to be taken as a whole. Students who miss modules will miss key parts to the story.

Interspersed with the individual skills activities are particular skills and activities which test an individual's ability to work within a team environment. These activities cover technical, tooling and "soft skills" perspectives. Part of this course is a case study, where you, working as part of a team, will need to deliver several increments of functionality through to production quality completion, including review and retrospective.

You should also be able to inspect and adapt at all times. This includes your ability to improve on trouble areas, reflect on your ability to usefully consume constructive criticism and process and utilize information under pressure.

Repetition grows muscle memory

Remember the movie "The Karate Kid" (http://imdb.to/iWaz)? In the movie, the ‘kid did not understand the value of "wax on, wax off" until he saw it applied in a different context. The simplicity of the physical motions obscured the value of developing the muscle memory needed for actual combat.

The same goes for this course. The planning, execution, review, and retrospective cycle are your “wax on, wax off” movements. Later, when your actual team gets into software combat, these memorized motions will serve you well.

Finally: training, assessment, and certification for the *team* ...

THE PROFESSIONAL SCRUM DEVELOPER PROGRAM

Professional Scrum Developer

a training and certification program by SCRUM.ORG™

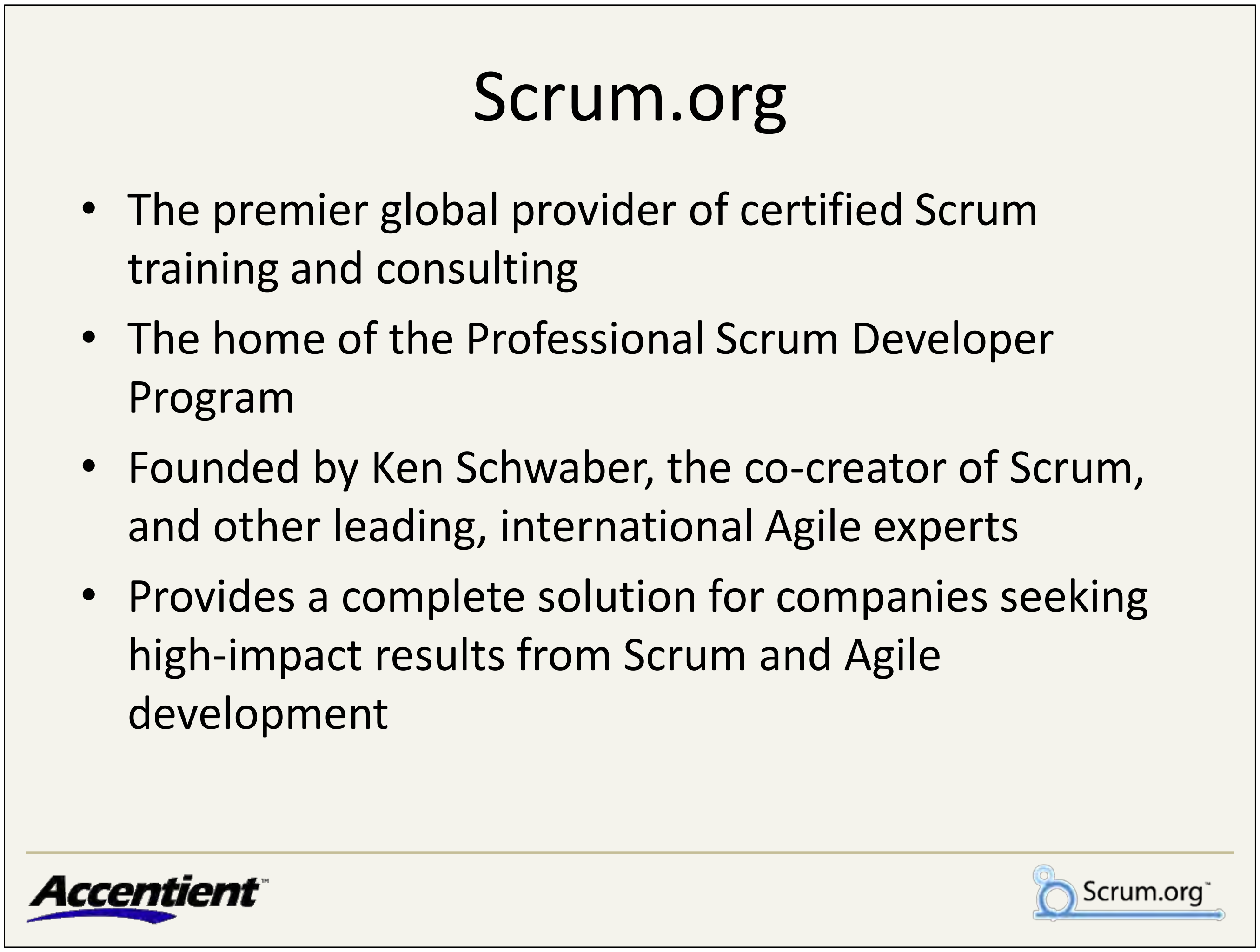
Accentient™ Scrum.org™

Professional Scrum Developer program

The Professional Scrum Developer program trains, assesses, and certifies Scrum developers working on a specific technology platform. The program includes a five-day course, an online assessment, and an industry-recognized certification. Each Professional Scrum Developer program targets a specific technology stack. At present, Scrum.org offers programs for Microsoft .NET and for Java.

The Professional Scrum Developer Program is a unique offering for a historically neglected role. The program teaches Scrum directly to the people doing the work. It extends the foundation of Scrum by interleaving industry best practices for software development. The classroom environment simulates real enterprise ALM, leading student teams from “requirements” to “release”. The SD program addresses real-world challenges faced by development teams. It incorporates “soft-skills” typically avoided in technical training, but which have become critical to the success of development teams today.

For more information watch this video of Ken Schwaber and Sam Guckenheimer discussing the Professional Scrum Developer program: http://bit.ly/fNYijc.



Scrum.org

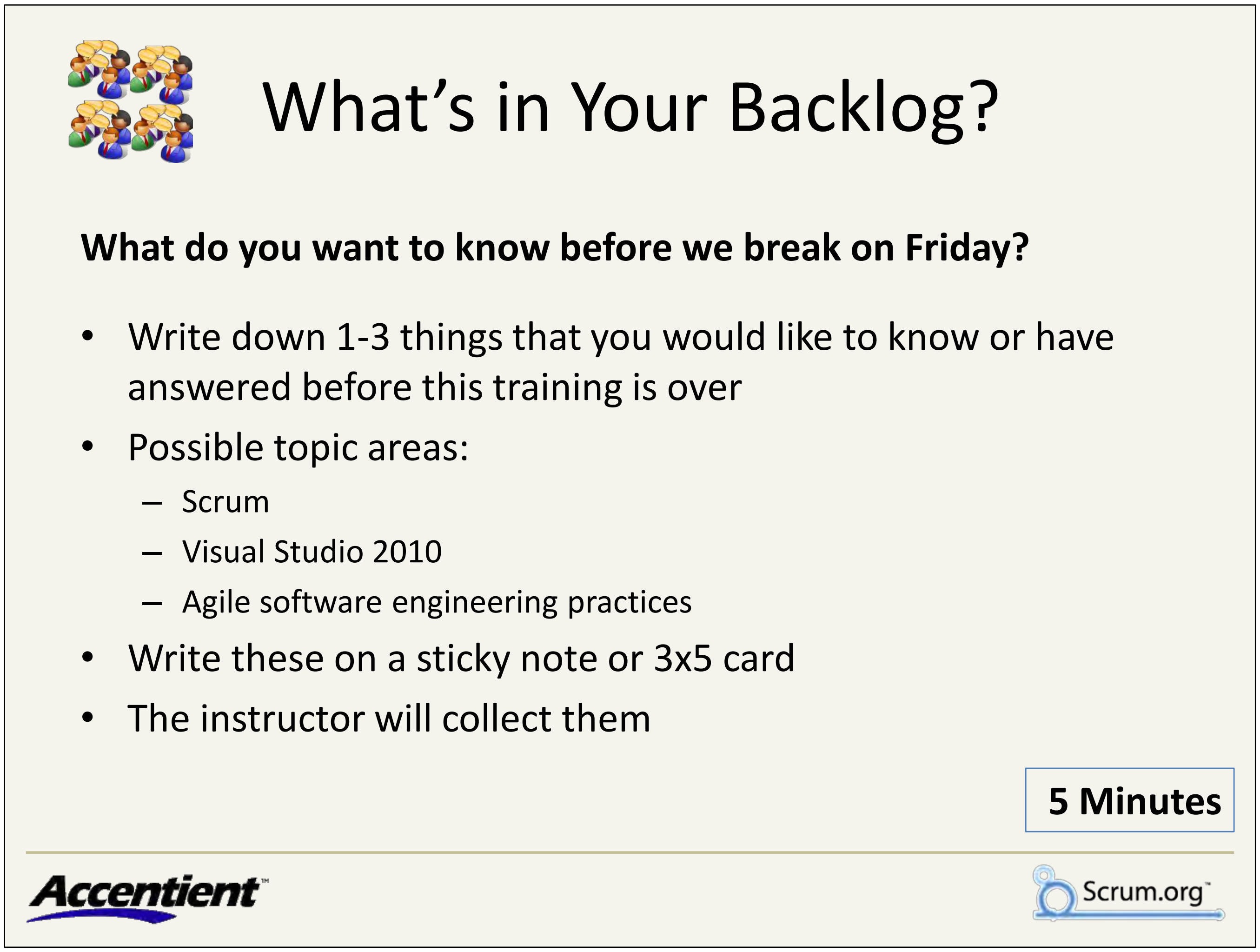
- The premier global provider of certified Scrum training and consulting
- The home of the Professional Scrum Developer Program
- Founded by Ken Schwaber, the co-creator of Scrum, and other leading, international Agile experts
- Provides a complete solution for companies seeking high-impact results from Scrum and Agile development

Scrum.org

Scrum.org was founded by Ken Schwaber, one of the creators of Scrum, in response to the emergence of the “Flaccid Scrum" phenomenon. Schwaber initially attempted to develop Scrum.org’s programs while part of the Scrum Alliance, an organization that he founded and managed for more than a decade. Ultimately, he found that to accomplish his goal of improving Scrum knowledge, training, and implementations, he would need to break away from the vested interests that had come to dominate the Scrum Alliance. This is what he and others have accomplished through Scrum.org.

Scrum.org is a consortium of people and organizations interested in improving the profession of software development. We work together in a emergent manner that uses frequent inspection (assessment) and adaptation (improvement and removal) to improve quality.

For more information about the origins of Scrum.org: http://bit.ly/dnK7px.



What's in Your Backlog?

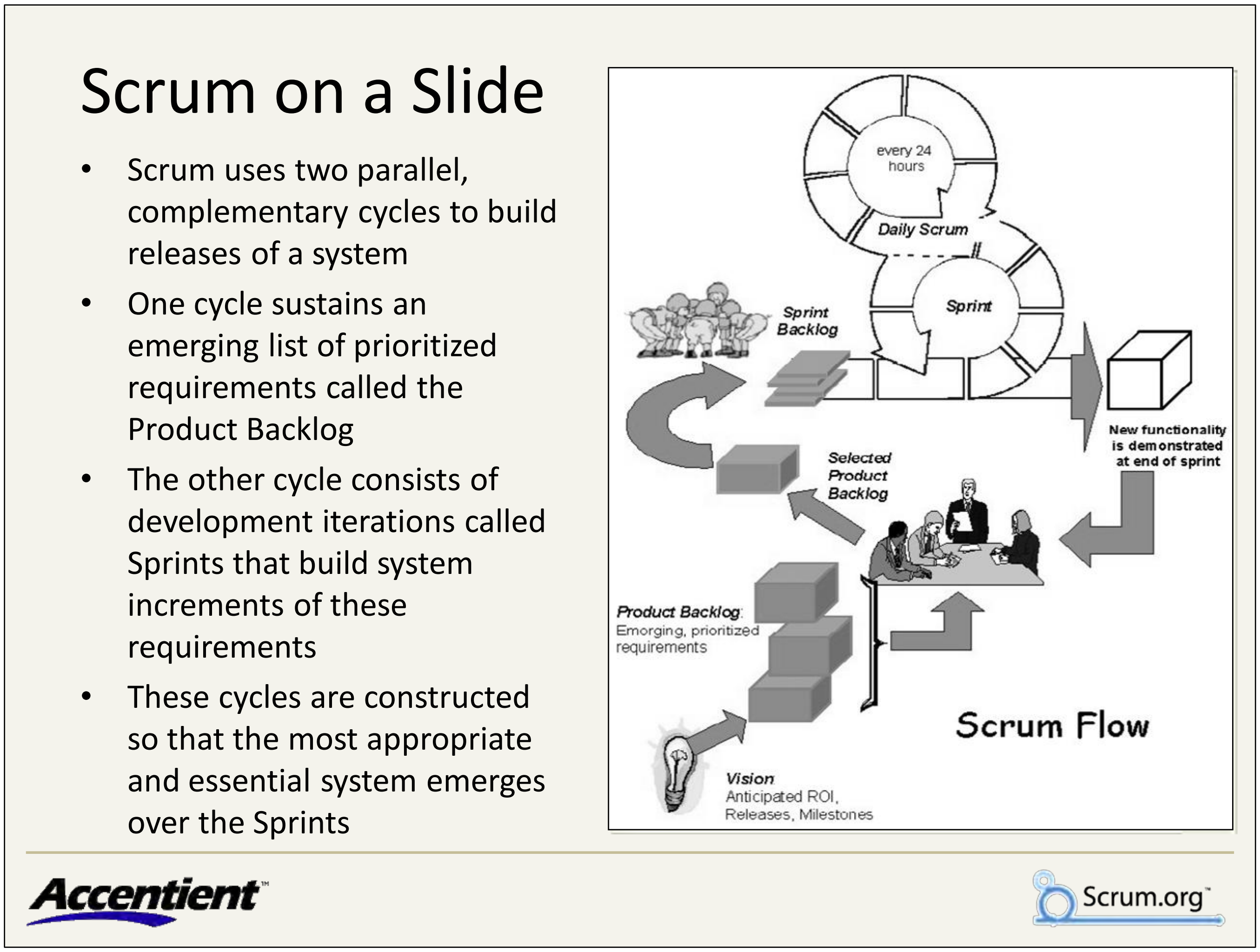
What do you want to know before we break on Friday?

- Write down 1-3 things that you would like to know or have answered before this training is over
- Possible topic areas:
  - Scrum
  - Visual Studio 2010
  - Agile software engineering practices
- Write these on a sticky note or 3x5 card
- The instructor will collect them

5 Minutes

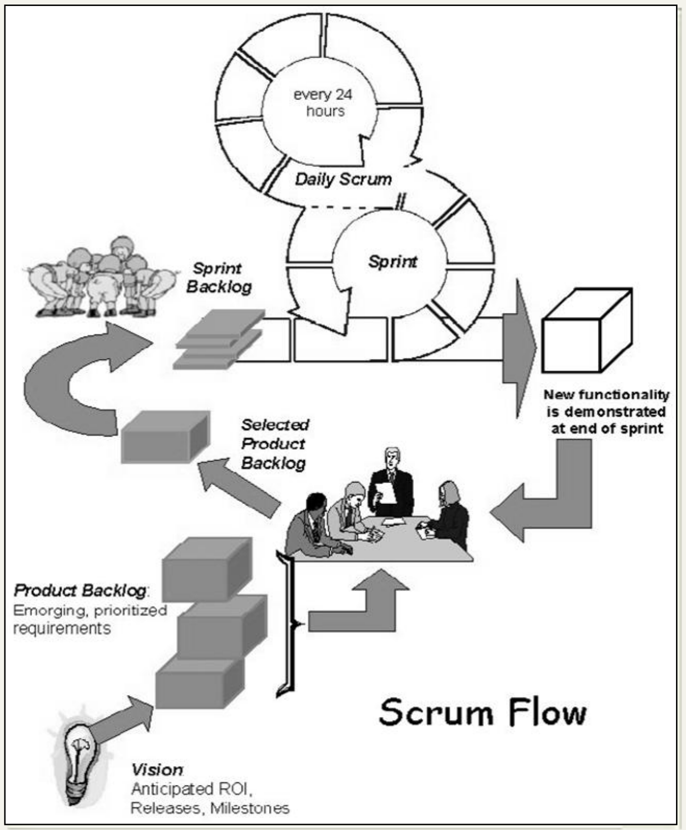
Accentient™ Scrum.org™

Activity: what's in your backlog?



Scrum on a Slide

- Scrum uses two parallel, complementary cycles to build releases of a system
- One cycle sustains an emerging list of prioritized requirements called the Product Backlog
- The other cycle consists of development iterations called Sprints that build system increments of these requirements
- These cycles are constructed so that the most appropriate and essential system emerges over the Sprints


every 24 hours
Daily Scrum
Sprint
Sprint Backlog
New functionality is demonstrated at end of sprint
Selected Product Backlog
Product Backlog
Emerging, prioritized requirements
Scrum Flow
Vision
Anticipated ROI, Releases, Milestones

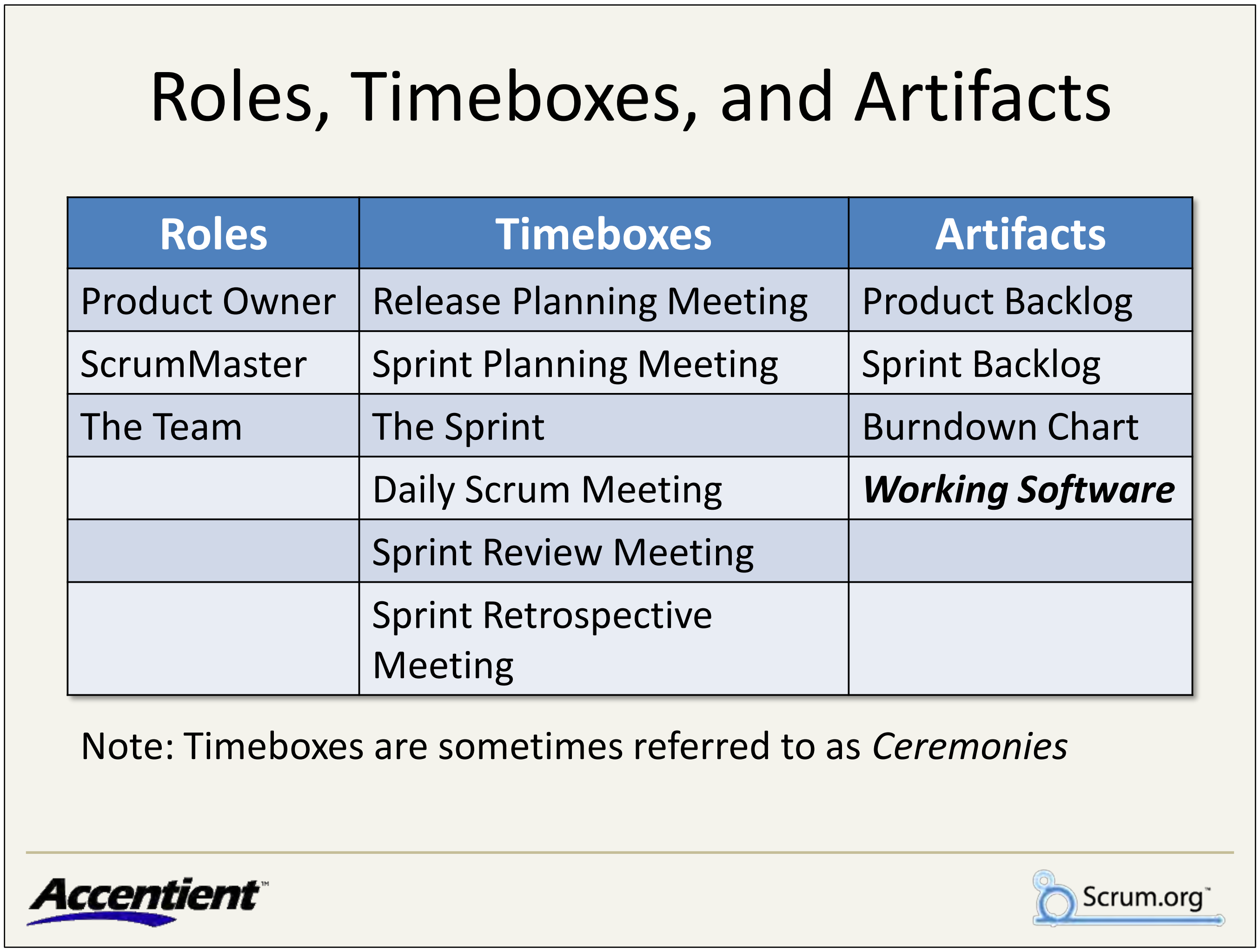
Accentient™ Scrum.org™

Scrum at a glance

- Is an empirical management and control process
- Has inspect and adapt feedback loops
- Has been successfully used to manage complex projects since 1990
- Delivers potentially shippable functionality in 2-4 weeks
- Is scalable to distributed, large and long projects
- Is CMM Level 3 and ISO 9001 compliant
- IS CMM Level 5 friendly
- Is extremely simple, but very hard because it is not a prescriptive process

Scrum provides the project delivery approach

- Improves communications and maximizes co-operation
- Incremental & Iterative – 2 to 4 week iterations delivering production quality code
- Regular input and feedback from users/business
- Continual focus on highest priority business requirements
- Analysis, design, development, testing performed throughout iteration
- Self-organizing cross-functional teams - including the customer
- Minimal creation of “Interim” documents - focus on working software
- Inspect & Adapt at the end of every iteration



Roles, Timeboxes, and Artifacts

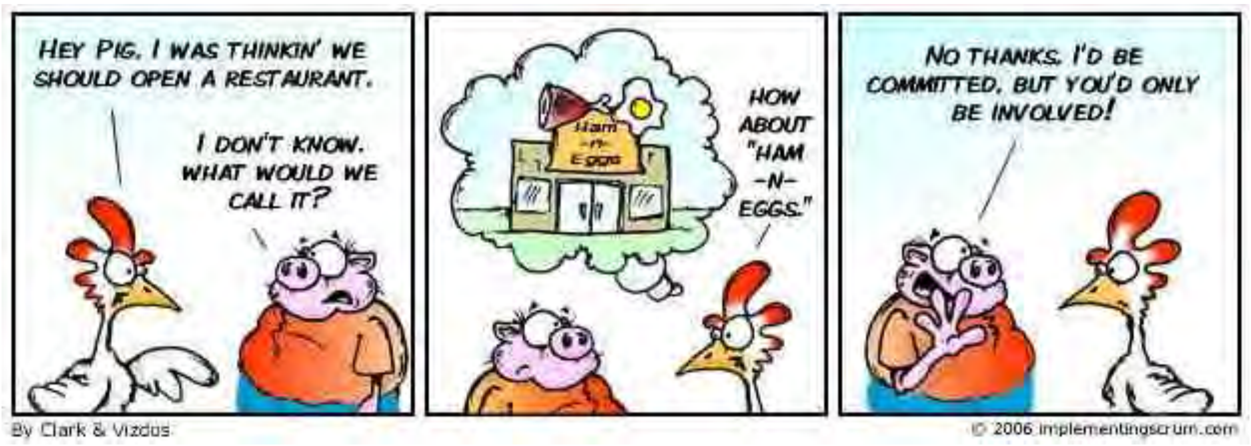
Roles	Timeboxes	Artifacts
Product Owner	Release Planning Meeting	Product Backlog
ScrumMaster	Sprint Planning Meeting	Sprint Backlog
The Team	The Sprint	Burndown Chart
	Daily Scrum Meeting	***Working Software***
	Sprint Review Meeting	
	Sprint Retrospective Meeting	

Note: Timeboxes are sometimes referred to as *Ceremonies*

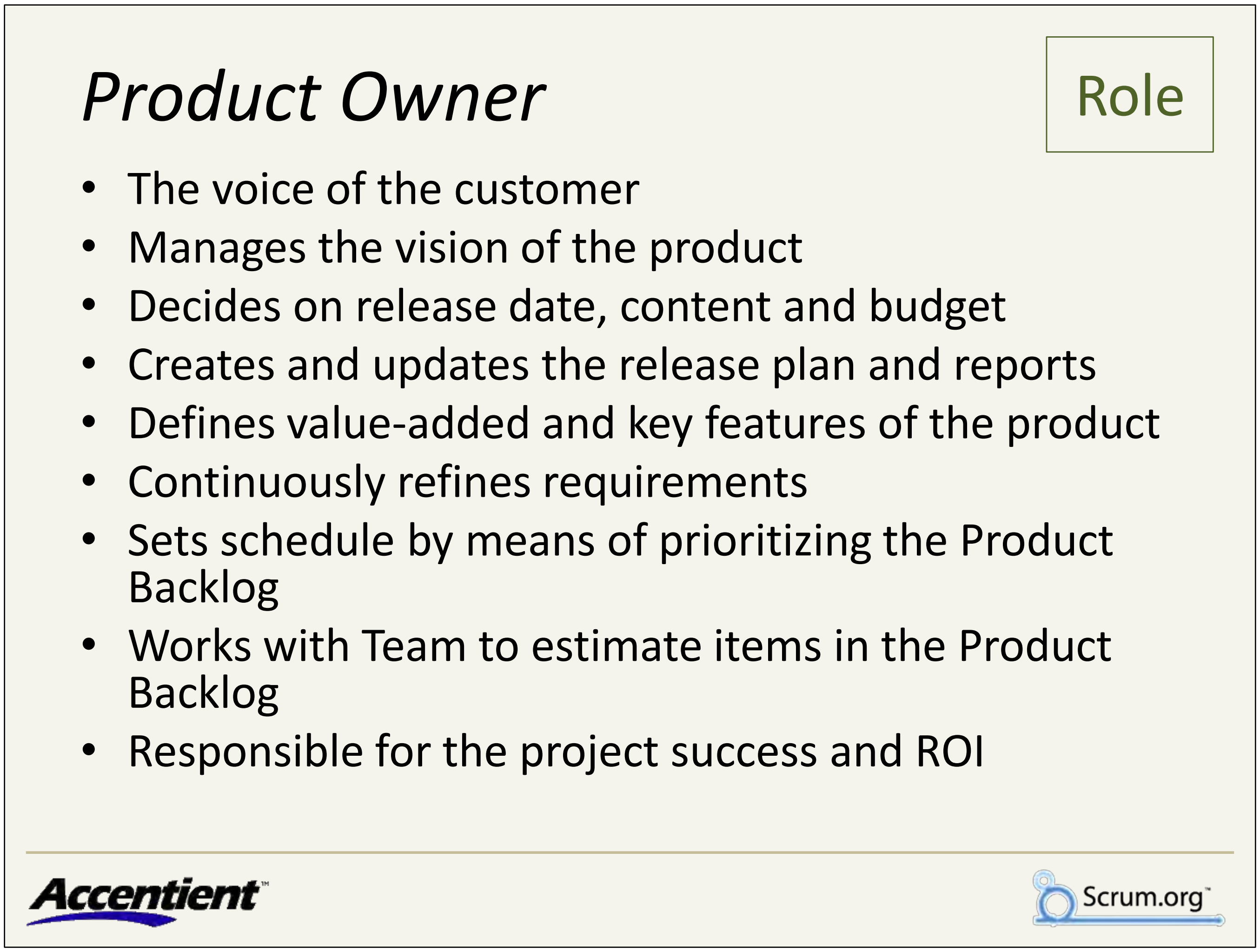
Accentient™ Scrum.org™

Pigs and chickens

The Scrum Team consists of the ScrumMaster, the Product Owner, and the team. Scrum Team members are called "pigs." Everyone else is a "chicken." Chickens cannot tell "pigs" how to do their work. Chickens and pigs come from the story:


HEY PIG. I WAS THINKIN' WE SHOULD OPEN A RESTAURANT.
I DON'T KNOW. WHAT WOULD WE CALL IT?
Ham -n- Eggs
HOW ABOUT "HAM -N- EGGS."
NO THANKS. I'D BE COMMITTED, BUT YOU'D ONLY BE INVOLVED!
By Clark & Vizdos
© 2006 implementingscrum.com

Members of the Scrum Team are known as Pigs because they are committed to delivering Sprint Goal. People who are involved but not dedicated to the project are known as Chickens. They can attend Scrum meetings but only as observers.



Product Owner

Role

- The voice of the customer
- Manages the vision of the product
- Decides on release date, content and budget
- Creates and updates the release plan and reports
- Defines value-added and key features of the product
- Continuously refines requirements
- Sets schedule by means of prioritizing the Product Backlog
- Works with Team to estimate items in the Product Backlog
- Responsible for the project success and ROI

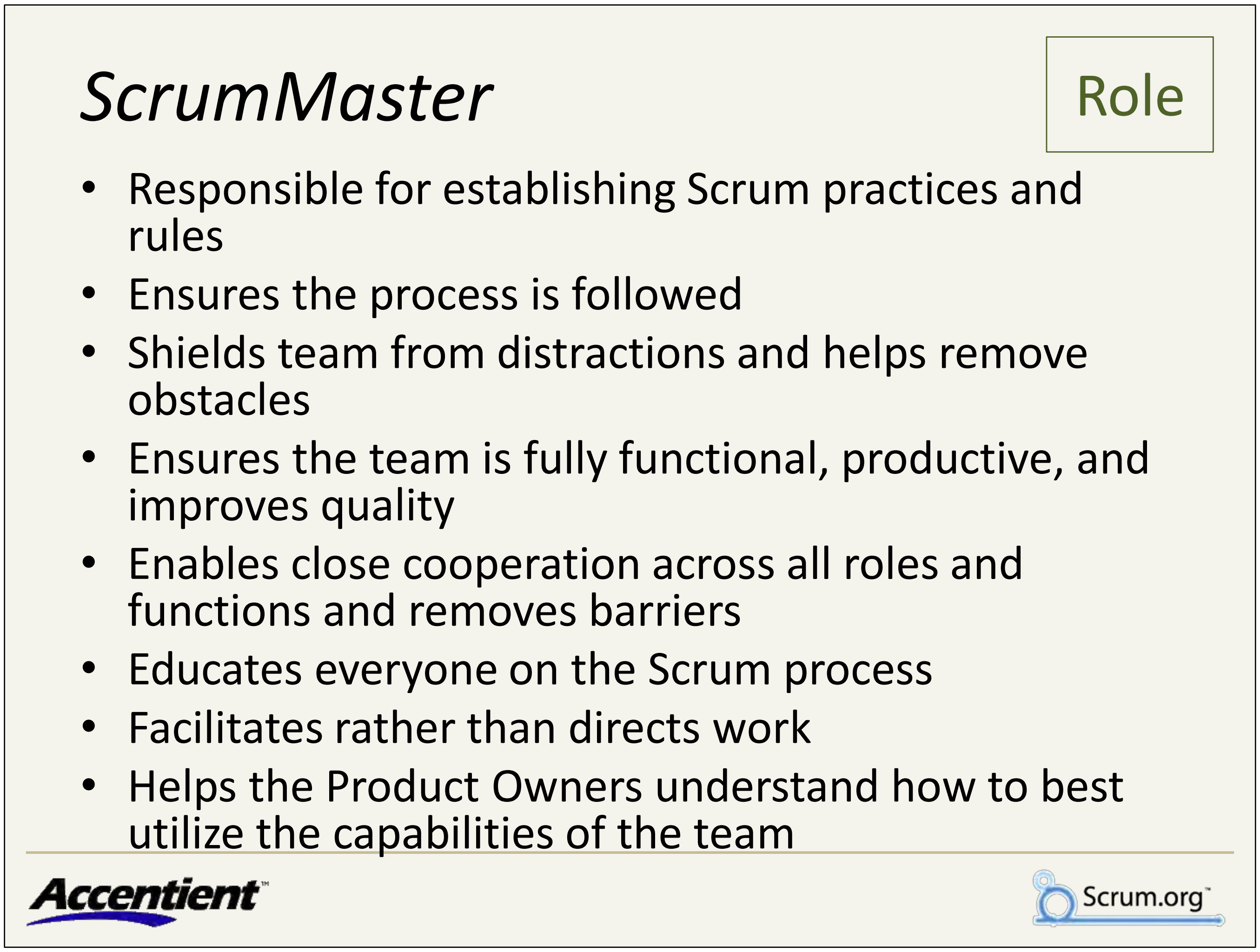
The Product Owner

The Product Owner is the one and only person responsible for managing the Product Backlog and ensuring the value of the work the team performs. This person maintains the Product Backlog and ensures that it is visible to everyone. Everyone knows what user stories and bugs have the highest priority, so everyone knows what will be worked on. The Product Owner is one person, not a committee. Committees may advise or influence this person, but people who want to change an item's priority have to convince the Product Owner. Companies that adopt Scrum may find that it influences their methods for setting priorities and requirements over time.

For the Product Owner to succeed, everyone in the organization has to respect his or her decisions. No one is allowed to tell the team to work from a different set of priorities, and teams aren't allowed to listen to anyone who says otherwise. The Product Owner's decisions are visible in the content and prioritization of the Product Backlog. This visibility requires the Product Owner to do his or her best, and it makes the role of Product Owner both a demanding and a rewarding one.

Manage the vision

The Product Owner establishes, nurtures and communicates the product vision. He or she achieves initial and on-going funding for the project by creating initial release plans and the initial Product Backlog.



ScrumMaster

Role

- Responsible for establishing Scrum practices and rules
- Ensures the process is followed
- Shields team from distractions and helps remove obstacles
- Ensures the team is fully functional, productive, and improves quality
- Enables close cooperation across all roles and functions and removes barriers
- Educates everyone on the Scrum process
- Facilitates rather than directs work
- Helps the Product Owners understand how to best utilize the capabilities of the team

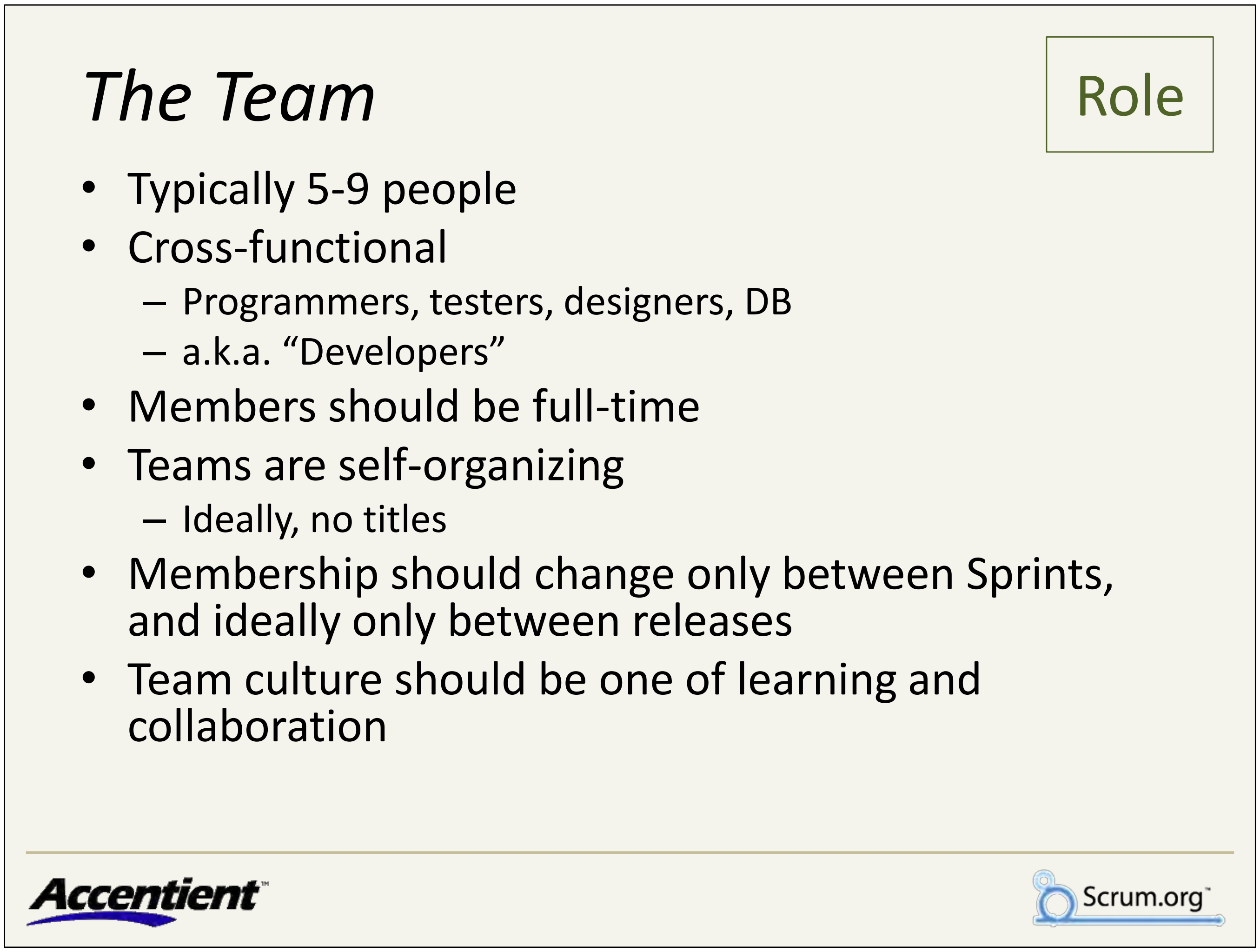
Accentient™ Scrum.org™

The ScrumMaster

The ScrumMaster is responsible for ensuring that the team adheres to Scrum values, practices, and rules. The ScrumMaster helps the team and the organization adopt Scrum. The ScrumMaster teaches the team by coaching and by leading it to be more productive and produce higher quality products. The ScrumMaster helps the team understand and use self-management and cross-functionality. However, the ScrumMaster does not manage the team – the team is self-organizing.

The ScrumMaster works with the customers and management to identify and train a Product Owner. The ScrumMaster teaches the Product Owner how to do his or her job. Product Owners are expected to know how to manage to optimize value using Scrum. If they don't, the ScrumMaster is held accountable.

As a ScrumMaster, you will have to contend with the tyranny of the waterfall, the illusion of command and control and the era of opacity. Adopting Scrum puts stress on the team and organization, exposing underlying problems and limitations. The ScrumMaster's job is to prioritize these problems and help the organization overcome them to get better at software development, managing software investments, and becoming a better community to work in. When equipped with a resolute, patient ScrumMaster, Scrum can be used to transform software development into a profession, projects into valuable endeavors, and development organizations into communities that people look forward to working in.



The Team

Role

- Typically 5-9 people
- Cross-functional
  - Programmers, testers, designers, DB
  - a.k.a. “Developers”
- Members should be full-time
- Teams are self-organizing
  - Ideally, no titles
- Membership should change only between Sprints, and ideally only between releases
- Team culture should be one of learning and collaboration

Accentient™ Scrum.org™

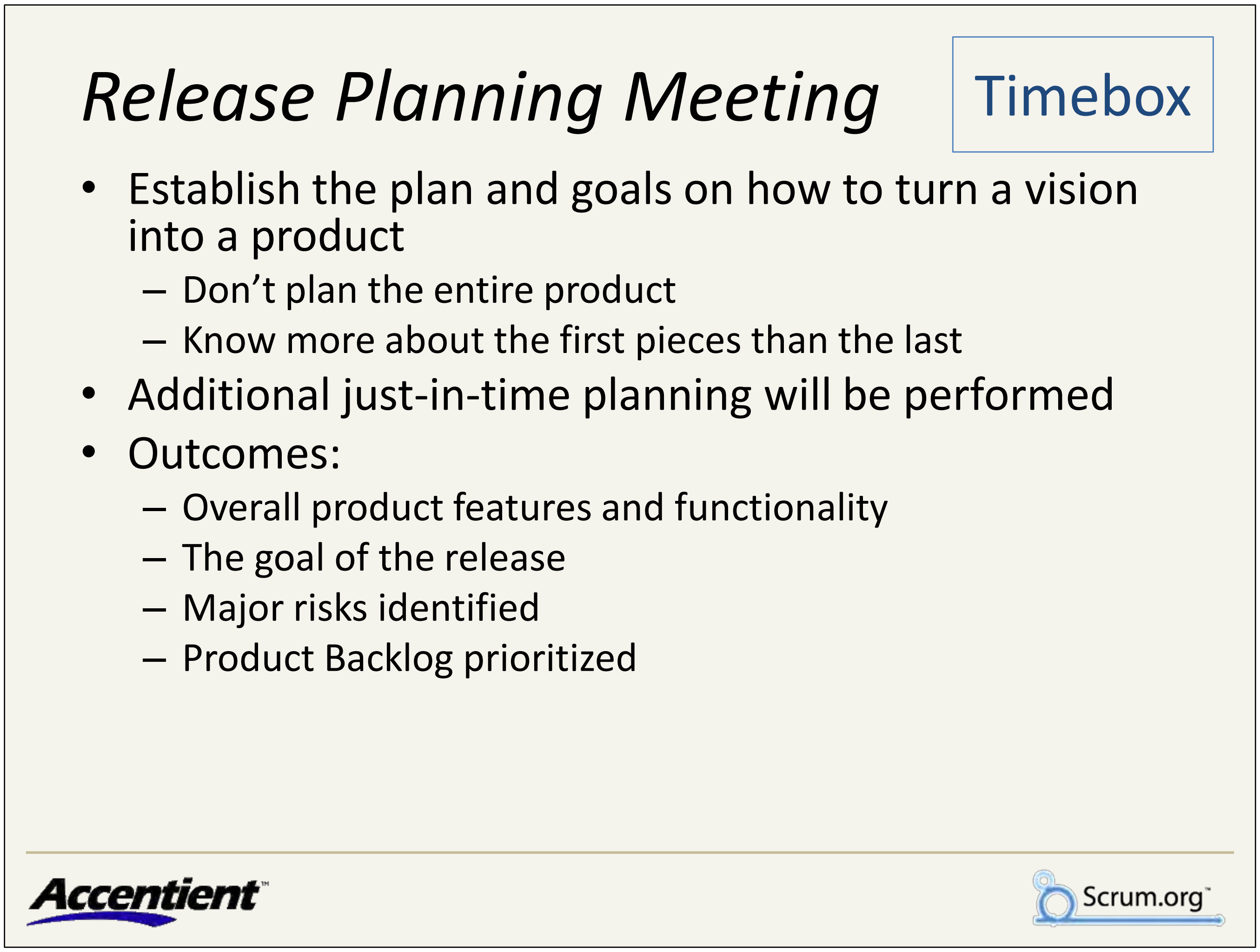
The team

Teams of developers turn Product Backlog items into increments of potentially shippable functionality every Sprint. Teams are cross-functional which is to say that team members must have all of the skills necessary to create an increment of work. Team members often have specialized skills, such as programming, quality control, business analysis, architecture, user interface design, or database design. Shared skills are important and teams should strive to overlap in their skills to some degree. This is not to say that a team of generalists is preferred. Some team members will still be “rock stars” in certain areas, but they should strive to pick up new skills, especially if they are needed but not available.

Put a different way, those team members who refuse to code because they are architects or designers are not good a fit for a Scrum Team. Everyone chips in, even if that requires learning new skills or remembering old ones. There are no titles on Scrum Teams, and there are no exceptions to this rule. Teams do not contain sub-teams dedicated to particular domains like testing or business analysis, either.

Team != Scrum Team

To be true to the definition, it’s important to point out that the two terms do not mean the same thing. The *Team* is a cross-functional group of people responsible for managing itself to develop the product. The *Scrum Team* is the Product Owner, ScrumMaster, and the team.



Release Planning Meeting

Timebox

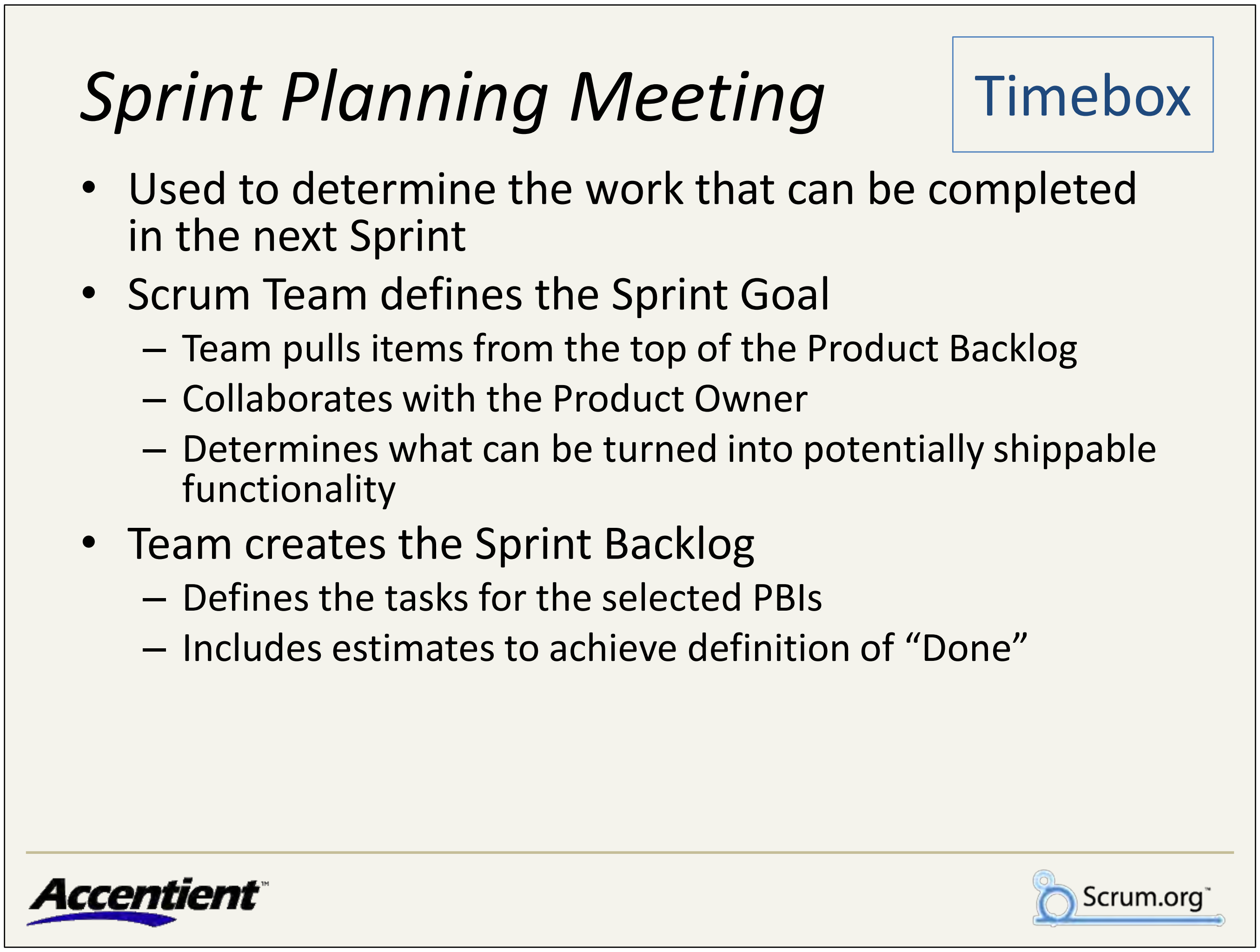
- Establish the plan and goals on how to turn a vision into a product
  - Don’t plan the entire product
  - Know more about the first pieces than the last
- Additional just-in-time planning will be performed
- Outcomes:
  - Overall product features and functionality
  - The goal of the release
  - Major risks identified
  - Product Backlog prioritized

Accentient™ Scrum.org™

The Release Planning meeting

Release planning is when you establish a plan and goals that the Scrum Teams and the rest of the organizations can understand and communicate. Release planning answers the questions, “How can we turn the vision into a winning product in the best possible way? How can we meet or exceed the desired customer satisfaction and return on investment?” The release plan establishes the goal of the release, the highest priority Product Backlog, the major risks, and the overall features and functionality that the release will contain. It also establishes a probable delivery date and cost that should hold if nothing changes. The organization can then inspect progress and make changes to this release plan on a Sprint-by-Sprint basis.

Products are built iteratively using Scrum, wherein each Sprint creates an increment of the product, starting with the most valuable and riskiest. More and more Sprints create additional increments of the product. Each increment is a potentially shippable slice of the entire product. When the Product Owner determines the product's latest increment of added business value is production-ready, the product is released.



Sprint Planning Meeting

Timebox

- Used to determine the work that can be completed in the next Sprint
- Scrum Team defines the Sprint Goal
  - Team pulls items from the top of the Product Backlog
  - Collaborates with the Product Owner
  - Determines what can be turned into potentially shippable functionality
- Team creates the Sprint Backlog
  - Defines the tasks for the selected PBIs
  - Includes estimates to achieve definition of "Done"

Accentient™ Scrum.org™

The Sprint Planning meeting

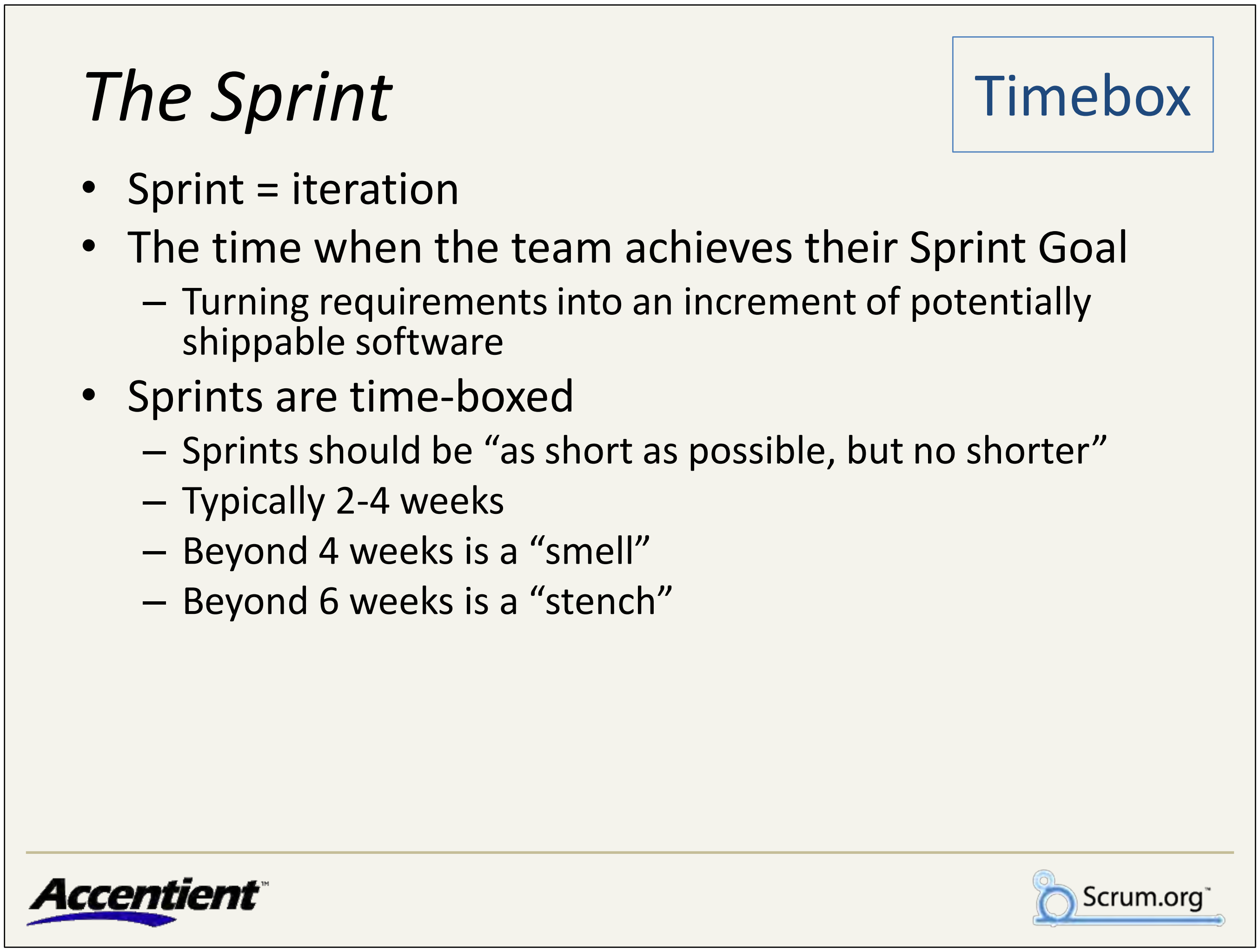
The Sprint Planning meeting is when the iteration is planned. There are two parts to the Sprint Planning Meeting: the "What?" part and the "How?" part. Some Scrum Teams combine the two. In the first part, the Scrum Team addresses the question of "What?" Here, the Product Owner presents the top priority Product Backlog to the team. They work together to figure out what functionality is to be developed during the next Sprint. The input to this meeting is the Product Backlog, the latest increment of product, the capacity of the team, and past performance of the team. The amount of backlog the team selects is solely up to the team. Only the team can assess what it can accomplish over the upcoming Sprint.

How long should the meeting be?

The Sprint Planning meeting is a time-boxed meeting. The actual length is a product of the Sprint length:

- For a 4 week (20 working days) Sprint, your meeting should be 8 hours
- For a 3 week (15 working days) Sprint, your meeting should be 6 hours
- For a 2 week (10 working days) Sprint, your meeting should be 4 hours

For shorter Sprints, allocate approximately 5% of the total Sprint length to the meeting.



The Sprint

Timebox

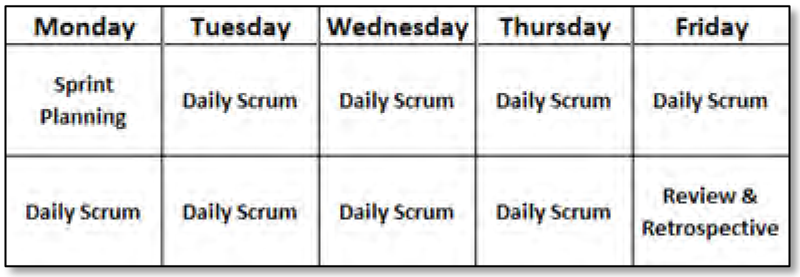
- Sprint = iteration
- The time when the team achieves their Sprint Goal
  - Turning requirements into an increment of potentially shippable software
- Sprints are time-boxed
  - Sprints should be "as short as possible, but no shorter"
  - Typically 2-4 weeks
  - Beyond 4 weeks is a "smell"
  - Beyond 6 weeks is a "stench"

Accentient™ Scrum.org™

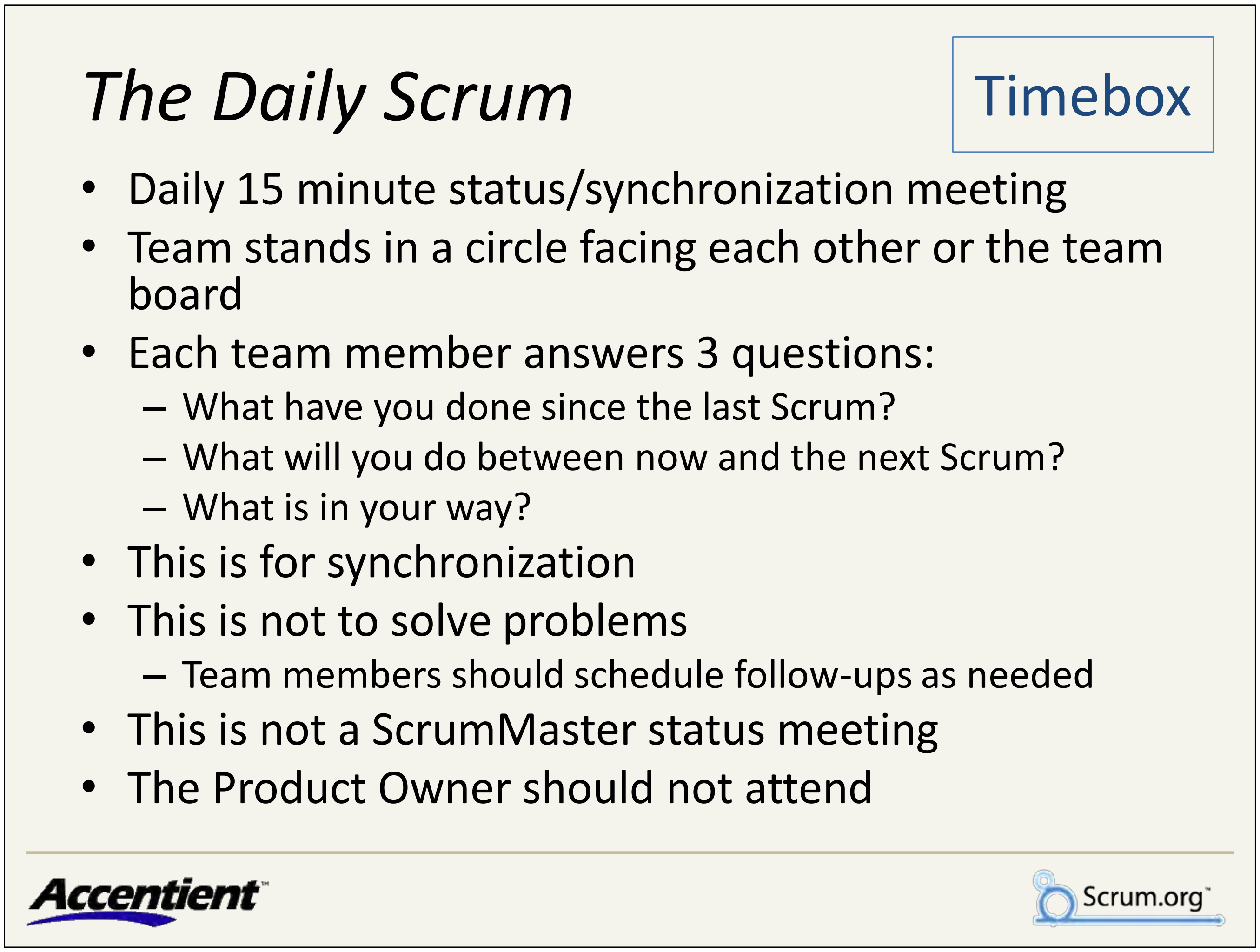
The Sprint

During the Sprint, the ScrumMaster ensures that no changes are made that would negatively impact the Sprint Goal. Both team composition and quality goals remain constant throughout the Sprint. Sprints contain and consist of the Sprint Planning meeting, the development work, the Sprint Review, and the Sprint Retrospective. Sprints occur one after another, with no time in between Sprints.

Example: two week Sprint



Monday	Tuesday	Wednesday	Thursday	Friday
Sprint Planning	Daily Scrum	Daily Scrum	Daily Scrum	Daily Scrum
Daily Scrum	Daily Scrum	Daily Scrum	Daily Scrum	Review & Retrospective



The Daily Scrum

Timebox

- Daily 15 minute status/synchronization meeting
- Team stands in a circle facing each other or the team board
- Each team member answers 3 questions:
  - What have you done since the last Scrum?
  - What will you do between now and the next Scrum?
  - What is in your way?
- This is for synchronization
- This is not to solve problems
  - Team members should schedule follow-ups as needed
- This is not a ScrumMaster status meeting
- The Product Owner should not attend

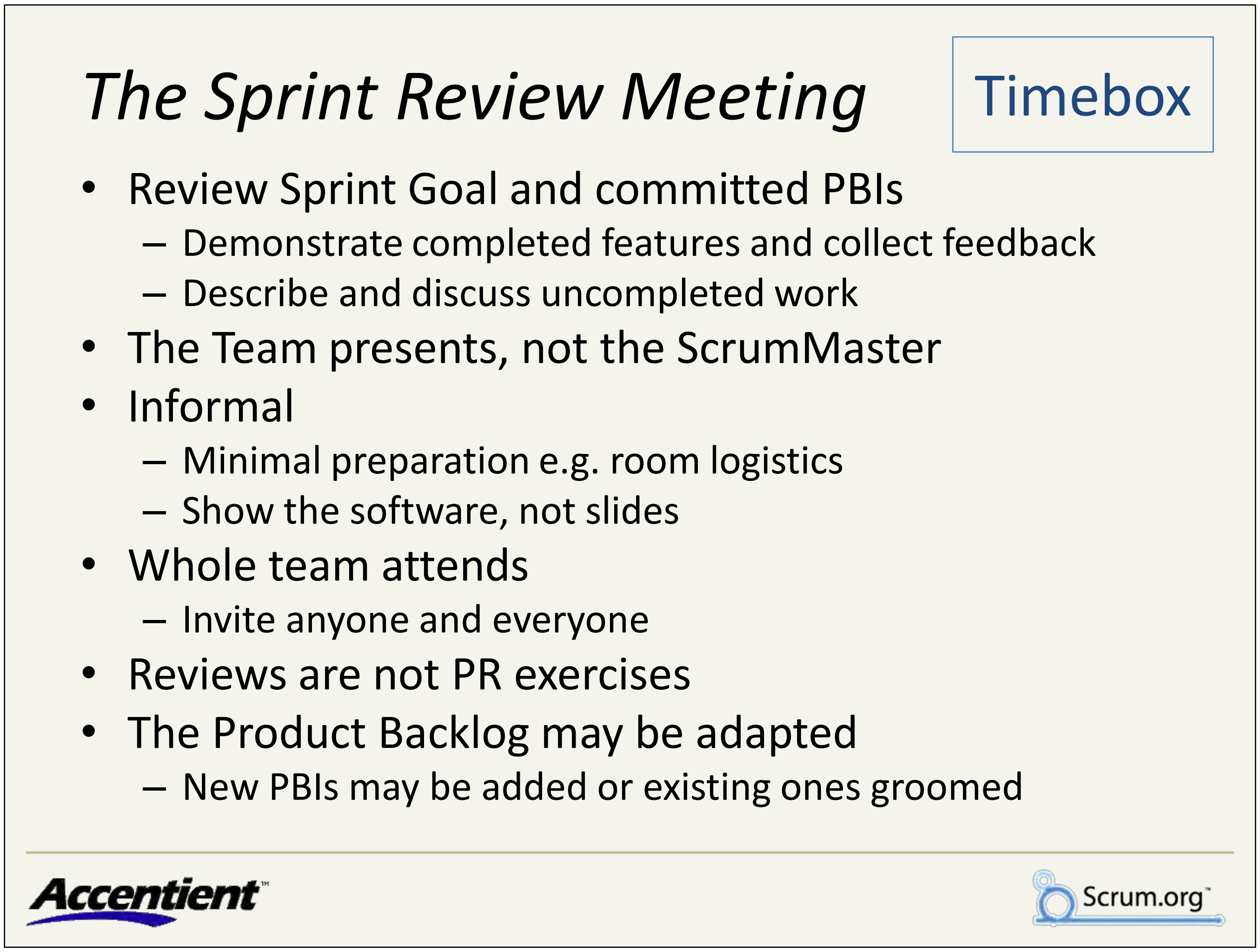
The Daily Scrum

Each Team meets daily for a 15-minute inspect and adapt meeting called the Daily Scrum. The meeting is for the sake of the team and is not meant to be attended by the Product Owner, users, or any stakeholders. The Daily Scrum is at the same time and same place throughout the Sprint. You can decide on the logistics during the Sprint Planning meeting.

During the meeting, each team member explains:

- What he or she has accomplished since the last meeting
- What he or she is going to do before the next meeting
- What obstacles are in his or her way

Daily Scrums improve communications, eliminate other meetings, identify and remove impediments to development, highlight and promote quick decision-making, and improve everyone’s level of project knowledge. Follow-up meetings usually occur to make adaptations to the upcoming work in the Sprint. The intent is to optimize the probability that the team will meet its Goal. This is crucial to be able to inspect and adapt the process.



The Sprint Review Meeting

Timebox

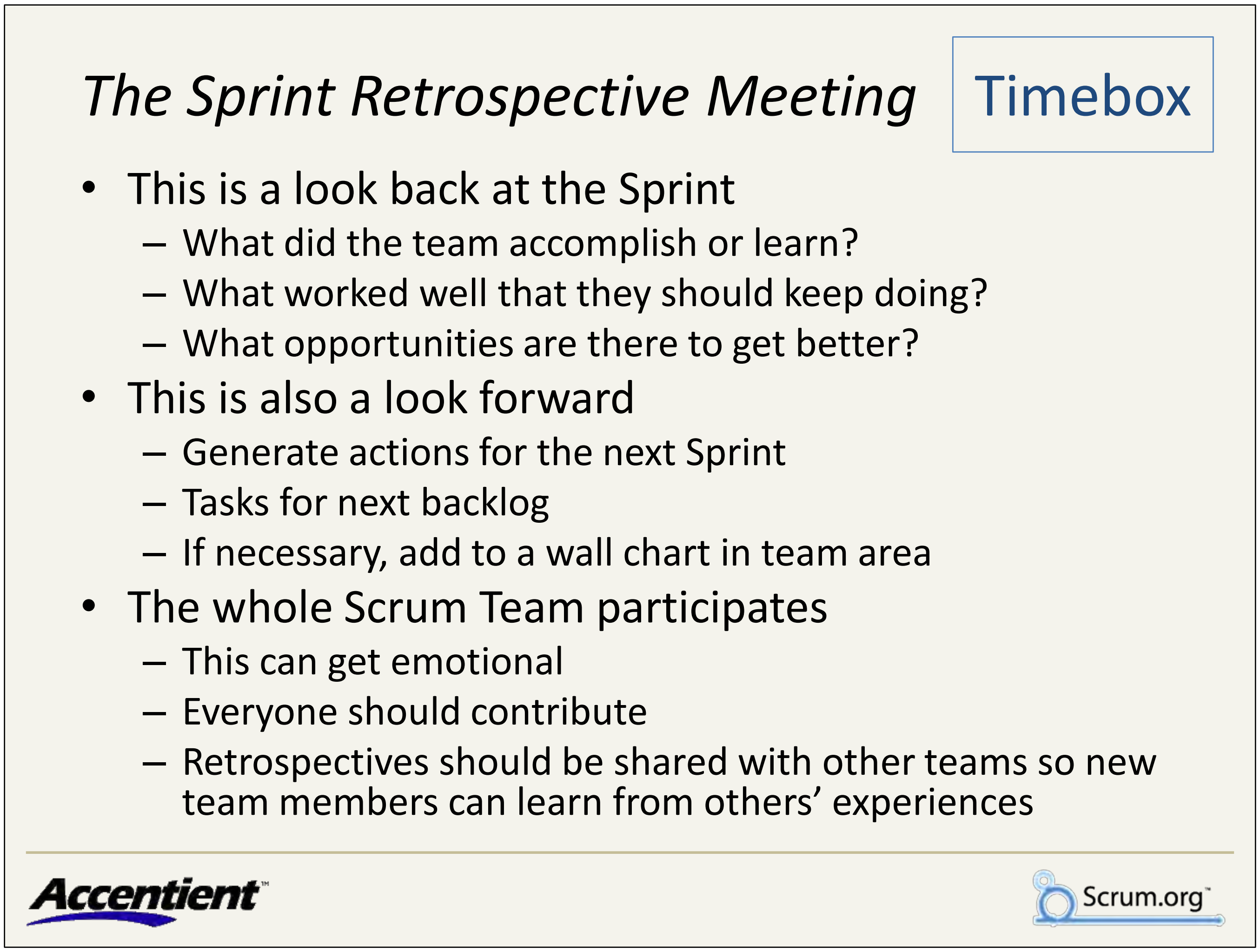
- Review Sprint Goal and committed PBIs
  - Demonstrate completed features and collect feedback
  - Describe and discuss uncompleted work
- The Team presents, not the ScrumMaster
- Informal
  - Minimal preparation e.g. room logistics
  - Show the software, not slides
- Whole team attends
  - Invite anyone and everyone
- Reviews are not PR exercises
- The Product Backlog may be adapted
  - New PBIs may be added or existing ones groomed

Accentient™ Scrum.org™

The Sprint Review meeting

At the end of the Sprint, a Sprint Review meeting is held. This is a four hour time-boxed meeting for one month Sprints. For Sprints of lesser duration, this meeting must not consume more than 5% of the total Sprint. During the Sprint Review, the Scrum Team and stakeholders collaborate about what was just done. Based on that and changes to the Product Backlog during the Sprint, they collaborate about what are the next things that could be done. This is an informal meeting, with the presentation of the functionality intended to foster collaboration about what to do next.

The meeting includes at least the following elements. The Product Owner identifies what has been done and what hasn't been done. The team may choose to share what went well (with the product) during the Sprint and what problems it ran into, and how it solved these problems. The team then demonstrates the work that is done and answers questions. The Product Owner then discusses the Product Backlog as it stands. He or she projects likely completion dates with various velocity assumptions. The entire group then collaborates about what it has seen and what this means regarding what to do next. The Sprint Review provides valuable input to subsequent Sprint Planning meetings.



The Sprint Retrospective Meeting

Timebox

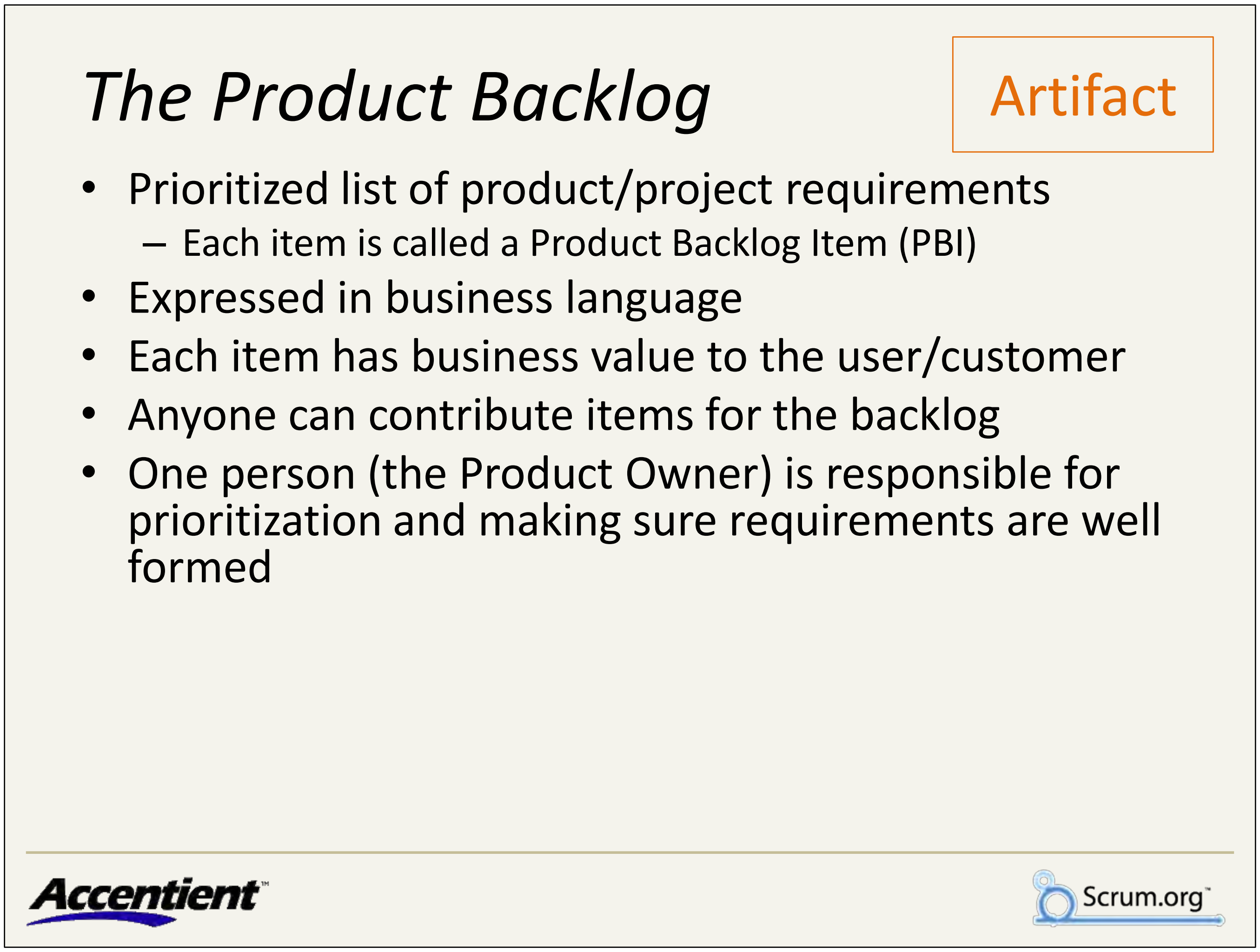
- This is a look back at the Sprint
  - What did the team accomplish or learn?
  - What worked well that they should keep doing?
  - What opportunities are there to get better?
- This is also a look forward
  - Generate actions for the next Sprint
  - Tasks for next backlog
  - If necessary, add to a wall chart in team area
- The whole Scrum Team participates
  - This can get emotional
  - Everyone should contribute
  - Retrospectives should be shared with other teams so new team members can learn from others' experiences

Accentient™ Scrum.org™

The Sprint Retrospective meeting

After the Sprint Review and prior to the next Sprint Planning meeting, the Scrum Team should hold a Sprint Retrospective meeting. At this three hour, time-boxed meeting the ScrumMaster encourages the revision, within the Scrum framework and practices, of the development process to make it more effective and enjoyable for the next Sprint.

The purpose of the Retrospective is to inspect how the last Sprint went in regards to people, relationships, process and tools. The inspection should identify and prioritize the major items that went well (with the process) and those items that-if done differently-could make things even better. These include Scrum Team composition, meeting arrangements, tools, definition of "done," methods of communication, and processes for turning Product Backlog items into something "done." By the end of the Sprint Retrospective, the Scrum Team should have identified actionable improvement measures that it implements in the next Sprint. These changes become the adaptation to the empirical inspection.



The Product Backlog

Artifact

- Prioritized list of product/project requirements
  - Each item is called a Product Backlog Item (PBI)
- Expressed in business language
- Each item has business value to the user/customer
- Anyone can contribute items for the backlog
- One person (the Product Owner) is responsible for prioritization and making sure requirements are well formed

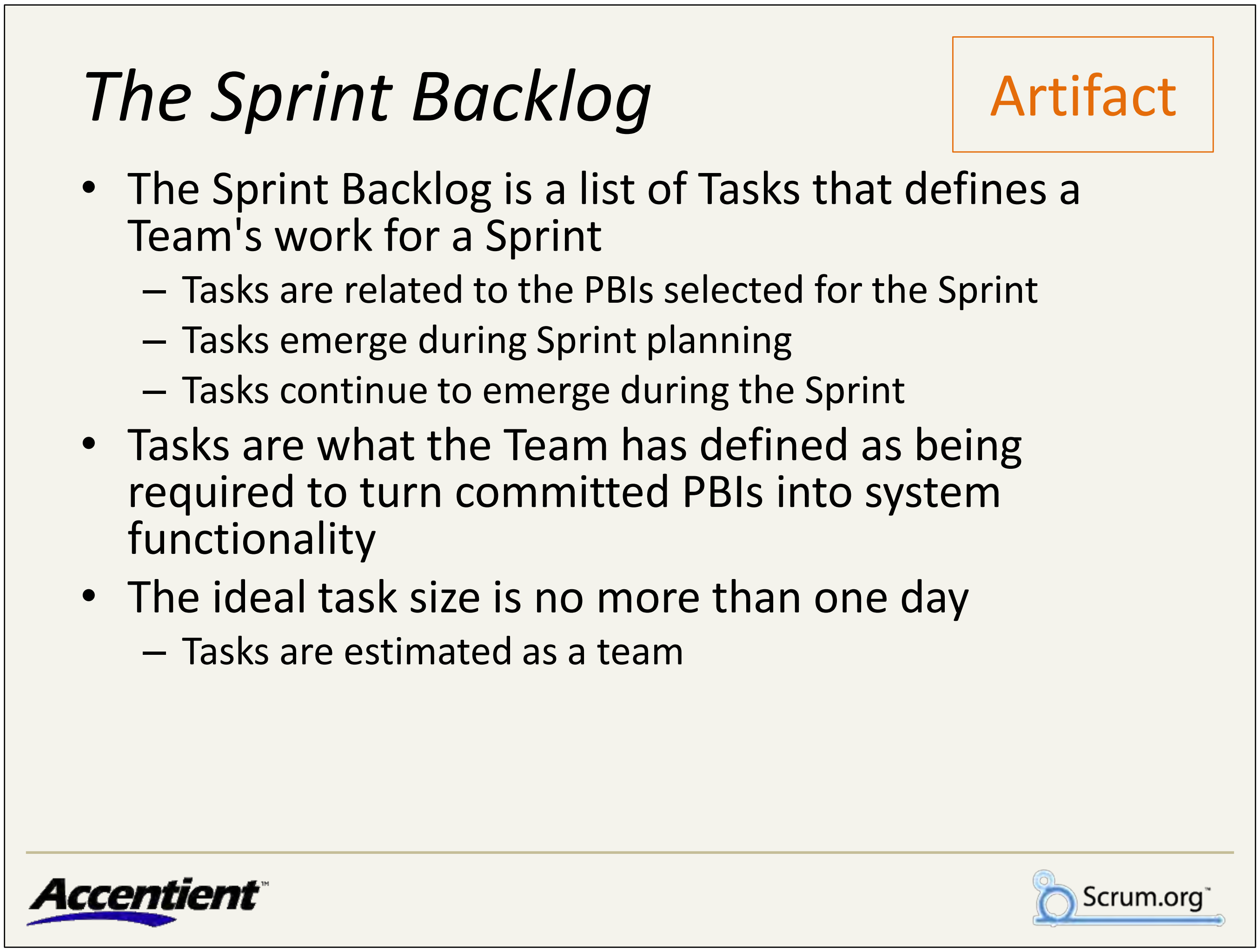
Accentient™ Scrum.org™

The Product Backlog

The requirements for the product that the team(s) is developing are listed in the Product Backlog. The Product Owner is responsible for the Product Backlog, its contents, its availability, and its prioritization. The Product Backlog is never complete. The initial cut at developing it only lays out the initially known and best-understood requirements. The Product Backlog evolves as the product and the environment in which it will be used evolves. The Backlog is dynamic in that it constantly changes to identify what the product needs to be appropriate, competitive, and useful. As long as a product exists, the Product Backlog also exists.

The Product Backlog represents everything necessary to develop and launch a successful product. It is a list of all features, functions, technologies, enhancements, and bug fixes that constitute the changes that will be made to the product for future releases. Product Backlog items have the attributes of a description, priority, and estimate. Priority is driven by risk, value, and necessity. There are many techniques for assessing these attributes.

Product Backlog is sorted in order of priority. Top priority Product Backlog drives immediate development activities: the higher the priority, the more urgent it is, the more it has been thought about, and the more consensus there is regarding its value. Higher priority backlog is clearer and has more detailed information than lower priority backlog. Better estimates are made based on the greater clarity and increased detail. The lower the priority, the less the detail, until you can barely make out the item.



The Sprint Backlog

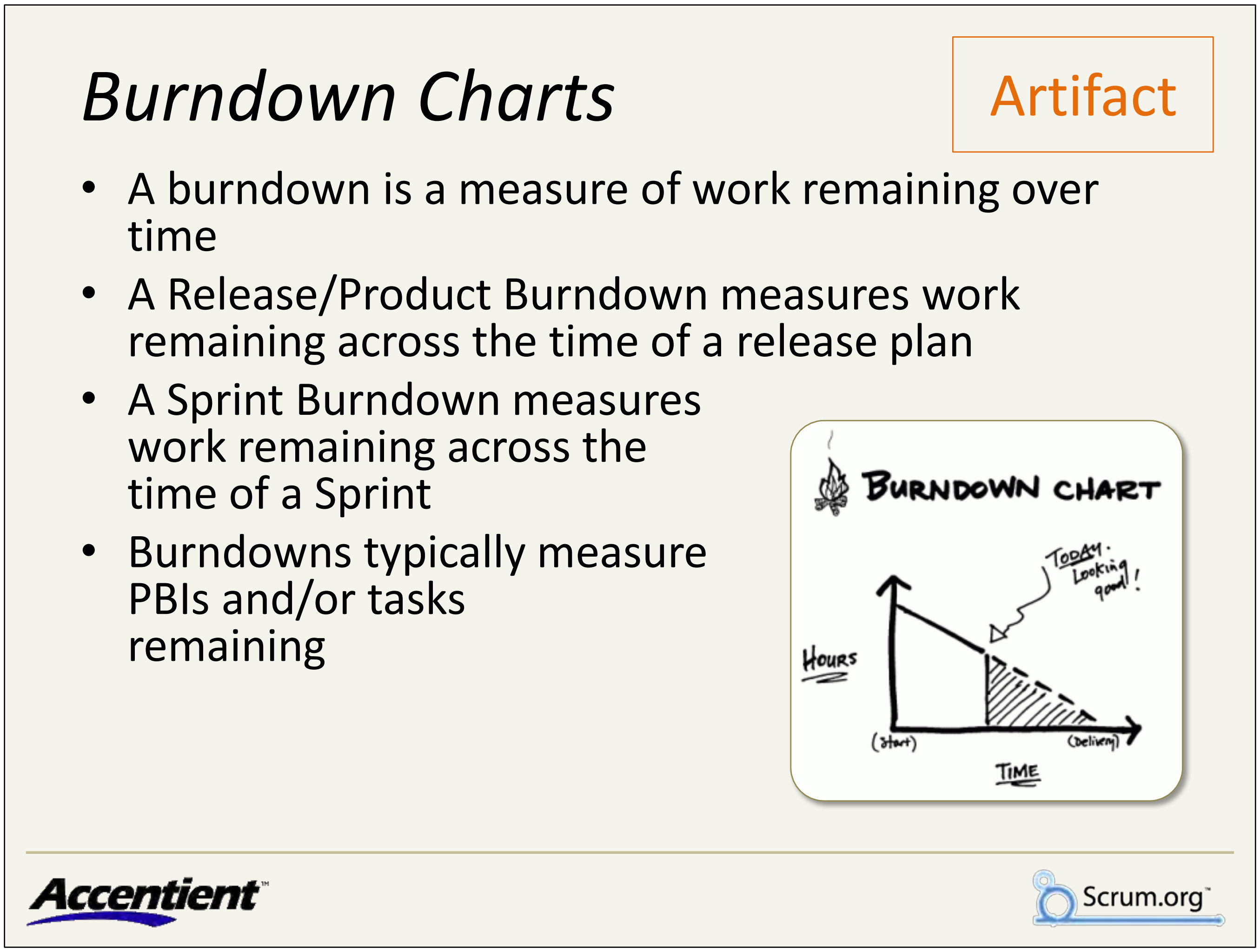
Artifact

- The Sprint Backlog is a list of Tasks that defines a Team's work for a Sprint
  - Tasks are related to the PBIs selected for the Sprint
  - Tasks emerge during Sprint planning
  - Tasks continue to emerge during the Sprint
- Tasks are what the Team has defined as being required to turn committed PBIs into system functionality
- The ideal task size is no more than one day
  - Tasks are estimated as a team

The Sprint Backlog

The Sprint Backlog consists of the tasks the team performs to turn Product Backlog items into a "done" increment. Many are developed during the Sprint Planning Meeting. It is all of the work that the team identifies as necessary to meet the Sprint goal. Tasks must be decomposed. The decomposition is enough so changes in progress can be understood in the Daily Scrum.

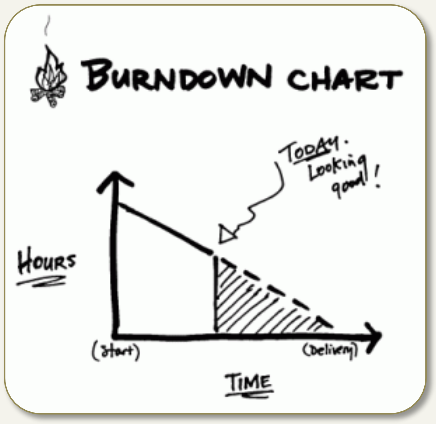
The team modifies the Sprint Backlog throughout the Sprint as more and more tasks emerge. As the team gets into individual tasks, it may find out that more or fewer tasks are needed, or that a given task will take more or less time than had been expected. As new work is required, the team adds it to the Sprint Backlog. As tasks are worked on or completed, the hours of estimated remaining work for each task is updated. When tasks are deemed unnecessary, they are removed. Only the team can change its Sprint Backlog during a Sprint. Only the team can change the contents or the estimates. The Sprint Backlog is a highly visible, real-time picture of the work that the team plans to accomplish during the Sprint, and it belongs solely to the team.



Burndown Charts

Artifact

- A burndown is a measure of work remaining over time
- A Release/Product Burndown measures work remaining across the time of a release plan
- A Sprint Burndown measures work remaining across the time of a Sprint
- Burndowns typically measure PBIs and/or tasks remaining



Accentient™

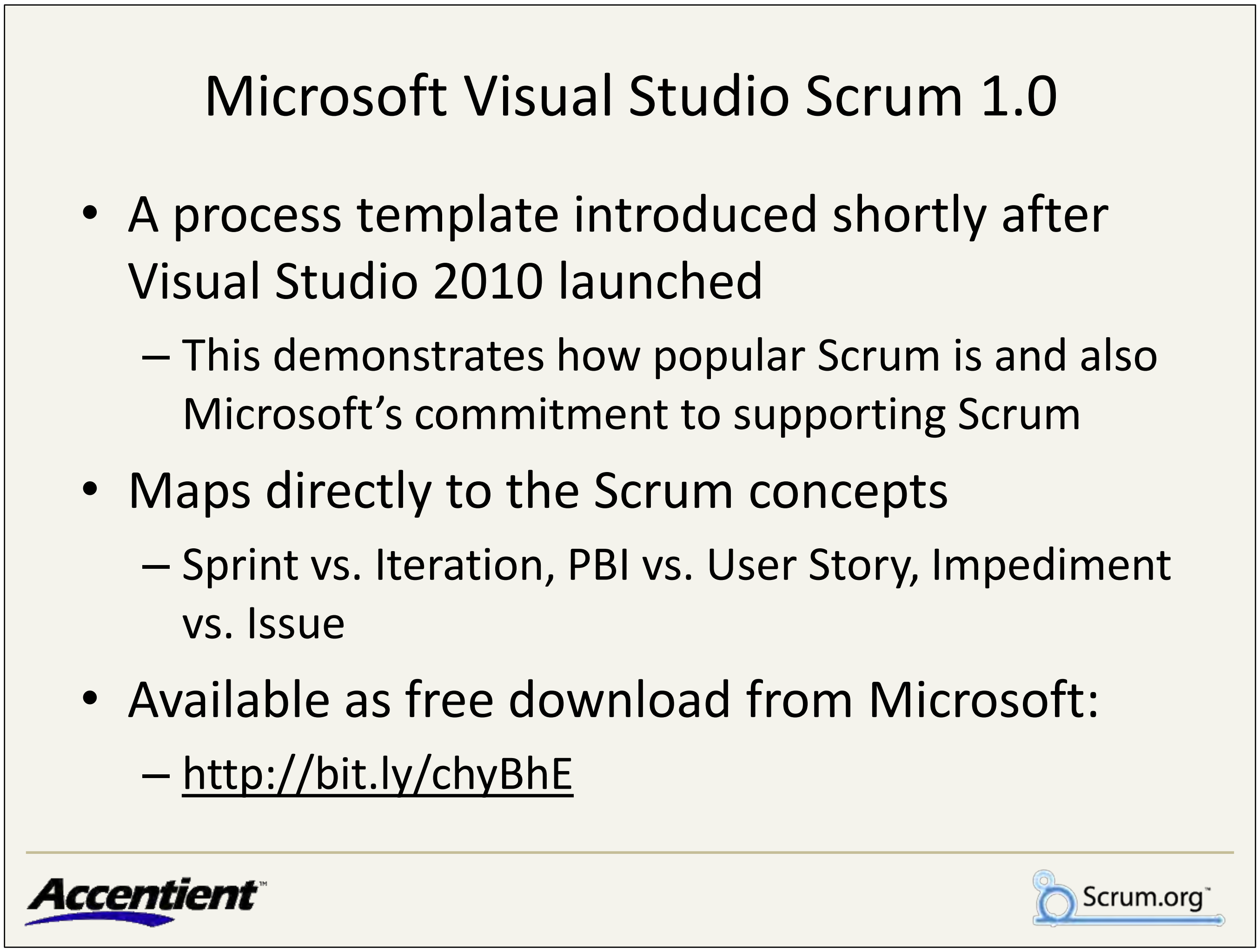
Scrum.org™

Burndown charts

The Release Burndown graph records the sum of remaining Product Backlog estimated effort across Sprints. The estimated effort is in whatever unit of work the Scrum Team and organization have decided upon. The units of time are usually Sprints. Product Backlog item estimates are calculated initially during Release Planning, and thereafter as they are created. During Product Backlog grooming they are reviewed and revised.

The team is responsible for all estimates and can make them at any time. The Product Owner may influence the team by helping understand trade-offs and/or breaking Product Backlog items apart into smaller items, but the final estimate is made by the team. The Product Owner should keep an updated Product Backlog list and Release Burndown posted at all times. A trend line can be drawn based on the change in remaining work.

The Sprint Backlog Burndown is a graph of the amount of Sprint Backlog work remaining in a Sprint across time in the Sprint. To create this graph, determine how much work remains by summing the backlog estimates every day of the Sprint. The amount of work remaining for a Sprint is the sum of the work remaining for the entire Sprint Backlog. Keep track of these sums by day and use them to create a graph that shows the work remaining over time. By drawing a line through the points on the graph, the team can manage its progress in completing a Sprint's work. Duration is not considered in Scrum. Work remaining and date are the only variables of interest.



Microsoft Visual Studio Scrum 1.0

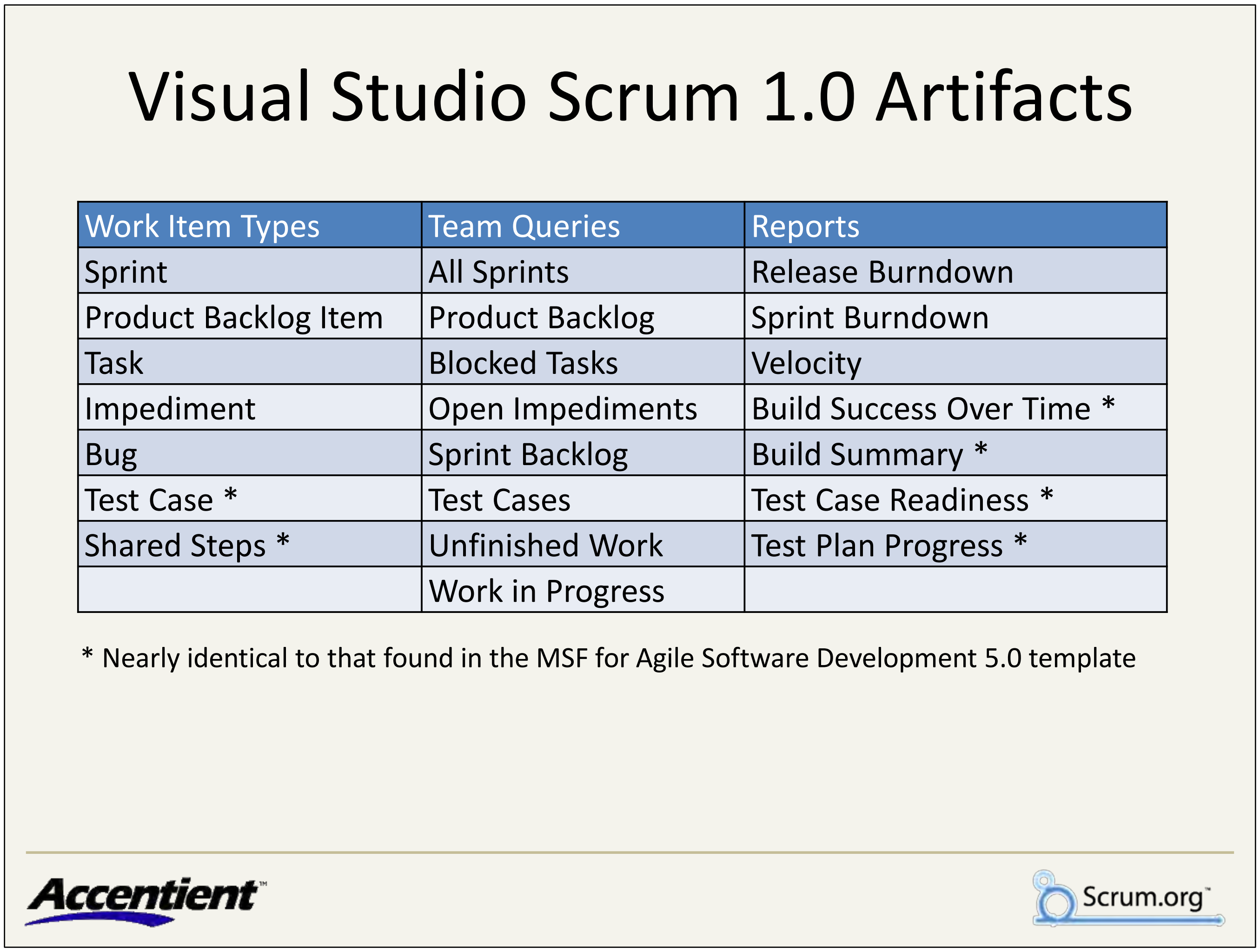
- A process template introduced shortly after Visual Studio 2010 launched
  - This demonstrates how popular Scrum is and also Microsoft's commitment to supporting Scrum
- Maps directly to the Scrum concepts
  - Sprint vs. Iteration, PBI vs. User Story, Impediment vs. Issue
- Available as free download from Microsoft:
  - http://bit.ly/chyBhE

Accentient™ Scrum.org™

Microsoft Visual Studio Scrum 1.0

Out of the box, Team Foundation Server 2010 installs two process templates: MSF for Agile Software Development v5.0 and MSF for CMMI Process Improvement v5.0. While both of these templates are very mature and robust in features, neither support Scrum that well. Many teams have successfully implemented Scrum using the MSF Agile process template, but with a fair amount of customization or guidance to do so.

Shortly after the release of Team Foundation Server 2010, Microsoft released Visual Studio Scrum v1.0. This is a brand new process template, built from the ground up, specifically for Scrum Teams. The creation of this template was a direct result of the community and stakeholders of this course telling Microsoft that they wanted one. As you know, Scrum has become a dominant methodology in software development and the community told Microsoft that they wanted a process template aimed directly at Scrum Teams.



Visual Studio Scrum 1.0 Artifacts

Work Item Types	Team Queries	Reports
Sprint	All Sprints	Release Burndown
Product Backlog Item	Product Backlog	Sprint Burndown
Task	Blocked Tasks	Velocity
Impediment	Open Impediments	Build Success Over Time *
Bug	Sprint Backlog	Build Summary *
Test Case *	Test Cases	Test Case Readiness *
Shared Steps *	Unfinished Work	Test Plan Progress *
	Work in Progress	

* Nearly identical to that found in the MSF for Agile Software Development 5.0 template

Work item types

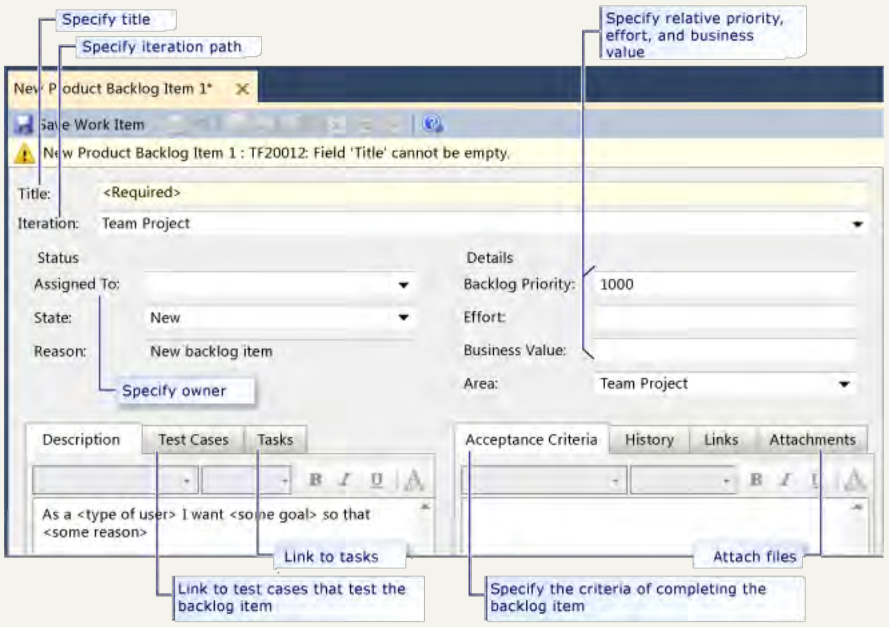
You use work items to track, monitor, and report the development progress of a product and its features. A work item is a database record that you create in Team Foundation Server to record the definition, assignment, priority, and state of work. The Visual Studio Scrum 1.0 template includes seven work item types. Three of these are new (Sprint, Product Backlog Item, and Impediment). The other four have been modified from the MSF Agile v5.0 template (Bug, Task, Test Case, and Shared Steps).

By default, to view a work item, a Team Foundation Server user must be a member of the *Readers* group or have the permission in the *View work items in this node* set to Allow. To modify a work item, a user must be a member of the *Contributors* group or have the permission in the *Edit work items in this node* set to Allow.

Note: Team Foundation Server security configuration is beyond the scope of this course.

Product Backlog Item Work Item Type

- Represents a product requirement



New Product Backlog Item 1*

New Product Backlog Item 1 : TF20012: Field 'Title' cannot be empty.

Title: <Required>

Iteration: Team Project

Status

Assigned To:

State: New

Reason: New backlog item

Details

Backlog Priority: 1000

Effort:

Business Value:

Area: Team Project

As a <type of user> I want <some goal> so that <some reason>

Accentient™ Scrum.org™

Product Backlog Item work item type

By defining and managing Product Backlog Items, a team can capture the requirements of the product. The Product Owner defines, prioritizes, and maintains the backlog items. The team estimates the effort and assists the Product Owner in understanding the business value and setting the priority for each item. The team then commits to delivering the highest priority items in each Sprint.

When the Product Owner defines a Product Backlog Item, he or she should focus on its value and avoid descriptions of how the team should develop it. The Product Owner can prioritize the Product Backlog based on each item's business value, effort, and relative dependency on other backlog items. The Product Backlog will evolve quickly if the business requirements of your project and other conditions of the team change constantly. To minimize redundant work, the team should specify details only for the highest priority items.

To create a new Product Backlog Item work item

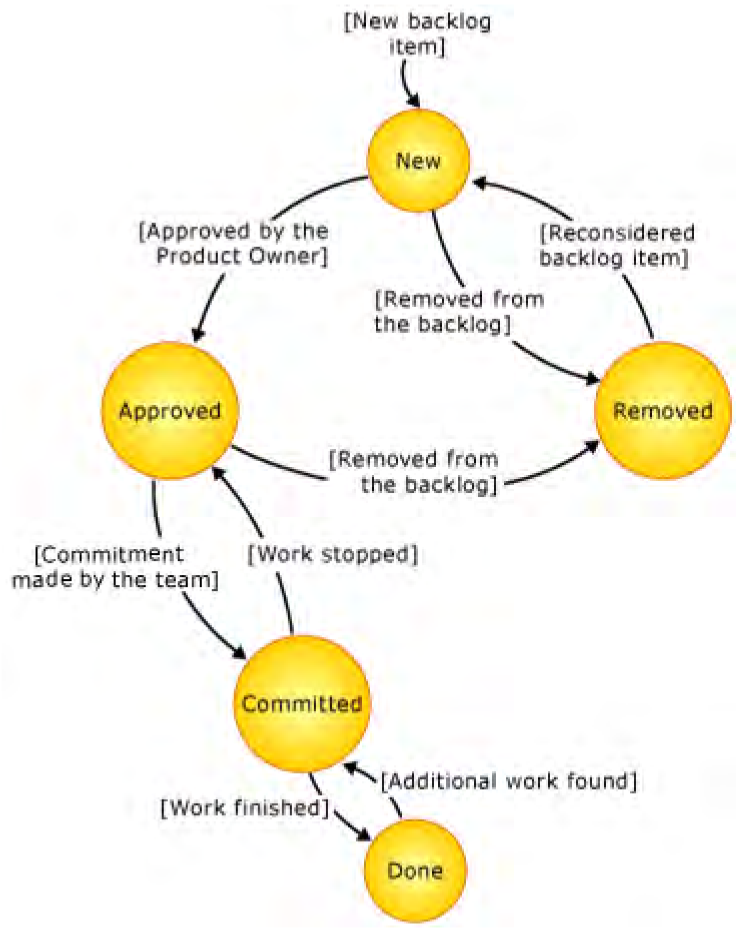
1. In the top section of the work item form for the Product Backlog Item, specify one or more of the following types of information:
    - In the **Title** field (required), type a short description; good story titles reflect the value to the customer or functionality that needs to be implemented
    - In the **Iteration** dropdown, select the Sprint in which you will commit to implement this item; if you have yet to commit to the work, leave it set to the default (root) value
    - In the **Assigned To** dropdown, select the name of the team member who owns the item; some Scrum Teams prefer to assign all Product Backlog Items to the Product Owner
    - In the **Backlog Priority** field, type a number that indicates the relative importance of this item compared to the other Product Backlog Items; smaller numbers indicate higher priority than larger numbers; the default value of this field is 1000, which places the item at the bottom when sorted by Backlog Priority
    - In the **Effort** field, type a number that indicates a relative rating for the amount of work that will be required to implement this Product Backlog Item; larger numbers indicate more work than smaller numbers
    - In the **Business Value** field, type a number that indicates the relative business value of the item
    - In the **Area** dropdown, select the appropriate area or leave it blank (set to the root) to be assigned later
2. On the **Description** tab, specify as much detail as you want to describe the Product Backlog Item; you should use the user story format (As a <type of user> I want <some goal> so that <some reason>) to ensure that a business value proposition is captured in addition to the basics of the requirement
3. On the **Acceptance Criteria** tab, describe the criteria that you use to verify whether your team has fulfilled the requirements of the item
4. On the **History** tab, add comments that you want to capture as part of the historical record
    - Every time that a team member updates the work item, its history shows the date of the change, the team member who made the change, and the fields that changed
5. On the **Attachments** tab, attach any files that provide more details about the Product Backlog Item
6. **Link** the product backlog item to other work items by performing the following tasks:
    - On the **Test Cases** tab, create links from the Product Backlog Item to test cases
    - On the **Tasks** tab, create links from the Product Backlog Item to tasks
    - On the **Links** tab, create links from the Product Backlog Item to other Product Backlog Items or to other types of work items, such as Impediments; you can also add a hyperlink to a Web site or to a file that is stored on a server or a Web site
7. **Save** the work item

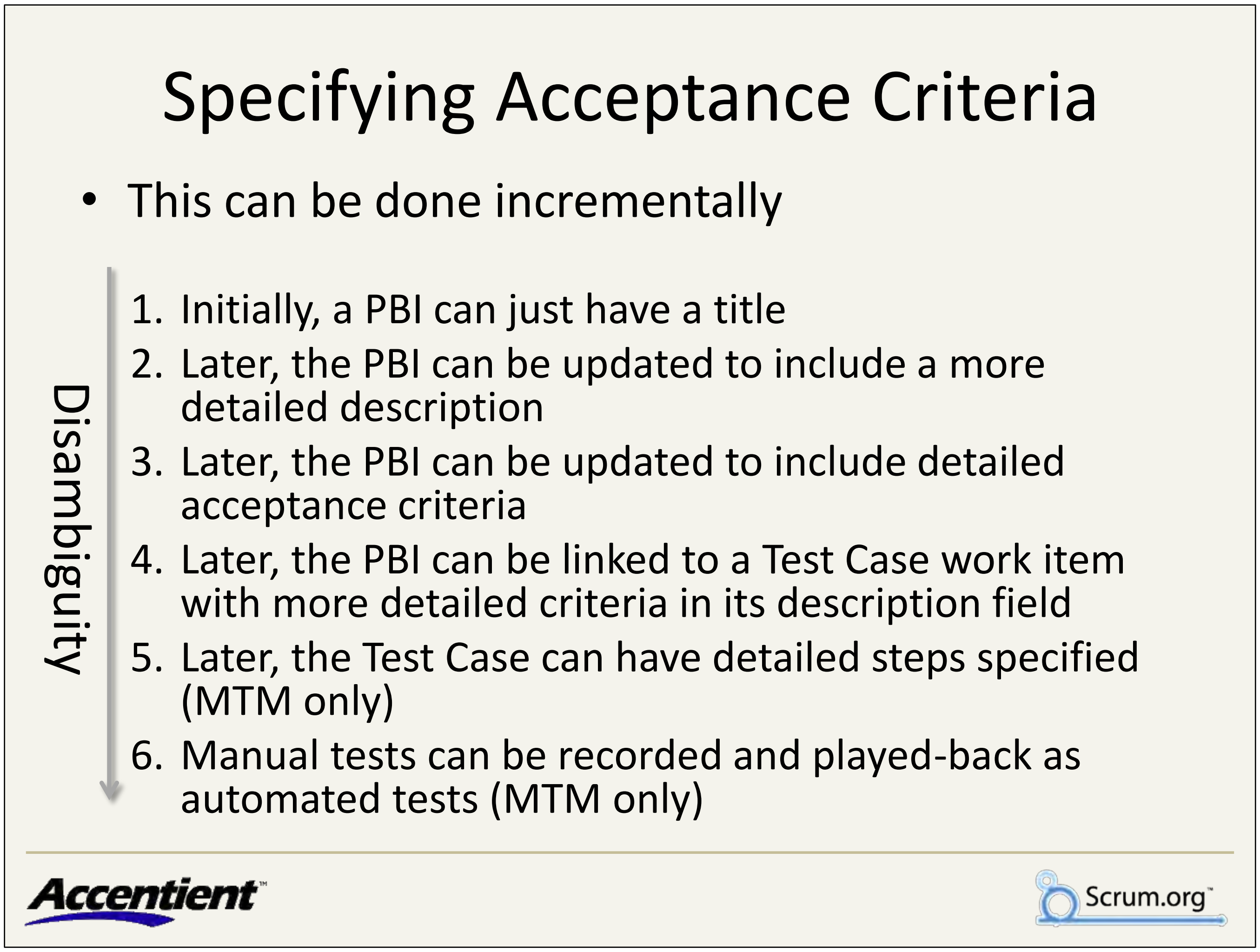
Managing Epic Stories

If you have a PBI which is larger than can be completed in one Sprint, you will need to break this down. If you wish to leave the original PBI work item, you can create two (or more) additional PBI work items and create parent-child links back to the original PBI. To be able to view your Epic story breakdowns, you should then create a direct links query showing PBI's directly-linked to PBIs.

Typical Product Backlog Item workflow progression in Scrum

When a Product Backlog Item is created it is in the **New** state with the default reason "New backlog item". When it is decided that the Product Backlog Item is valid, its state should be changed from New to **Approved** with the reason "Approved by the Product Owner". When the team commits to deliver the Product Backlog Item, its state will be changed to **Committed** with the reason "Commitment made by the team". Finally, when the Product Backlog Item is done per the team's definition and accepted by the Product Owner, the state will be changed to **Done** with the reason "Work finished".


[New backlog item]
New
[Approved by the Product Owner]
[Reconsidered backlog item]
[Removed from the backlog]
Approved
Removed
[Removed from the backlog]
[Commitment made by the team]
[Work stopped]
Committed
[Additional work found]
[Work finished]
Done



Specifying Acceptance Criteria

- This can be done incrementally

Disambiguity

1. Initially, a PBI can just have a title
2. Later, the PBI can be updated to include a more detailed description
3. Later, the PBI can be updated to include detailed acceptance criteria
4. Later, the PBI can be linked to a Test Case work item with more detailed criteria in its description field
5. Later, the Test Case can have detailed steps specified (MTM only)
6. Manual tests can be recorded and played-back as automated tests (MTM only)

Accentient™ Scrum.org™

Specifying acceptance criteria

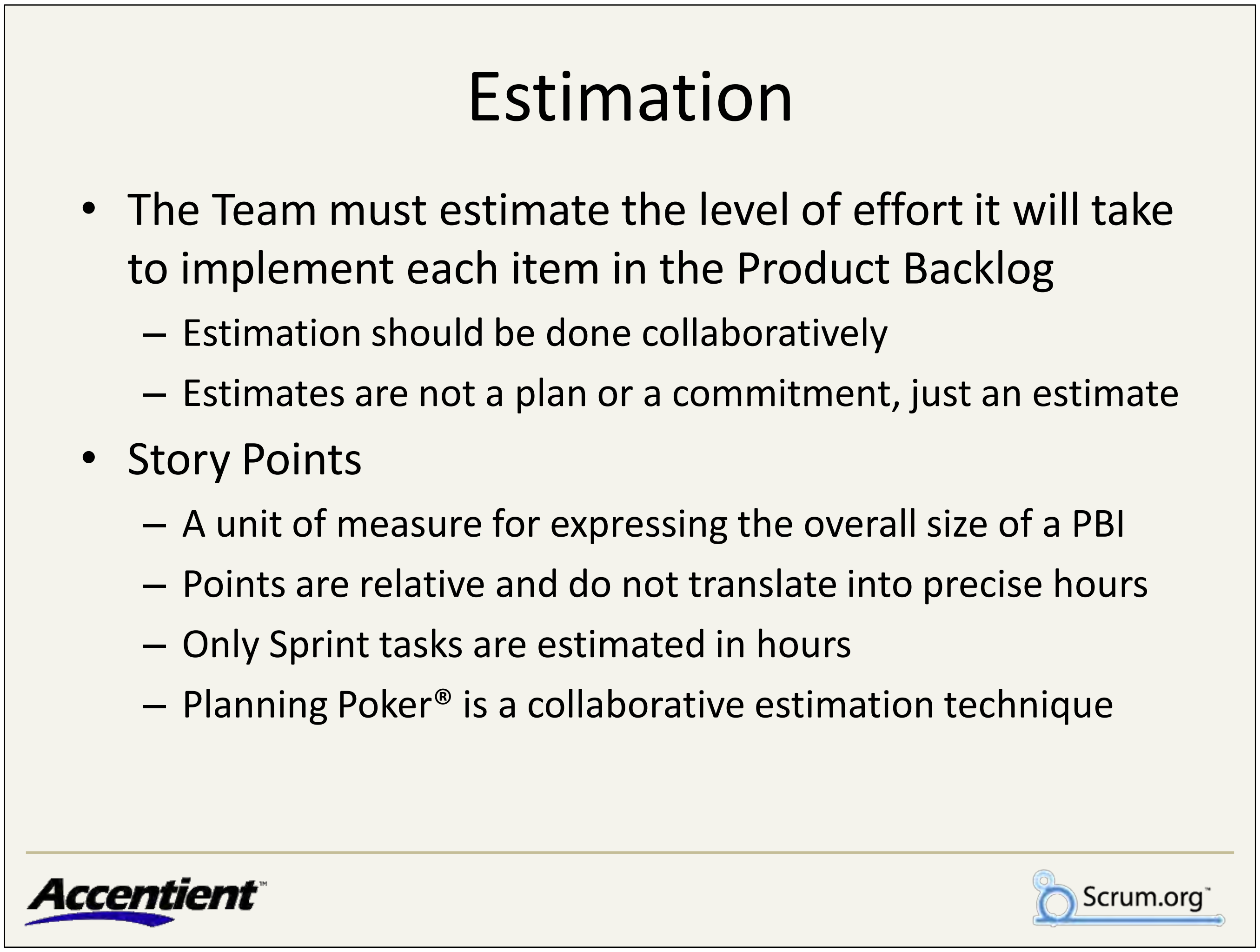
Not all of the acceptance criteria for user stories will be known in the beginning of the Sprint. As the Sprint progresses and the team develops the code, the team will also continuously develop, extend and refine the acceptance tests (along with other types of tests, such as unit and integration tests).

Acceptance tests often evolve from a broad statement of success at the story level into multiple acceptance tests with more detailed criteria, generally resulting in actionable test steps to setup the test, execute the test and evaluate the test results.

Some Questions to Ponder

Given the evolution of acceptance criteria outlined on the slide:

1. At what point would you feel comfortable committing to the work.
2. At what point would it become waterfall if you waited to commit to the work?



Estimation

- The Team must estimate the level of effort it will take to implement each item in the Product Backlog
  - Estimation should be done collaboratively
  - Estimates are not a plan or a commitment, just an estimate
- Story Points
  - A unit of measure for expressing the overall size of a PBI
  - Points are relative and do not translate into precise hours
  - Only Sprint tasks are estimated in hours
  - Planning Poker® is a collaborative estimation technique

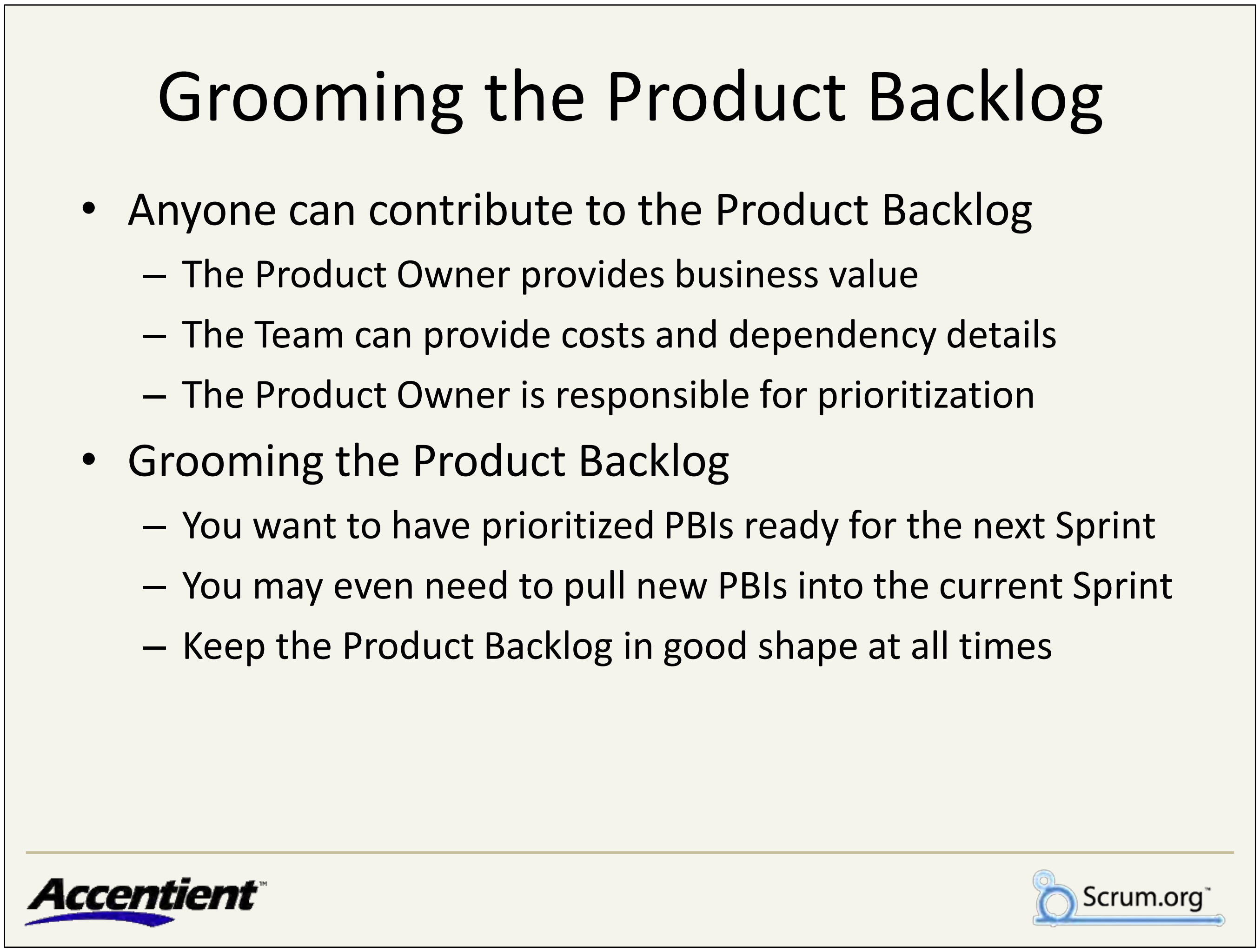
Accentient™ Scrum.org™

Estimating in story points

Teams can use whatever value they wish for estimating. Story points are the most common, but some teams may prefer to call them complexity points or even acorns. One team building pharmaceutical software even used Vicodin points. Vicodin is a prescription pain medication. Whatever term you decide on, using an abstract measure like a “story point” is preferred for Agile software development as it doesn’t imply anything about scheduling. If you were to use the term days, then a Product Owner or stakeholder may perceive a different expectation.

For teams new to Agile estimation, substituting days for points can work until the team gets a better handle on its ability to estimate effort and compute its velocity. These teams should strive to shift to story points as soon as possible. They can then use relative estimation techniques to more accurately size their Product Backlog items.

Sprint tasks, on the other hand, should be estimated in hours in Visual Studio 2010 as the queries, spreadsheets, and reports are geared for that.



Grooming the Product Backlog

- Anyone can contribute to the Product Backlog
  - The Product Owner provides business value
  - The Team can provide costs and dependency details
  - The Product Owner is responsible for prioritization
- Grooming the Product Backlog
  - You want to have prioritized PBIs ready for the next Sprint
  - You may even need to pull new PBIs into the current Sprint
  - Keep the Product Backlog in good shape at all times

Accentient™ Scrum.org™

The Product Backlog

The Product Backlog represents everything necessary to develop and launch a successful product. It is a list of all features, functions, technologies, enhancements, and bug fixes that constitute the changes that will be made to the product for future releases. Product Backlog items have the attributes of a description, priority, and estimate. Priority is driven by risk, value, and necessity. There are many techniques for assessing these attributes.

Any idea or requirement can be added to the Product Backlog - from the most grandiose feature request, to the smallest non-functional requirement. This makes the Product Backlog a very useful tool for calming business users; some business users can cause friction in meetings by demanding or championing new features - in traditional projects this can cause conflict because the Project Manager is trying to keep a handle on Scope, whereas in Scrum the answer becomes "Great idea, we'll put it on the Product Backlog and prioritize against the rest of the Backlog". This technique allows the "Novelty value" of the hot new feature to dissipate so that a considered, rational business priority can be ascribed.

Grooming the Product Backlog

Grooming the backlog is an ongoing activity. The Product Owner needs to work with the team to discuss the Product Backlog items, break them down, and perform high-level estimates (in story points, not days/hours). Since there are no breaks between Sprints, this grooming will have to be performed ongoing during the Sprints. Some organizations have instituted a “Story-time Meeting” or “Requirements Workshop” that happens for a few hours every week or two. Regardless of what you call it, or when you schedule it, a team should budget 5-10% of its time to dedicate to this ongoing effort.

Prioritizing the Product Backlog

A product backlog is a prioritized list of requirements with the estimated effort to turn them into completed product functionality. Ideally, these estimates are in story points and are more precise the higher the item is in priority. Priority should be assigned based on the items of most value to the business or that offer the earliest return on investment. This list should evolve, changing as the business conditions or technology changes.

Product Backlog items can be functional requirements, non-functional requirements, and issues. The precision of the estimate depends on the priority and granularity of the Product Backlog item, with the highest priority items that can be selected in the first few Sprints being very granular and precise.

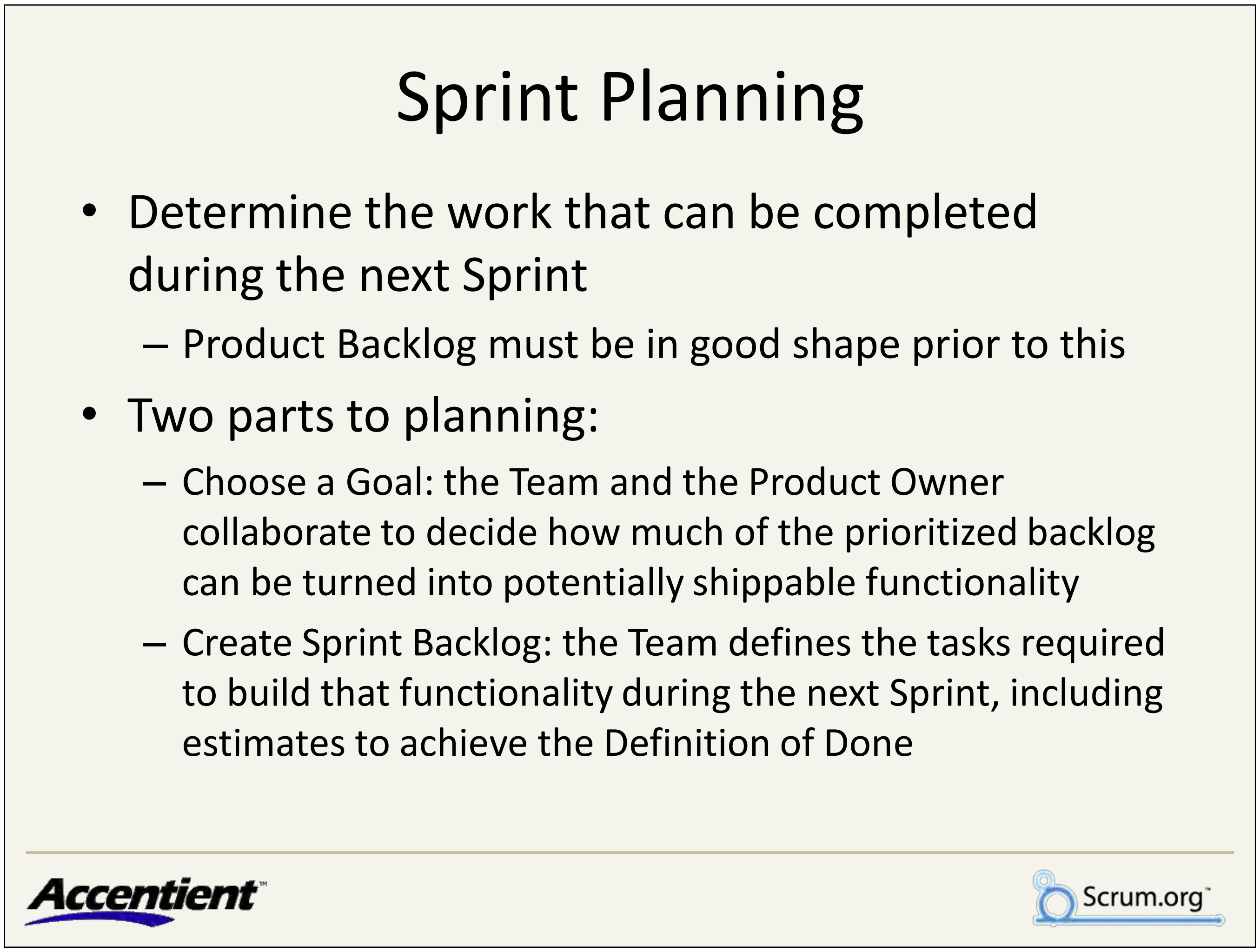
Formulas for prioritizing

There are many schools of thought on how to prioritize the product backlog. Let’s say that you have a Product Backlog containing 75 Product Backlog items.

Stack Rank	Spend $100	Formula
The Product Owner assigns each Product Backlog item a unique priority from 1-75	The Product Owner takes $100 and “spends” portions of it on each Product Backlog item until the money is gone.	The Product Owner provides a business value. The Product Owner & team provide a cost (points, days, dollars) as well as possible learning value. The team also identifies any technical dependencies. The priority is a factor of all of these inputs.
The team considers the items with the highest priorities first, taking into account any technical dependencies	The team considers the items with the highest $ first, taking into account any technical dependencies	The team considers the items with the highest ROI first

Tip

User stories should be written in business language and reflect business value. You should avoid technical stories if possible and if you must include them, try to record the business value. For example, instead of a creating a story “Add an index on Employees table” write the story as “Decrease system resources/time when users view thousands of Employees”. The associated task would be to “Add an index on the Employees table”.



Sprint Planning

- Determine the work that can be completed during the next Sprint
  - Product Backlog must be in good shape prior to this
- Two parts to planning:
  - Choose a Goal: the Team and the Product Owner collaborate to decide how much of the prioritized backlog can be turned into potentially shippable functionality
  - Create Sprint Backlog: the Team defines the tasks required to build that functionality during the next Sprint, including estimates to achieve the Definition of Done

Sprint Planning meeting

The Sprint Planning meeting is when the iteration is planned. There are two parts to the Sprint Planning Meeting: the “What?” part and the “How?” part. Some Scrum Teams combine the two. In the first part, the Scrum Team addresses the question of “What?” Here, the Product Owner presents the top priority Product Backlog to the team. They work together to figure out what functionality is to be developed during the next Sprint. The input to this meeting is the Product Backlog, the latest increment of product, the capacity of the team, and past performance of the team. The amount of backlog the team selects is solely up to the team. Only the team can assess what it can accomplish over the upcoming Sprint.

How long should the meeting be?

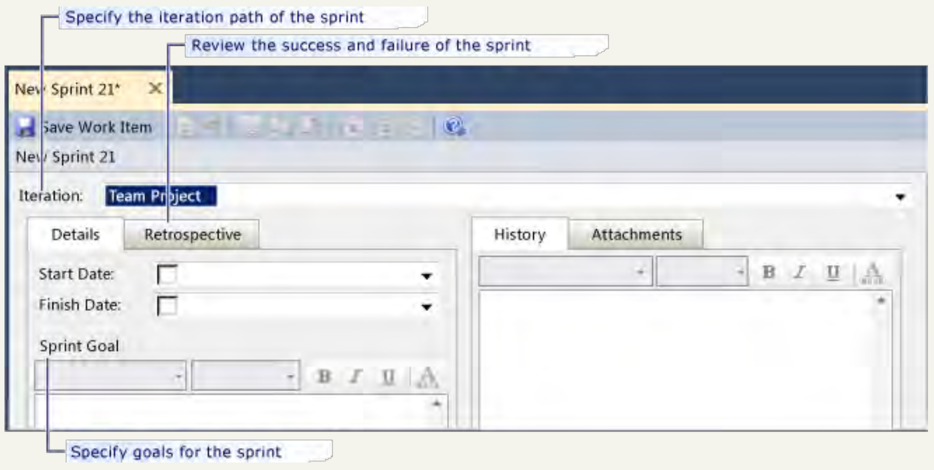
The Sprint Planning meeting is a time-boxed meeting. The actual length is a product of the Sprint length:

- For a 4 week (20 working days) Sprint, your meeting should be 8 hours
- For a 3 week (15 working days) Sprint, your meeting should be 6 hours
- For a 2 week (10 working days) Sprint, your meeting should be 4 hours

For shorter Sprints, allocate approximately 5% of the total Sprint length to the meeting.

Sprint Work Item Type

- Provides a way for your team to capture additional metadata about the Sprint
  - Start date, finish date, goal, and retrospective



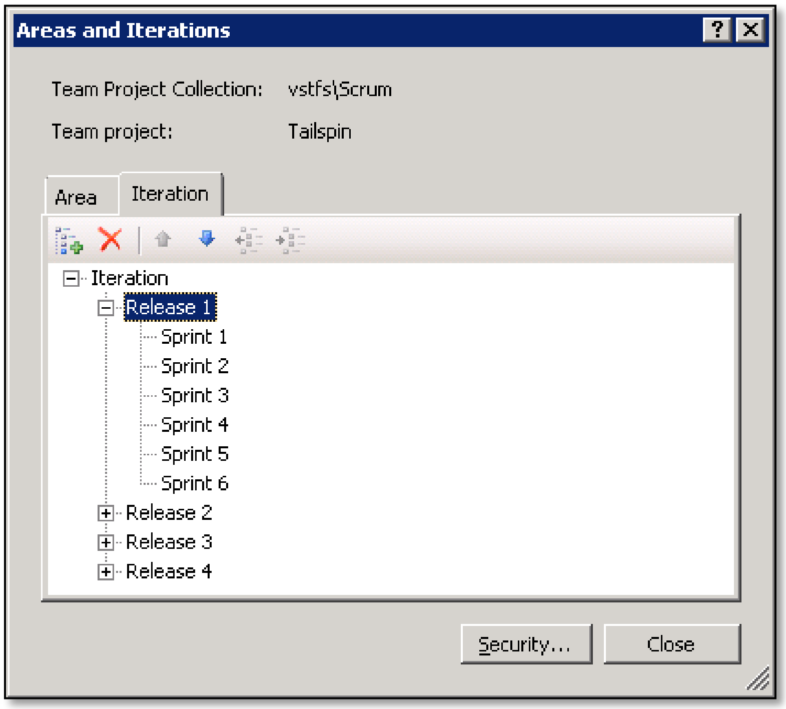
Accentient™ Scrum.org™

Sprint work item type

By defining and managing Sprint work items, a team can capture the Sprint Goal, the start and end dates, and the retrospective notes for each Sprint in a project. During the Sprint Planning meeting, the Scrum Team determines the Sprint Goal and the number of Product Backlog items that team members can accomplish over the upcoming sprint. Your ScrumMaster makes sure that the Sprint Goal remains constant throughout the Sprint. At the end of each Sprint, your team discusses what went well and what did not go well during the Sprint and then decides what the team can do differently to make the next sprint more effective. The Retrospective tab on the Sprint work item gives you the ability to capture this information.

Mapping iterations to releases and Sprints

When you first create a team project, it contains a default set of iterations: releases 1-4 and Sprints 1-6 within each. You can leave this hierarchy as is, or make changes to it such as renaming, renumbering, or removing the nodes.


Areas and Iterations

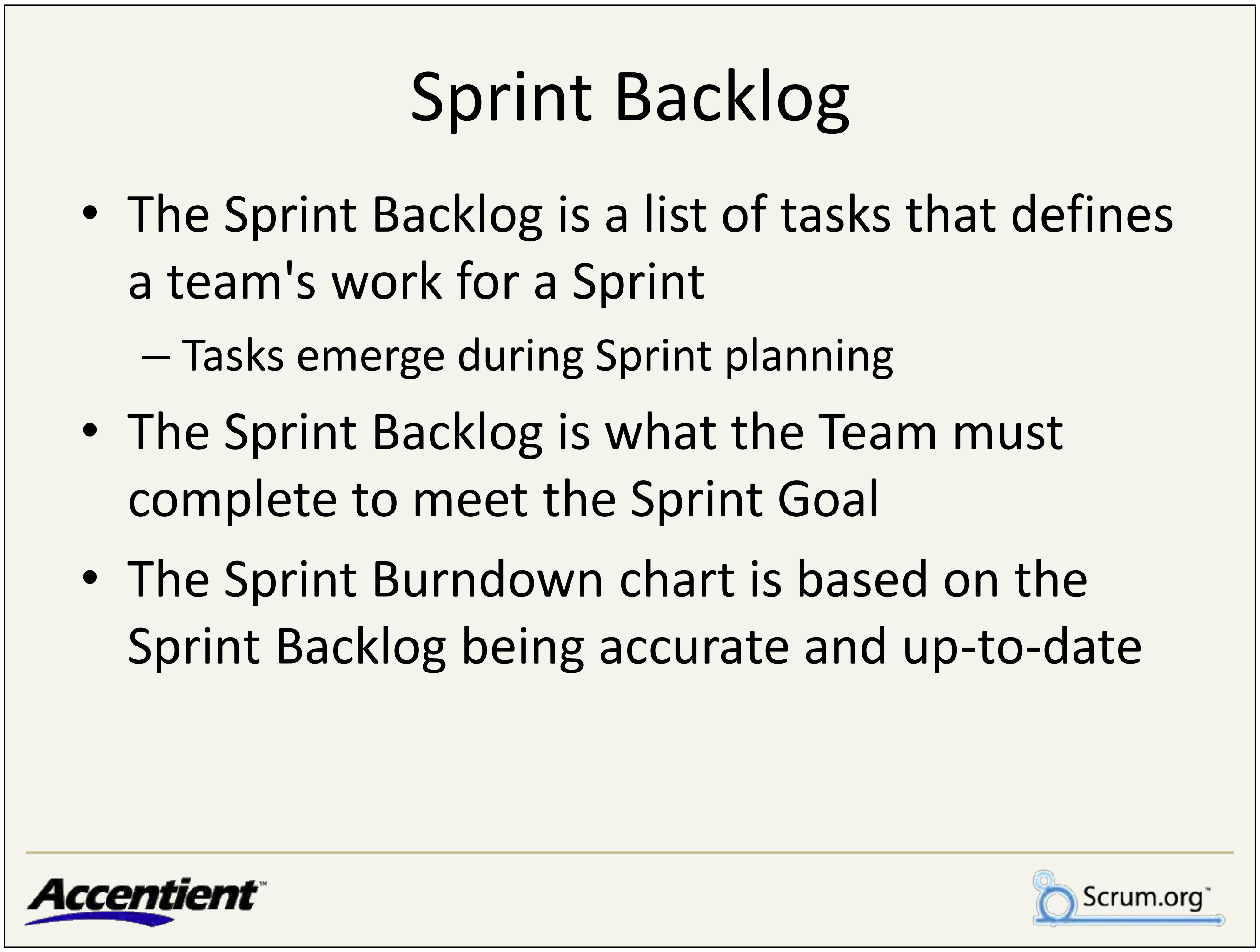
Team Project Collection: vstfs\Scrum

Team project: Tailspin

Keep in mind that the template also creates 24 accompanying Sprint work items that you will also need to manage. You can run the *All Sprints* team query to see these work items.

Iteration related activities

Activity	Tasks
Adding a new release	In the Areas and Iterations dialog, add a new, top-level Iteration node; no Sprint work item needs to be created
Renaming a release	In the Areas and Iterations dialog, rename the release node; all Sprint work items for that release will be renamed
Deleting a release	In the Areas and Iterations dialog, delete the node (and all Sprint nodes underneath it); you may be “orphaning” Sprint work items which you can either ignore or destroy using witadmin.exe
Adding a new Sprint	In the Areas and Iterations dialog, add a new, bottom-level Iteration node; next, create a new Sprint work item for that iteration and specify any additional Sprint details
Renaming a Sprint	In the Areas and Iterations dialog, rename the node; the associated Sprint work item will be renamed
Moving a Sprint	In the Areas and Iterations dialog, drag a (lower-level) Sprint node to a different (higher-level) release node; you may have to rename the node after moving it
Deleting a Sprint	In the Areas and Iterations dialog, delete the node; you may be “orphaning” a Sprint work item which you can either ignore or destroy using witadmin.exe



Sprint Backlog

- The Sprint Backlog is a list of tasks that defines a team's work for a Sprint
  - Tasks emerge during Sprint planning
- The Sprint Backlog is what the Team must complete to meet the Sprint Goal
- The Sprint Burndown chart is based on the Sprint Backlog being accurate and up-to-date

Accentient™ Scrum.org™

Sprint Backlog

The tasks are what the team has defined as being required to turn committed Product Backlog items into system functionality. The task sizes are estimated by the team and are ideally, no more than a day. The Sprint Backlog should be updated by the team members at least daily. Everything that gets done should be visible on the backlog. Emergent tasks are added by the team to the backlog throughout the Sprint.

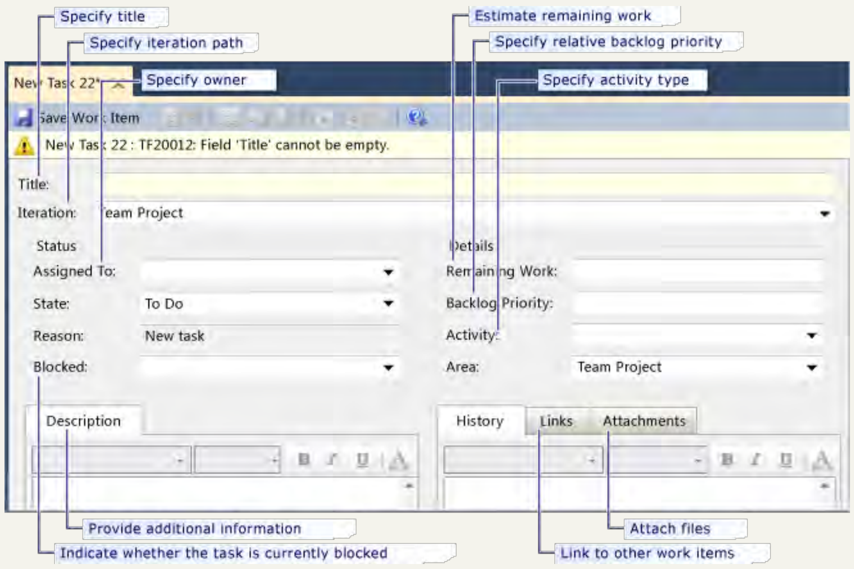
The act of decomposing work implies that the team has a consensus about how the work will be accomplished. This requires a reasonably clear understanding of the design from both a UX perspective and an architectural perspective. Consensus on the approach is key.

Signing-up for Work != Assigning Work

Don't think in terms of "assignment". Scrum work is not assigned. Team members sign-up for their own work. The team collectively organizes the work and individuals take responsibility for the appropriate parts. Remember that there is no success unless the whole team succeeds.

Task Work Item Type

- Represents the detailed work the team must do in order to achieve their Sprint Goal



New Task 22 : TF20012: Field 'Title' cannot be empty.

Title:
Iteration: Team Project

Status
Assigned To:
State: To Do
Reason: New task
Blocked:

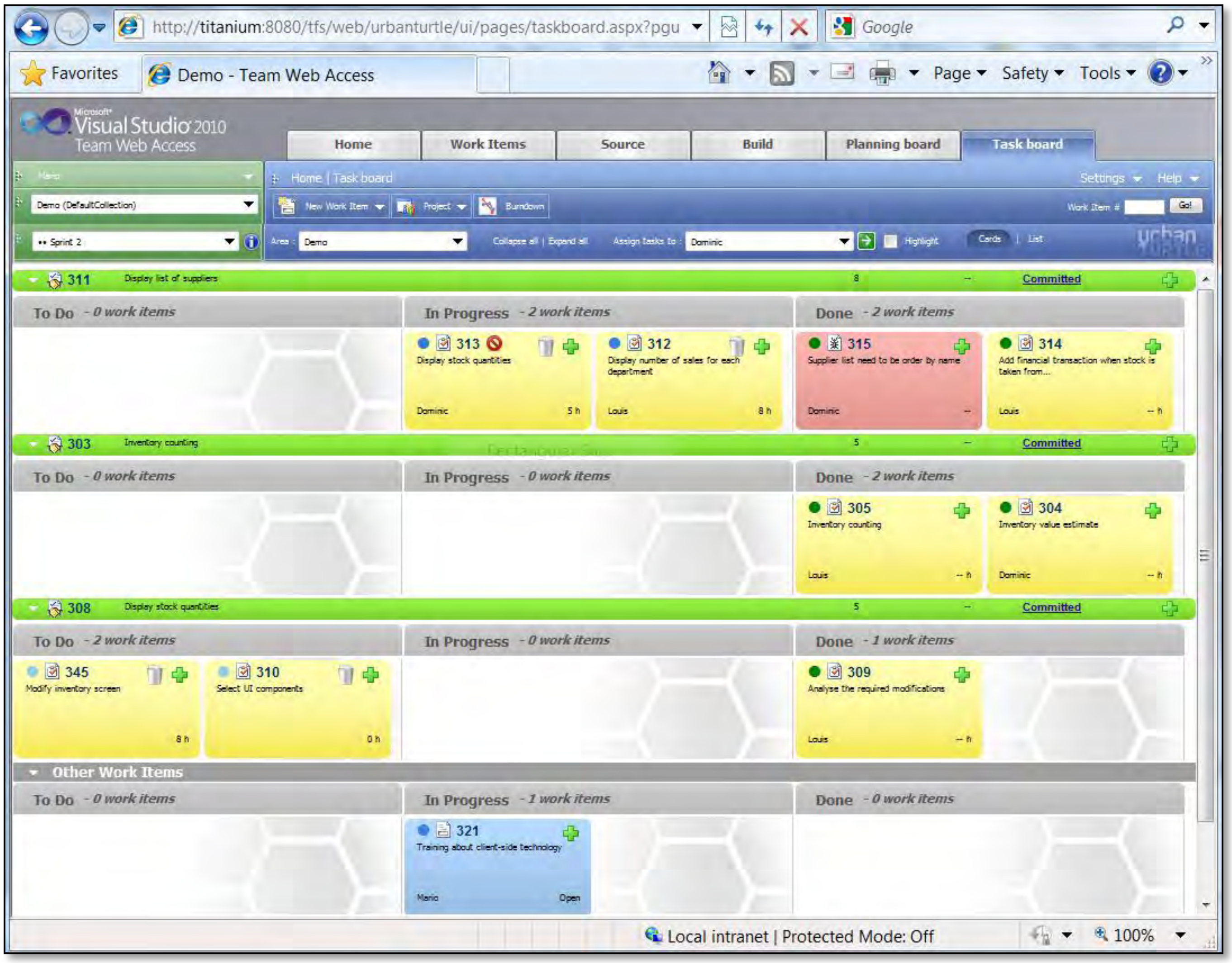
Details
Remaining Work:
Backlog Priority:
Activity:
Area: Team Project

Task work item type

By defining and managing task work items, a team can track and report on the detailed work that it must accomplish when implementing a Product Backlog Item. Teams typically forecast work and define tasks at the start of each Sprint, and each team member performs a subset of those tasks. Tasks can include design, development, testing, documentation, and other types of activities. For example, a developer can define tasks to implement Product Backlog Items and a tester can define tasks to write and run test cases.

Urban Turtle by Pyxis

Urban Turtle is a web-based Scrum tool that integrates with Team Foundation Server and Visual Studio Scrum. It enables a team to plan and track their backlog items and manage a Sprint using slick, web-based, drag and drop task and planning boards.



311 Display list of suppliers — 8 — Committed

To Do - 0 work items | In Progress - 2 work items | Done - 2 work items

313 Display stock quantities — Dominic 5 h
312 Display number of sales for each department — Louis 8 h
315 Supplier list need to be order by name — Dominic --
314 Add financial transaction when stock is taken from... — Louis -- h

303 Inventory counting — 5 — Committed

To Do - 0 work items | In Progress - 0 work items | Done - 2 work items

305 Inventory counting — Louis -- h
304 Inventory value estimate — Dominic -- h

308 Display stock quantities — 5 — Committed

To Do - 2 work items | In Progress - 0 work items | Done - 1 work items

345 Modify inventory screen — 8 h
310 Select UI components — 0 h
309 Analyse the required modifications — Louis -- h

Other Work Items

To Do - 0 work items | In Progress - 1 work items | Done - 0 work items

321 Training about client-side technology — Mario Open

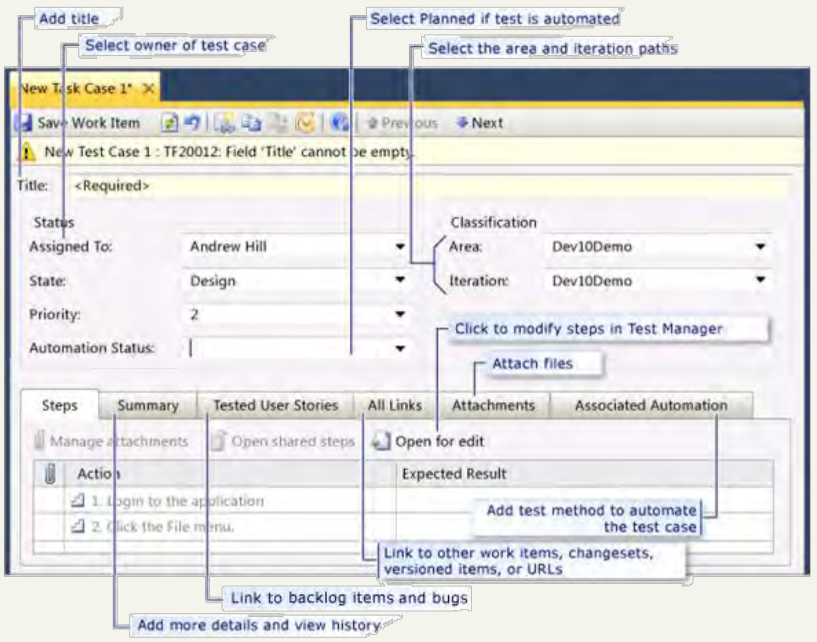
Main features

- Integrates with Team Web Access
- Planning Board enables grooming of the Product Backlog and planning the Sprint
- Task Board allows tracking tasks and simplifies the daily Scrum
- Works with most TFS templates: Visual Studio Scrum v1.0, MSF Agile v5.0, EMC's Scrum for Team System v3.0, or your own custom process template

For more information about Urban Turtle, please visit www.urbanturtle.com.

Test Case Work Item Type

- Used to define both manual and automated tests


Add title
Select owner of test case
Select Planned if test is automated
Select the area and iteration paths
Click to modify steps in Test Manager
Attach files
Add test method to automate the test case
Link to other work items, changesets, versioned items, or URLs
Link to backlog items and bugs
Add more details and view history
New Test Case 1*
Save Work Item
Previous
Next
Steps
Summary
Tested User Stories
All Links
Attachments
Associated Automation
Manage attachments
Open shared steps
Open for edit

New Test Case 1 : TF20012: Field 'Title' cannot be empty

Title: <Required>

Status

Assigned To: Andrew Hill

State: Design

Priority: 2

Automation Status:

Classification

Area: Dev10Demo

Iteration: Dev10Demo

Action	Expected Result
1. Login to the application	
2. Click the File menu.	

Accentient™

Scrum.org™

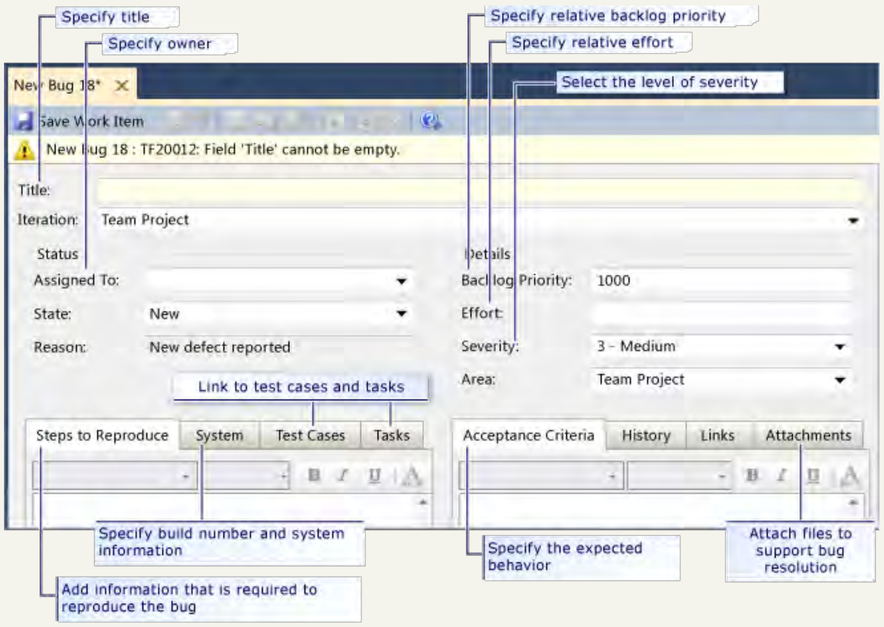
Test Case work item type

A team uses test cases to define both manual and automated tests that can be managed and run using Microsoft Test Manager. Test cases allow a team to further define the acceptance criteria of a Product Backlog Item or Bug in the form of executable and verifiable acceptance test steps. Test cases may be defined at a high level prior to the Sprint or during the Sprint Planning meeting and they will continue to emerge throughout the Sprint, especially as the test steps and test recordings are created and associated.

Acceptance Test-Driven Development (ATDD) occurs when acceptance criteria is turned into automated acceptance tests which then drive a traditional (unit) Test-Driven Development (TDD) process. This process ensures the team has an automated mechanism to decide whether the software meets the requirements. With ATDD, the team has a specific target to satisfy (the acceptance tests) which keeps them continuously focused on what the Product Owner really wants from that Product Backlog Item or bug fix.

Bug Work Item Type

- Represents a defect in the product


Specify title
Specify owner
Specify relative backlog priority
Specify relative effort
Select the level of severity
New Bug 18*
Save Work Item
Link to test cases and tasks
Steps to Reproduce
System
Test Cases
Tasks
Acceptance Criteria
History
Links
Attachments
Specify build number and system information
Add information that is required to reproduce the bug
Specify the expected behavior
Attach files to support bug resolution

New Bug 18 : TF20012: Field 'Title' cannot be empty.

Title:

Iteration: Team Project

Status

Assigned To:

State: New

Reason: New defect reported

Details

Backlog Priority: 1000

Effort:

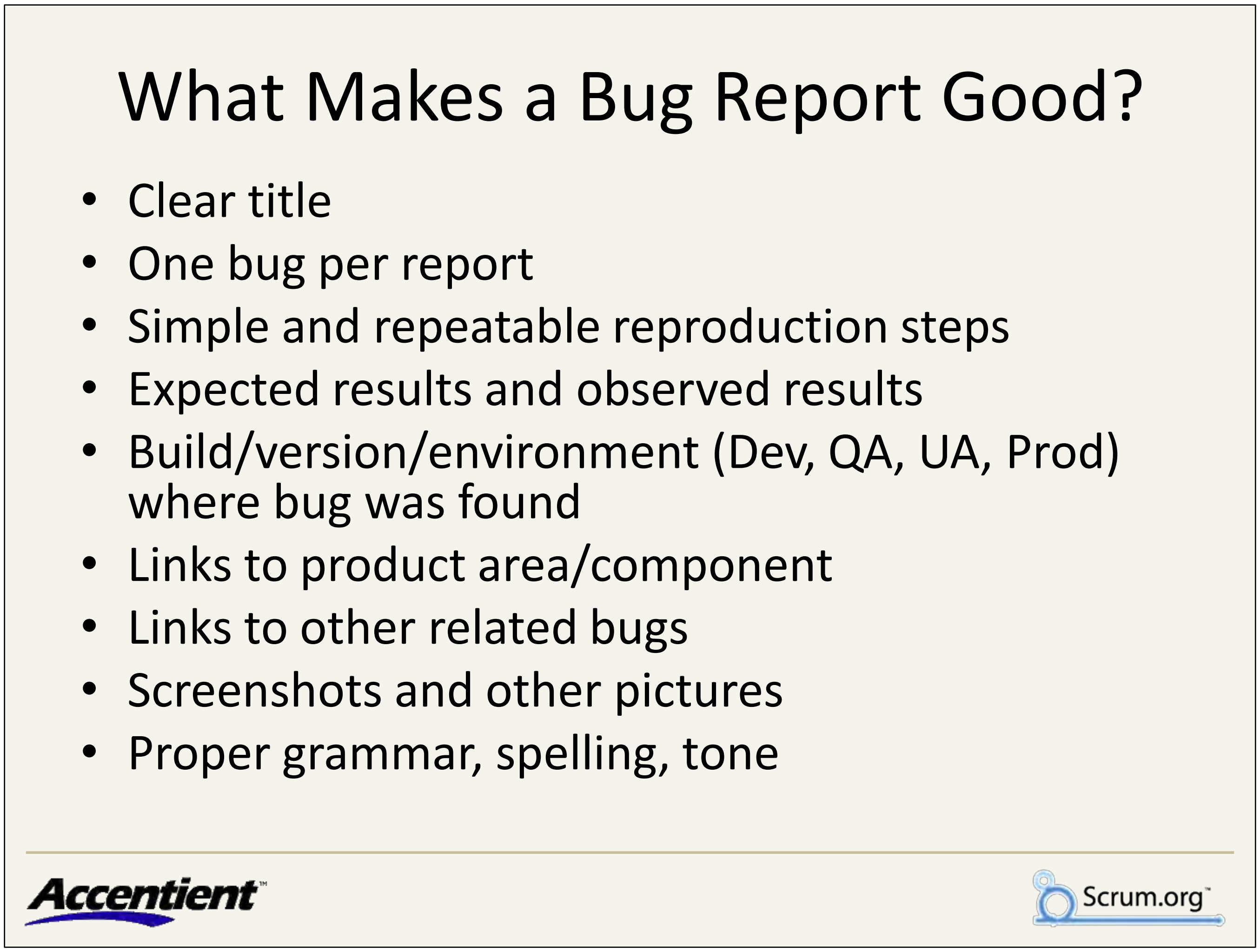
Severity: 3 - Medium

Area: Team Project

Accentient™ Scrum.org™

Bug work item type

A bug communicates that a potential problem exists in the code that your team is developing or has already shipped. By defining and managing bug work items, your team can track defects in the product and prioritize efforts to resolve them. When you create a bug work item, you want to accurately report the problem in a way that helps the reader to understand the full impact of the problem. The steps to reproduce the bug should also be listed so that other members of the team can more easily reproduce the behavior.



What Makes a Bug Report Good?

- Clear title
- One bug per report
- Simple and repeatable reproduction steps
- Expected results and observed results
- Build/version/environment (Dev, QA, UA, Prod) where bug was found
- Links to product area/component
- Links to other related bugs
- Screenshots and other pictures
- Proper grammar, spelling, tone

Accentient™
Scrum.org™

Bug reports

A bug report is just that – it's a reporting of a bug or other unwanted behavior in the system. In order to write a good bug report, it must contain enough information for the team, including the Product Owner, to understand it and gauge its impact on the business.

Clear title

A good, clear title is a must, as the developer should be able to grasp the essence of the bug report from the title alone. If you have a large work item repository, having a clear title will help the team assign bug reports to the correct team member without even reading the whole reports.

One bug per report

Only one bug should be listed per report. If you document more than one bug, some of them may be overlooked. Atomic bug tracking helps in the same way atomic testing does – it provides a very precise understanding of what's working and what isn't.

Triage

Before reporting a bug, the team should ensure that it is a valid bug, and not a by-design behavior of the system, a training issue, or something that has already been reported. This identifying and sorting process is known as bug triage. Triage also includes identifying the severity, frequency, risk, and other related factors. Triaging bugs can sometimes be a collaboration between team members, the Product Owner, and other business analysts and stakeholders who can elaborate on specific domain issues and risks.

When should you report a bug?

This is a very common question asked of teams using Scrum. The answer can depend on a lot of things, such as: How big is the bug? Can it be fixed right now? How important is quality to the product? Are there dedicated testers as part of the team? Regardless of your answers, there are basically two types of bugs: those found in code currently being developed (in-Sprint) and those found in code previously thought done (out-of-Sprint).

In-Sprint bugs

For stories that are currently in progress, here are some guidelines:

- Typically you want to fix all bugs discovered during the Sprint or else they could impact the team's ability to achieve its Sprint goal
- If it's a small bug (< n hours to fix) and won't impact the burndown, then just fix it
- If it's a larger bug (> n hours to fix) and won't impact the ability to achieve the Sprint goal, then create a Bug work item, associate a Task and have a team member perform the fix during the Sprint
- If it's a larger bug (> n hours to fix) and will impact the ability to achieve the Sprint goal, then create a Bug work item to be prioritized (by the Product Owner) and to be fixed in a later Sprint; you may not be able to achieve your Sprint goal
- If another team member finds the bug, then they should create a Bug work item and then the team decides if it can be fixed in the current Sprint or needs to be prioritized (by the Product Owner) to be fixed in a later Sprint, in which case you may not be able to achieve your Sprint goal
- The team decides how many hours "n" equals

  Note: the team may decide not to create a Bug work item, but rather just immediately pair-up/collaborate to identify, triage, and possibly resolve the bug.

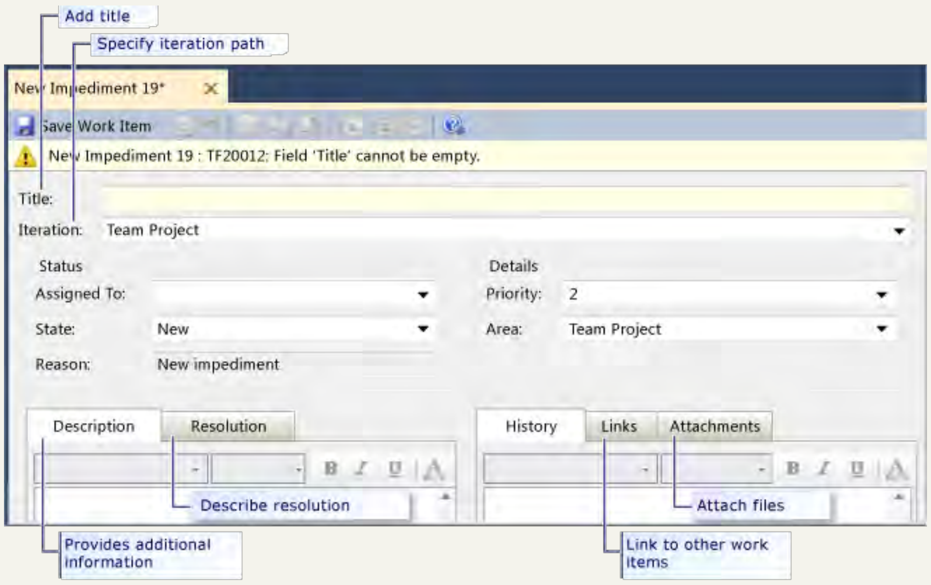
Out-of-Sprint bugs

For stories that the team had previously considered done, here are some guidelines:

- If the bug is not critical (i.e. a hotfix), then create a Bug work item to be prioritized and fixed in a later Sprint
- For a critical hotfix, do whatever needs to be done to get the fix into production, knowing that your current Sprint commitments may be impacted
- Adjust the team's capacity accordingly, especially if lots of hotfixes start occurring
- General tip: don't create an associated task to fix a bug until the Sprint in which the team commits to fix it

Impediment Work Item Type

- Represents an issue or problem that prevents the team from completing its tasks efficiently

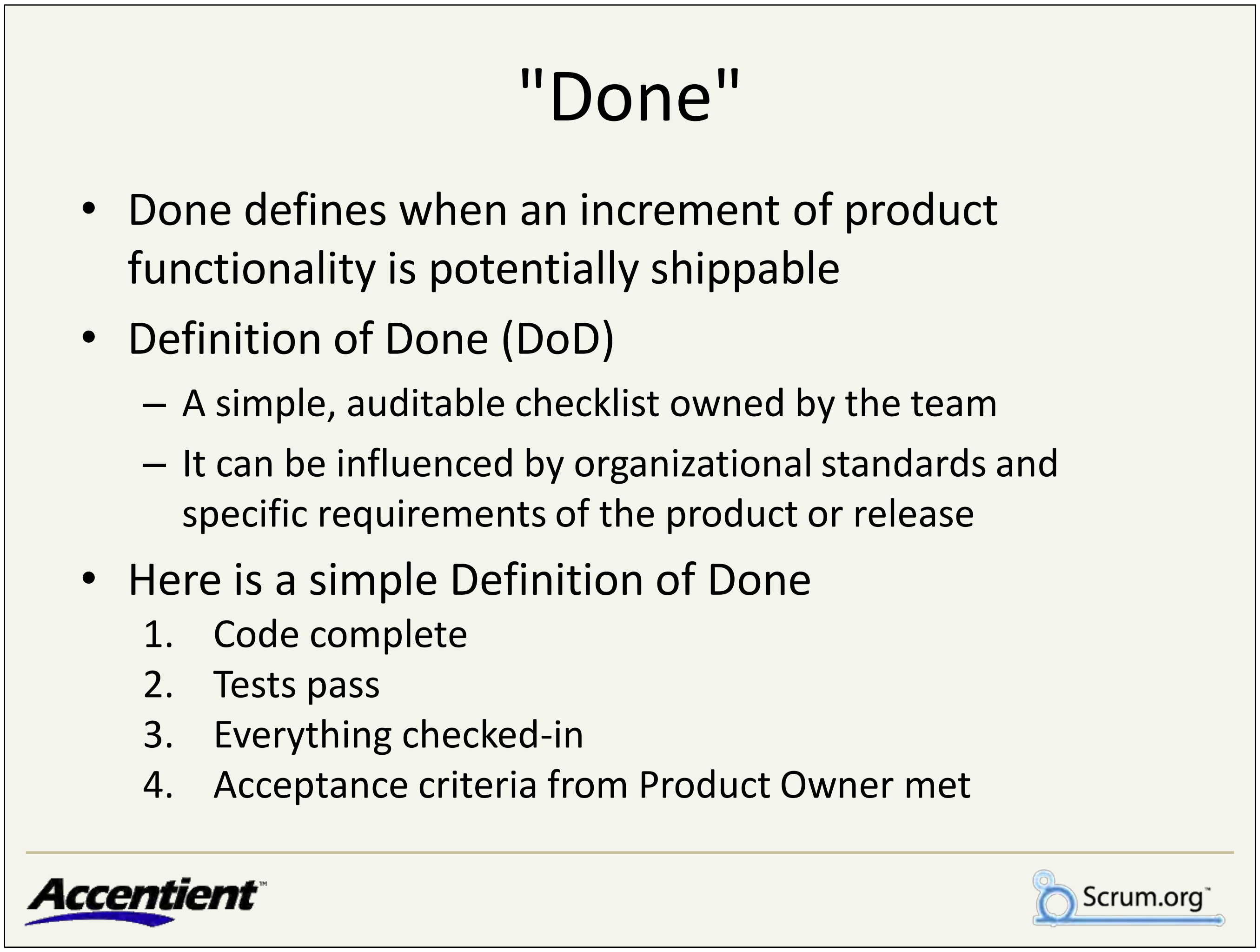


New Impediment 19 : TF20012: Field 'Title' cannot be empty.

Title:			
Iteration:	Team Project		
Status		Details	
Assigned To:		Priority:	2
State:	New	Area:	Team Project
Reason:	New impediment		

Impediment work item type

By defining and managing impediment work items, a team can identify and track problems that prevent it from completing tasks efficiently. Teams typically identify impediments during the daily Scrum, but they can be inspected and recorded at any time. The ScrumMaster is responsible for facilitating the resolution of these impediments and improving team productivity.



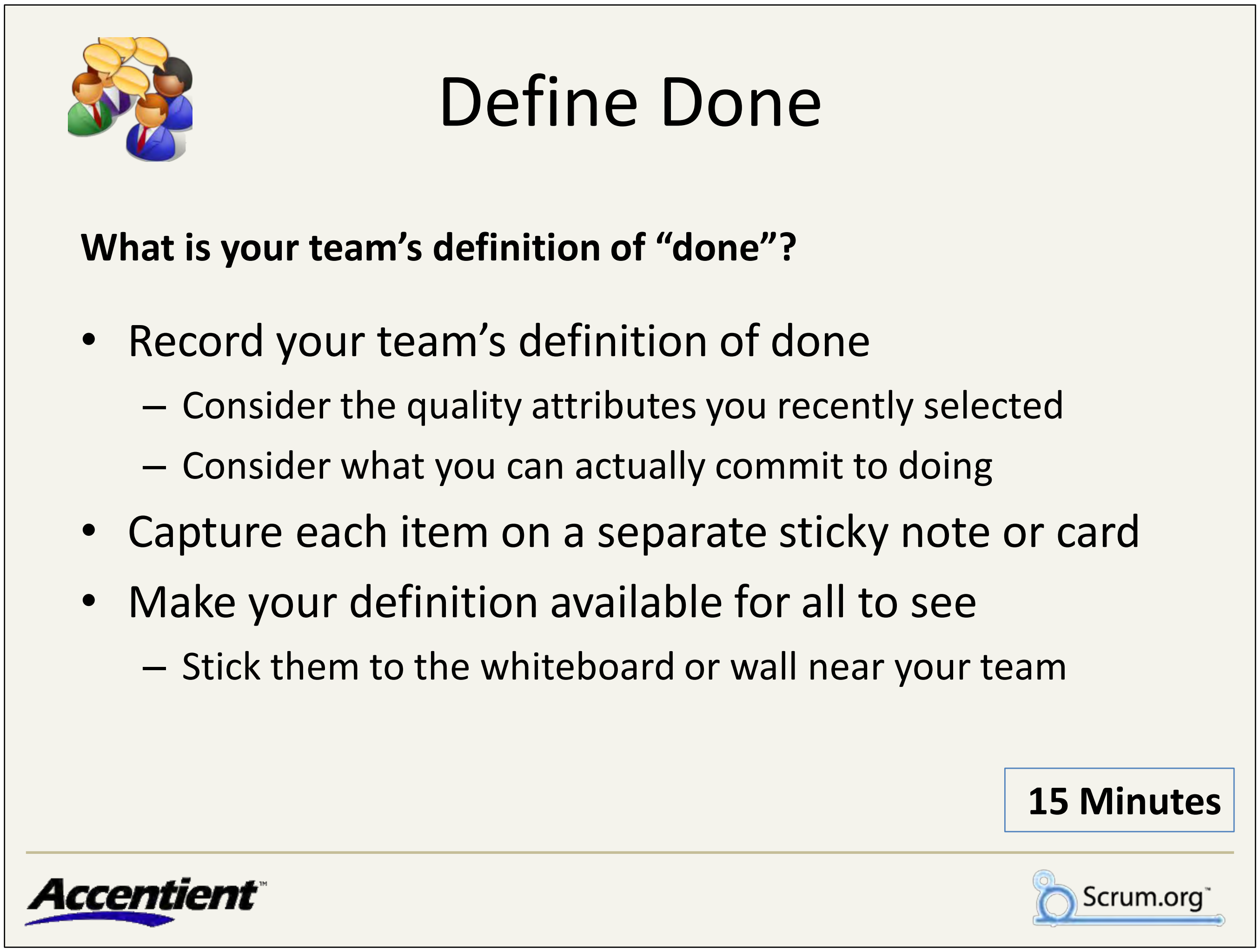
"Done"

- Done defines when an increment of product functionality is potentially shippable
- Definition of Done (DoD)
  - A simple, auditable checklist owned by the team
  - It can be influenced by organizational standards and specific requirements of the product or release
- Here is a simple Definition of Done
  1. Code complete
  2. Tests pass
  3. Everything checked-in
  4. Acceptance criteria from Product Owner met

Defining "Done"

A Scrum development team must be able to produce a completely "done" increment of software every Sprint. The definition of "done" can be different from enterprise to enterprise and product to product. The Product Owner must understand and agree to the done criteria. Done defines the current technical capability of the team, which means that over time "done" should include everything needed before deployment and that "not done" backlog items should not be demonstrated at Sprint Review. Most organizations define "done" to indicate "production-ready" functionality.

An explicit and concrete definition of done may seem small but it can be the most critical checkpoint of an Agile project. Without a consistent meaning of done, velocity cannot be estimated. Conversely, a common definition of done ensures that the increment produced at the end of Sprint is of high quality, with minimal defects. The definition of done is the soul of the entire Scrum process.



Define Done

What is your team's definition of "done"?

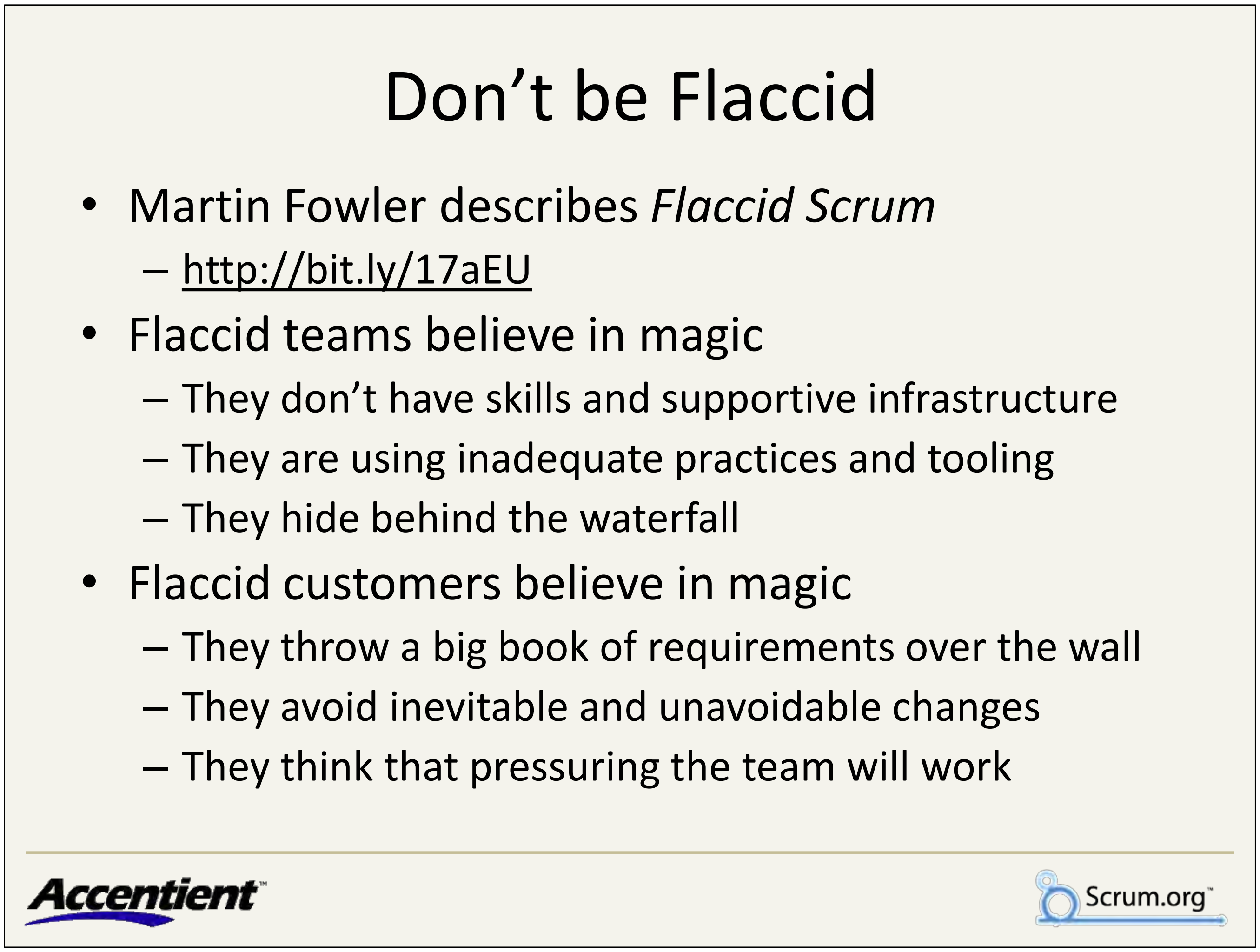
- Record your team's definition of done
  - Consider the quality attributes you recently selected
  - Consider what you can actually commit to doing
- Capture each item on a separate sticky note or card
- Make your definition available for all to see
  - Stick them to the whiteboard or wall near your team

15 Minutes

Accentient™ Scrum.org™

Activity: define "Done"

In this activity each team will brainstorm and define their done criteria. This will then be posted and reported by each team, so other teams can get a feel for the variety of items in every team's done list.



Don’t be Flaccid

- Martin Fowler describes *Flaccid Scrum*
  - http://bit.ly/17aEU
- Flaccid teams believe in magic
  - They don’t have skills and supportive infrastructure
  - They are using inadequate practices and tooling
  - They hide behind the waterfall
- Flaccid customers believe in magic
  - They throw a big book of requirements over the wall
  - They avoid inevitable and unavoidable changes
  - They think that pressuring the team will work

Accentient™ Scrum.org™

Flaccid Scrum

Scrum has been a very widely adopted Agile process, used for managing such complex work as systems development and development of product releases. When waterfall is no longer in place, however, a lot of long standing habits and dysfunctions have come to light. This is particularly true with Scrum, because transparency is emphasized in Scrum projects. Some of the dysfunctions include poor quality product and completely inadequate development practices and infrastructure. These arose because the effects of them couldn’t be seen very clearly in a waterfall project. In a Scrum project, the impact of poor quality caused by inadequate practices and tooling are seen in every Sprint. The primary habits that hinder us are flaccid developers and flaccid customers who believe in magic.

ScrumButs and Modifying Scrum

ScrumButs are reasons why teams can't take full advantage of Scrum to solve the problems and realize the benefits. Every Scrum role, rule, and timebox is designed to provide the desired benefits and address the problems. ScrumButs mean that Scrum has exposed a dysfunction that is contributing to the problem, but is too hard to fix. A ScrumBut retains the problem while modifying Scrum to make it invisible so that the dysfunction is no longer a thorn in the side.

A ScrumBut has a particular syntax:

(ScrumBut)(Reason)(Workaround)

Example:

"We use Scrum, but (the daily Scrum meetings are too much overhead) (so we only have them once a week, unless we need them more often)"

ScrumButs can come from many sources. Here are some examples:

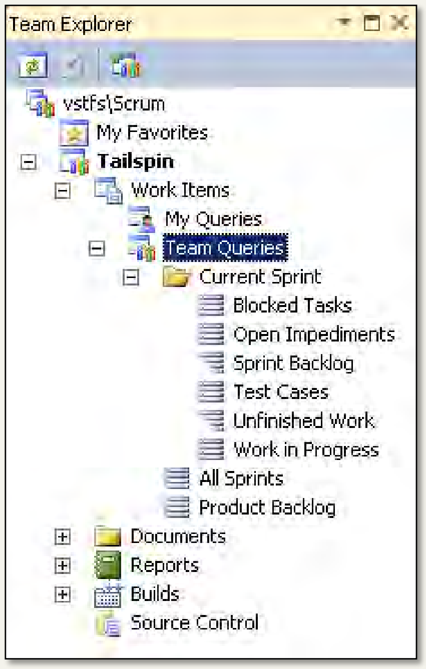
- The business doesn't want to be involved
- Everyone wants their features first and can't agree on a priority
- Teams don't know how to self-organize
- People aren't available to work on teams full-time
- Teams don't see a need for a daily Scrum
- Teams can't get a piece of functionality done in one Sprint
- Teams don't have the skills to "do" something
- Teams can't fit testing into the same Sprint as development
- The ScrumMaster tells the team what to do and how to do it
- Other managers can't stay out of a Sprint
- Important things come up that require interrupting the Sprint
- The Sprints can't start until all of the other groups do their up-front work
- Other groups are building hardware or using waterfall

Sometimes organizations make short term changes to Scrum to give them time to correct deficiencies. For example, "done" may not initially include regression and performance testing because it will take several months to develop automated testing. For these months, transparency is compromised, but restored as quickly as possible.

For more information on ScrumButs visit: http://bit.ly/i5QGta

Queries

- Product queries
  - All Sprints
  - Product Backlog
- Current Sprint queries
  - Blocked Tasks
  - Open Impediments
  - Sprint Backlog
  - Test Cases
  - Unfinished Work
  - Work in Progress

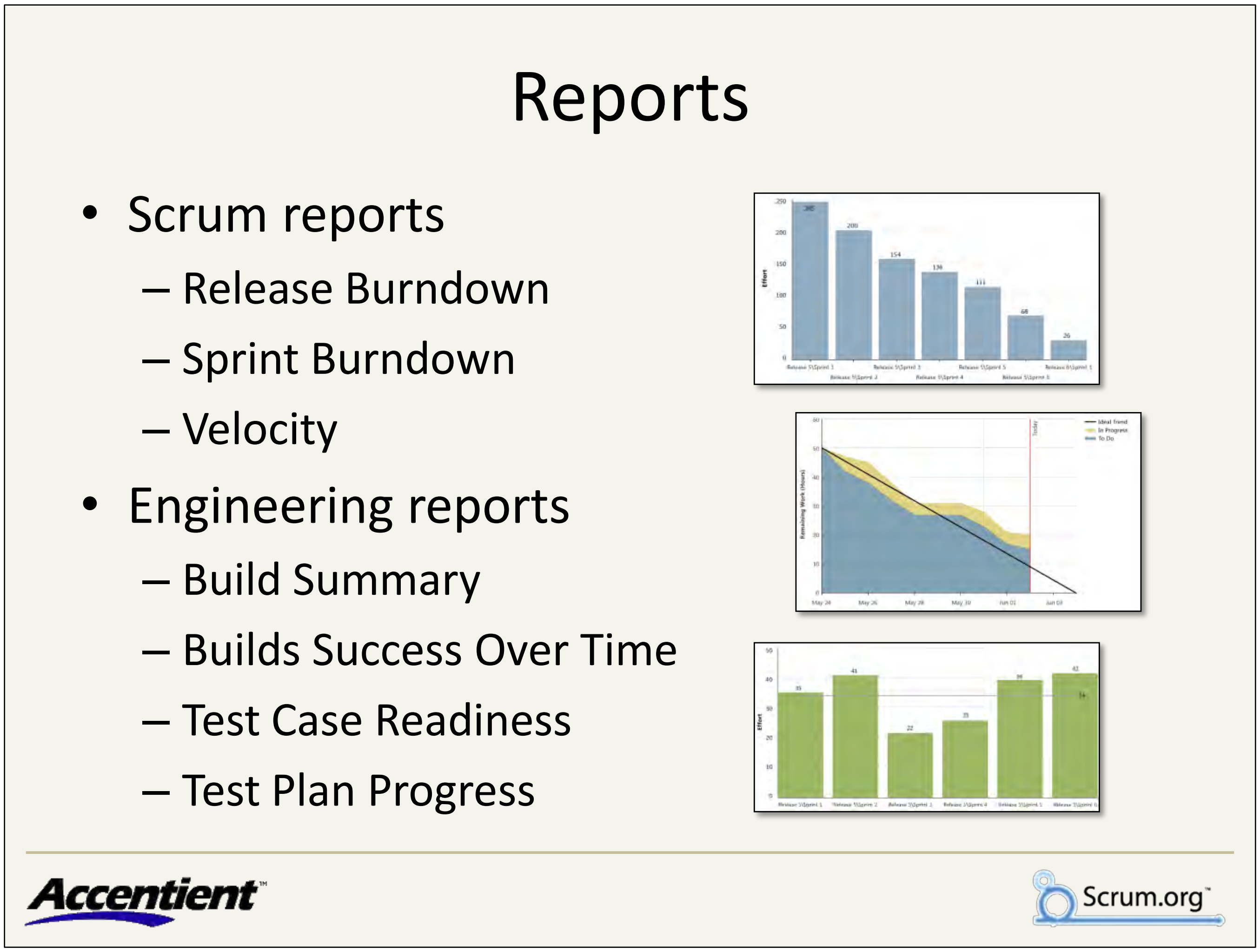


Accentient™ Scrum.org™

Queries

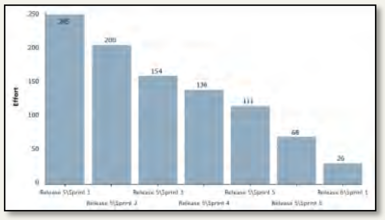
You can manage your workload more effectively by frequently identifying the Product Backlog Items, tasks, impediments, and other work items on which you want to take action. To more easily find all such work items in a particular team project, you can run the default queries or create your own. You can open these queries from Team Explorer by expanding your team project node, expanding Work Items, and then expanding Team Queries. Queries are also available from Team Web Access.

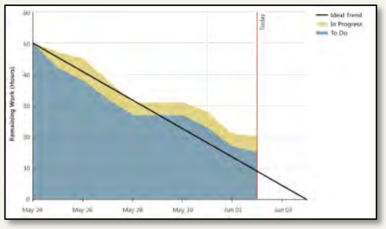
Your product owner can plan and track Product Backlog Items and bugs in the Product Backlog by using the Product Backlog query. Your team can arrange and track sprints by using the team query of All Sprints. You can find work items that are assigned to the current sprint by using the team queries that are listed under the Current Sprint folder.

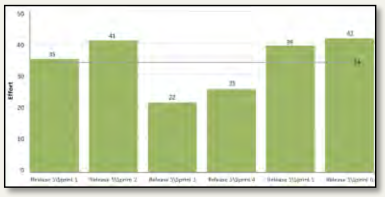


Reports

- Scrum reports
  - Release Burndown
  - Sprint Burndown
  - Velocity
- Engineering reports
  - Build Summary
  - Builds Success Over Time
  - Test Case Readiness
  - Test Plan Progress







Accentient™ Scrum.org™

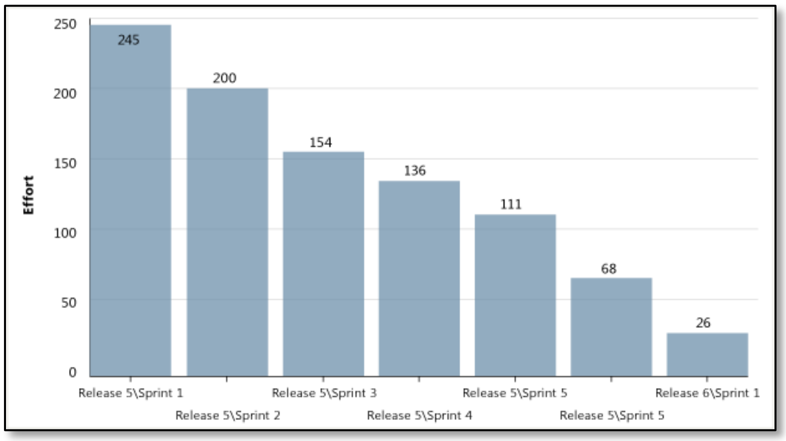
Reports

You can analyze the progress and quality of your project by using the reports in SQL Server Reporting Services. These reports are provided by the Visual Studio Scrum process template. These reports aggregate metrics from work items, version control, test results, and builds. These reports answer questions about the actual state of your project.

Visual Studio Scrum introduces three project management reports: Release Burndown, Sprint Burndown, and the Velocity report. It also includes four engineering-centric reports: Build Success Over Time, Build Summary, Test Case Readiness, and Test Plan Progress.

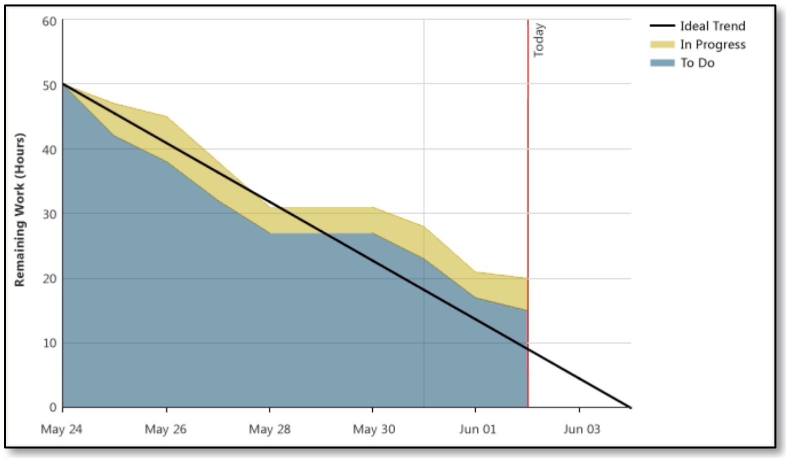
Release Burndown report

By reviewing the Release Burndown report, you can understand how quickly your team has delivered Product Backlog items and track how much work the team must still perform to complete a product release. The graph shows how much work remained at the start of each Sprint in a release. The source of the raw data is your product backlog. Each Sprint appears along the horizontal axis, and the vertical axis measures the effort that remained when each Sprint started. The amount of estimated effort on the vertical axis is in whatever unit that your Scrum Team has decided to use (for example, story points or hours).



Sprint Burndown report

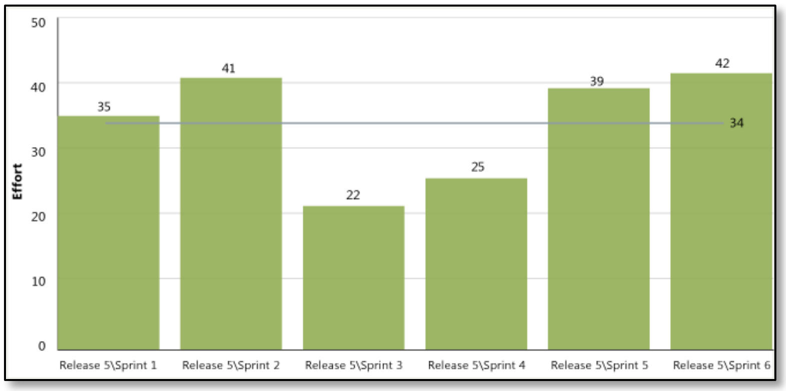
By reviewing the Sprint Burndown report, you can track how much work remains in a Sprint Backlog, understand how quickly your team has completed tasks, and predict when your team will achieve the goal or goals of the sprint. The graph shows how much work remained at the end of specified intervals during a Sprint. The source of the raw data is the Sprint Backlog. The horizontal axis shows days in a Sprint, and the vertical axis measures the amount of work that remains to complete the tasks in the Sprint. The work that remains is shown in hours.



The Ideal Trend line indicates an ideal situation in which the team burns down all of the effort that remains at a constant rate by the end of the sprint. The In Progress series shows how many hours remain for tasks that are marked as In Progress in a sprint. The To Do series shows how many hours remain for tasks that are marked as To Do in a sprint. Both the In Progress and the To Do series are drawn based on the actual progress of your team as it completes tasks.

Velocity Report

If your team has completed multiple Sprints, you can forecast release and product completion dates and plan future projects more accurately by reviewing the velocity report. Based on the velocity of previous Sprints that the report illustrates, you can track how much effort your team has reported as complete for each Sprint and estimate how much backlog effort your team can handle in future sprints if your team composition and Sprint duration stay constant.

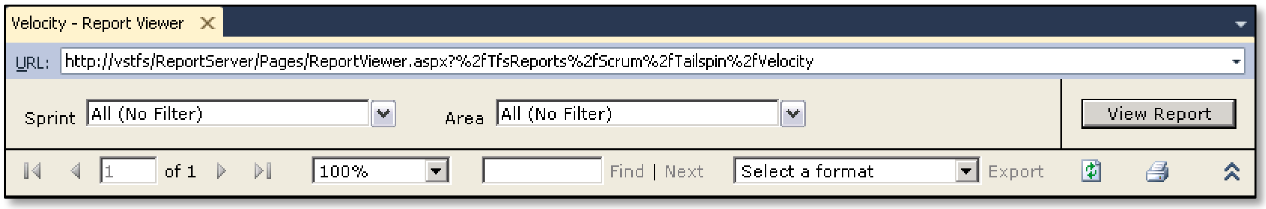


The graph shows the amount of effort that your team has reported as complete for each sprint. The source of the raw data is your product backlog. The horizontal axis represents sprints, and the vertical axis measures the backlog effort that your team has reported as complete. The vertical axis shows effort in whatever unit your team uses (for example, story points or hours). The graph also displays a horizontal line that represents the average velocity across all the sprints.

Running the reports

These reports are available from Team Explorer, Team Web Access, or directly from the respective SQL Server Reporting Services Report Manager site. To view these reports, you must be assigned or belong to a group that has been assigned the appropriate permission in SQL Server Reporting Services.

Many of the reports provide filters that you can use to specify contents to include in the report. Filters include Sprint, Area, time period, state, priority, etc. Each report will define its own set of filters and available values. These typically include an *All* value which indicates no filter be applied.



URL: http://vstfs/ReportServer/Pages/ReportViewer.aspx?%2fTfsReports%2fScrum%2fTailspin%2fVelocity

Sprint All (No Filter)

Area All (No Filter)

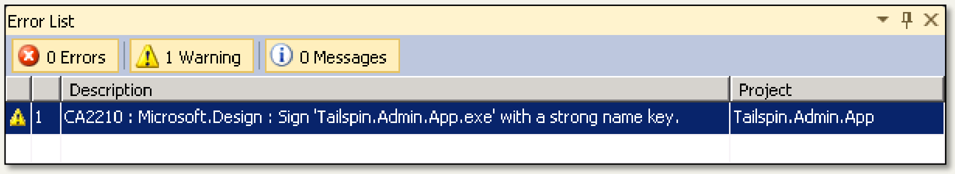
1 of 1 100% Select a format

.

What if you’re not really Done?

As a team, how would you handle this situation?

Your team believes that they have met their Sprint Goal and the Product Owner is quite happy with the increment’s functionality. One of the PBIs wasn’t completed according to your definition of done. It seems that a Code Analysis warning remains:



Error List

0 Errors | 1 Warning | 0 Messages

		Description	Project
⚠	1	CA2210 : Microsoft.Design : Sign 'Tailspin.Admin.App.exe' with a strong name key.	Tailspin.Admin.App

5 Minutes

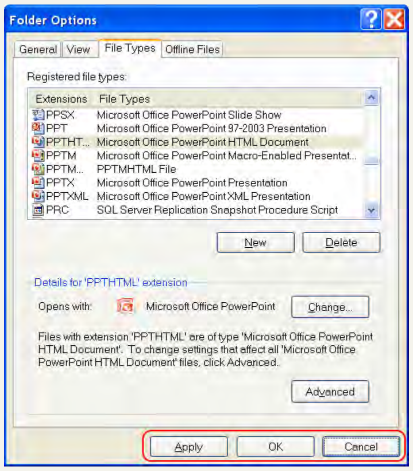
Accentient™ Scrum.org™

Activity: what if you’re not really done?

What about that dialog window?

As a team, how would you handle this situation?

Your Product Owner has accepted this dialog box as being fit for purpose, but you are concerned that the buttons at the bottom do not match the rest of those in your application or in Windows. The feature is done and has been accepted.



Registered file types:

Extensions	File Types
PPSX	Microsoft Office PowerPoint Slide Show
PPT	Microsoft Office PowerPoint 97-2003 Presentation
PPTHT...	Microsoft Office PowerPoint HTML Document
PPTM	Microsoft Office PowerPoint Macro-Enabled Presentat...
PPTM...	PPTMHTML File
PPTX	Microsoft Office PowerPoint Presentation
PPTXML	Microsoft Office PowerPoint XML Presentation
PRC	SQL Server Replication Snapshot Procedure Script

Details for 'PPTHTML' extension

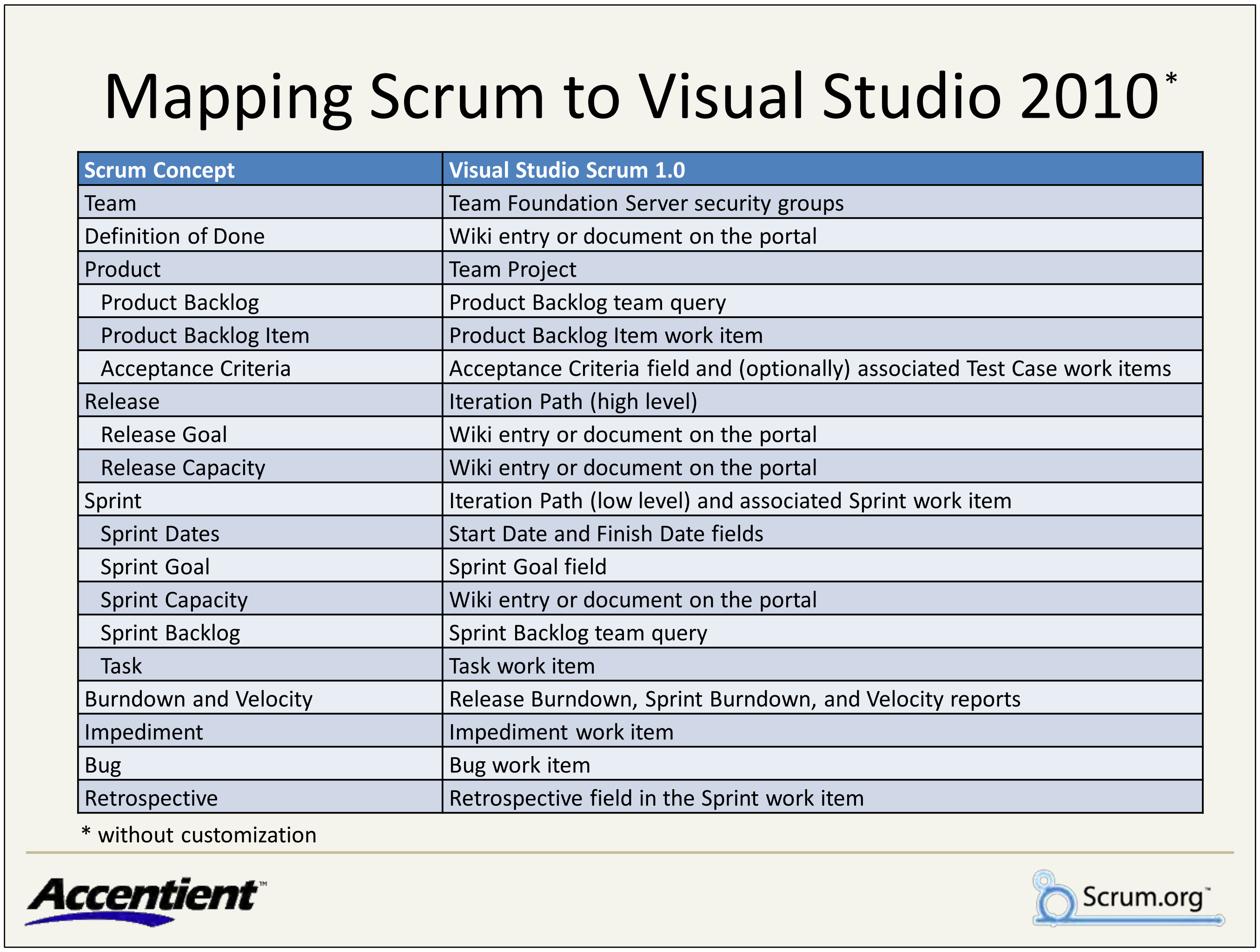
Opens with: Microsoft Office PowerPoint

Files with extension 'PPTHTML' are of type 'Microsoft Office PowerPoint HTML Document'. To change settings that affect all 'Microsoft Office PowerPoint HTML Document' files, click Advanced.

5 Minutes

Accentient™ Scrum.org™

Activity: what about that dialog window?



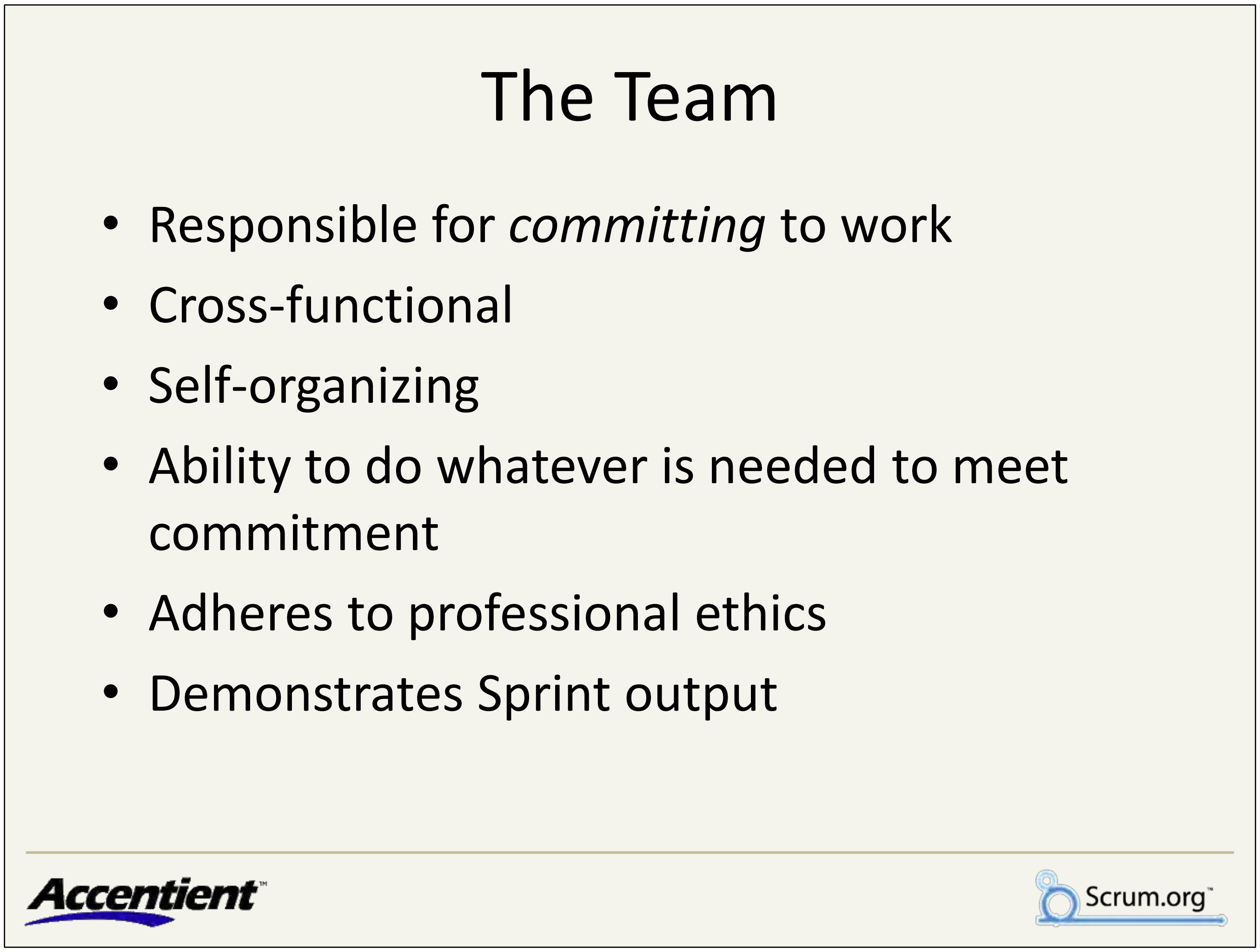
Mapping Scrum to Visual Studio 2010*

Scrum Concept	Visual Studio Scrum 1.0
Team	Team Foundation Server security groups
Definition of Done	Wiki entry or document on the portal
Product	Team Project
Product Backlog	Product Backlog team query
Product Backlog Item	Product Backlog Item work item
Acceptance Criteria	Acceptance Criteria field and (optionally) associated Test Case work items
Release	Iteration Path (high level)
Release Goal	Wiki entry or document on the portal
Release Capacity	Wiki entry or document on the portal
Sprint	Iteration Path (low level) and associated Sprint work item
Sprint Dates	Start Date and Finish Date fields
Sprint Goal	Sprint Goal field
Sprint Capacity	Wiki entry or document on the portal
Sprint Backlog	Sprint Backlog team query
Task	Task work item
Burndown and Velocity	Release Burndown, Sprint Burndown, and Velocity reports
Impediment	Impediment work item
Bug	Bug work item
Retrospective	Retrospective field in the Sprint work item

* without customization

Mapping Scrum to Visual Studio

The Visual Studio Scrum 1.0 template does a good job of mapping the Scrum concepts to Visual Studio 2010. There are some areas where the support is not as “first class” as it could be. An example would be in capturing a team’s definition of done. There is no proper place to record this, which leaves a team with the option of using Microsoft Word or Excel or the SharePoint wiki to capture the information. Microsoft is committed to increasing the support for Scrum. Future versions of Visual Studio will include more first class support for the Scrum concepts and better tooling for planning and managing releases and Sprints.



The Team

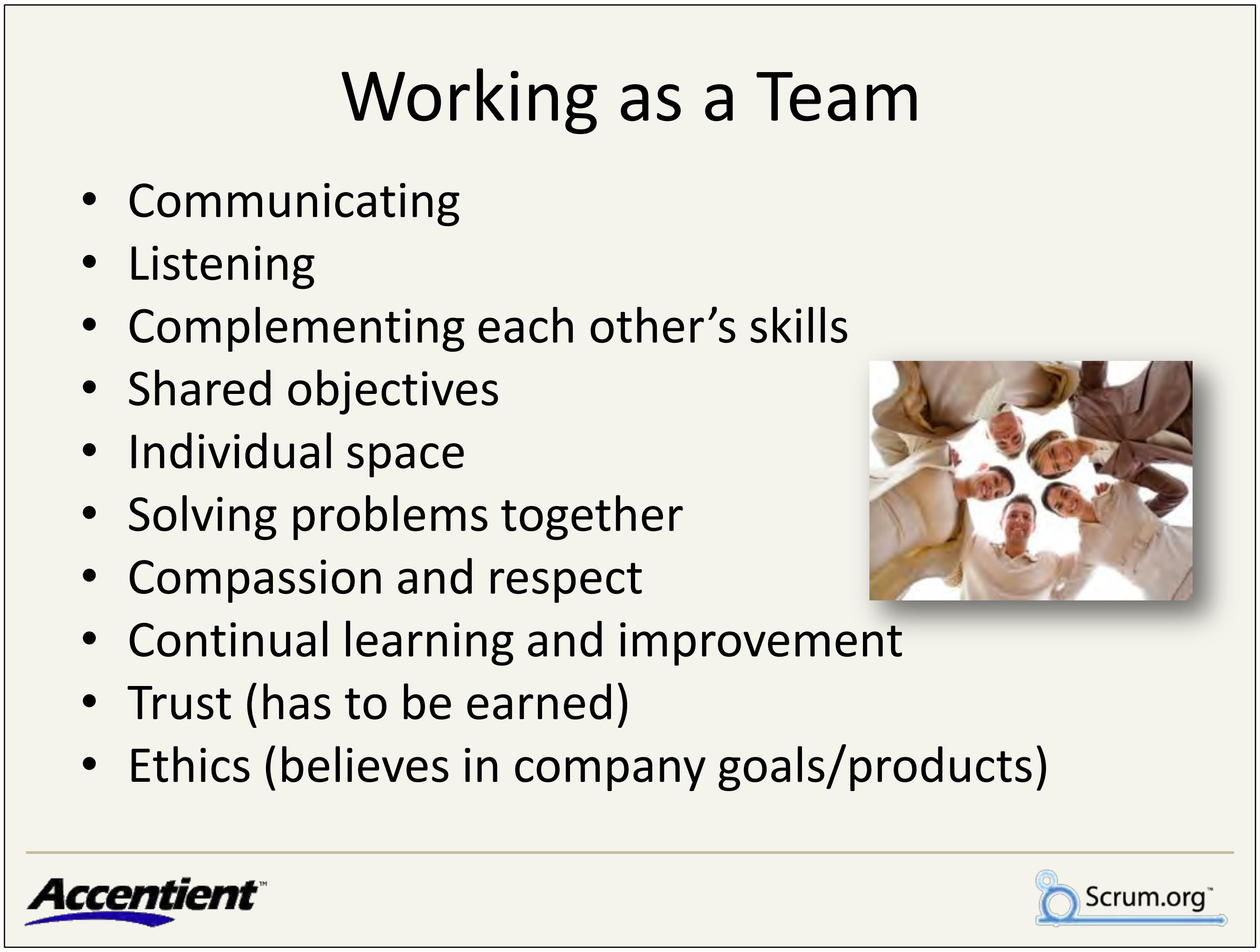
- Responsible for *committing* to work
- Cross-functional
- Self-organizing
- Ability to do whatever is needed to meet commitment
- Adheres to professional ethics
- Demonstrates Sprint output

Accentient™ Scrum.org™

The Team

This is a cross-functional group of people with all the different skills that are needed to turn requirements into something that is an increment of potentially shippable functionality. Team members will need to develop the skills of business analyst, designer, tester, developer, technical writer, etc. Team members basically need to know all the skills necessary to turn the requirements into something that the organization defines as “done”. The team commits to the Product Owner what they'll accomplish every iteration and then self-organizes to deliver it. At the end of the iteration they show it to the Product Owner and then the Product Owner can decide what to do next.

There are often centers of excellence in software engineering organizations that group people of unique, highly specialized skills, like usability engineering, database experts, security experts and systems architects. When these people's skills are needed they should become part of the team. That is they're not outside experts who simply give advice but they become part of the team for those iterations when they're needed to build architecture, to build infrastructure or to work on the database. They are doing the work while mentoring, monitoring and guiding the other team members. The key differentiator here is that they are no longer a chicken. We get them in at the start of an iteration and they commit along with the team to get something done by the end of that iteration. So for a short time they become a pig.



Working as a Team

- Communicating
- Listening
- Complementing each other's skills
- Shared objectives
- Individual space
- Solving problems together
- Compassion and respect
- Continual learning and improvement
- Trust (has to be earned)
- Ethics (believes in company goals/products)

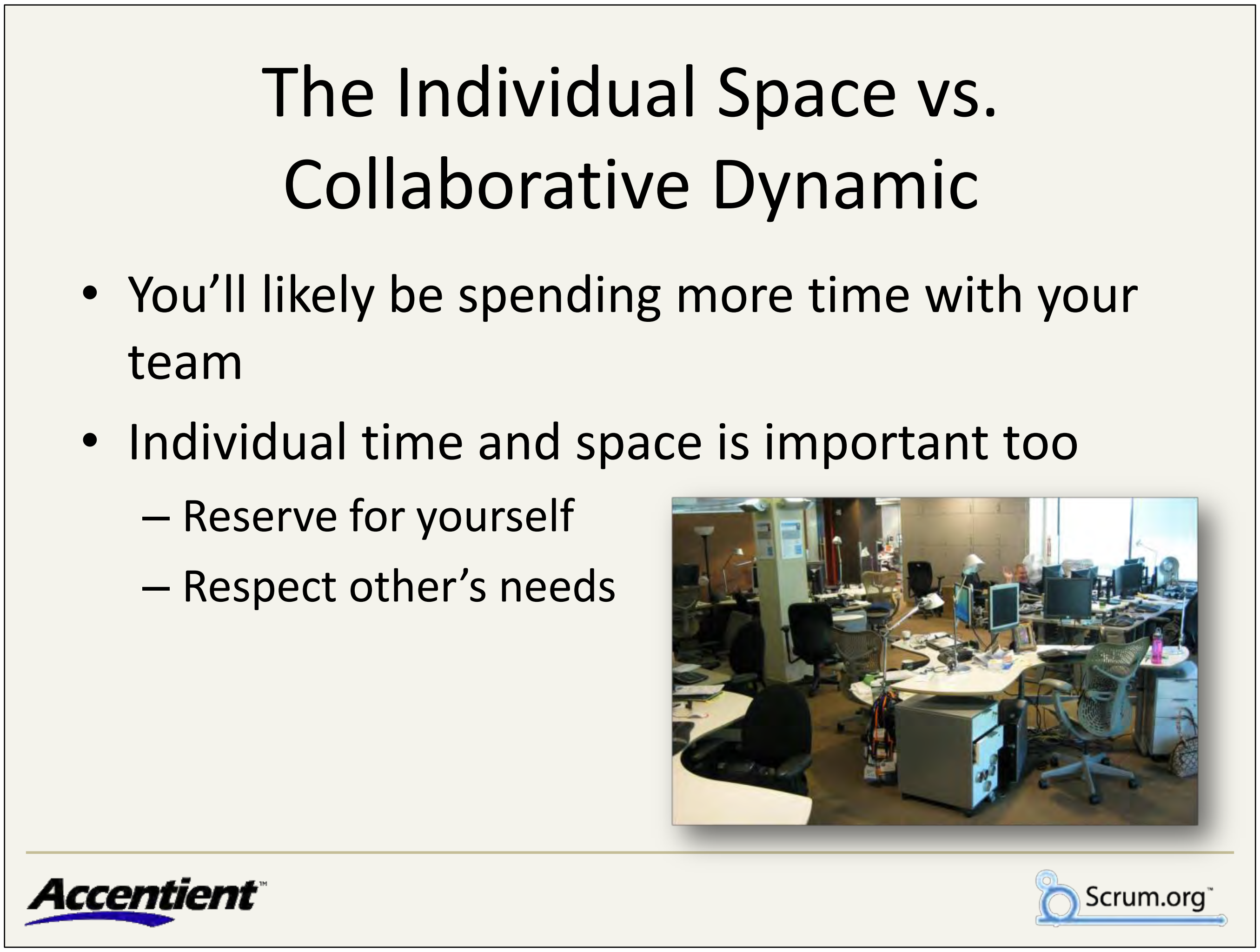
Working as a team

We recap and build on what we have covered earlier in the week (i.e. Active Listening)

7 Habits of Highly Effective People

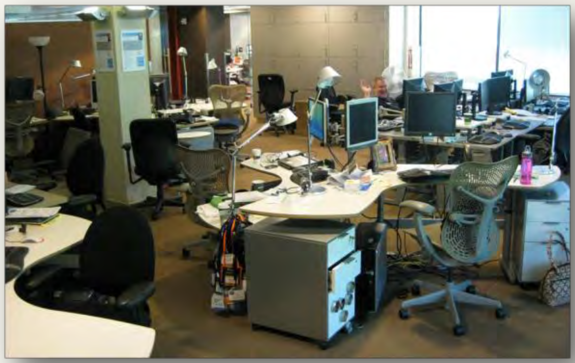
- Be proactive (circle of concern vs. circle of influence)
- Begin with the end in mind (decide what's important)
- Put first things first (important results vs. busy results)
- Think win-win (we all win or lose together)
- Seek first to understand, then to be understood (listen without being defensive)
- Synergize (we accomplish more together than all of us working individually)
- Sharpen the saw (renewal)

From the self-help book written by Stephen R. Covey (http://bit.ly/9Mjksg)



The Individual Space vs. Collaborative Dynamic

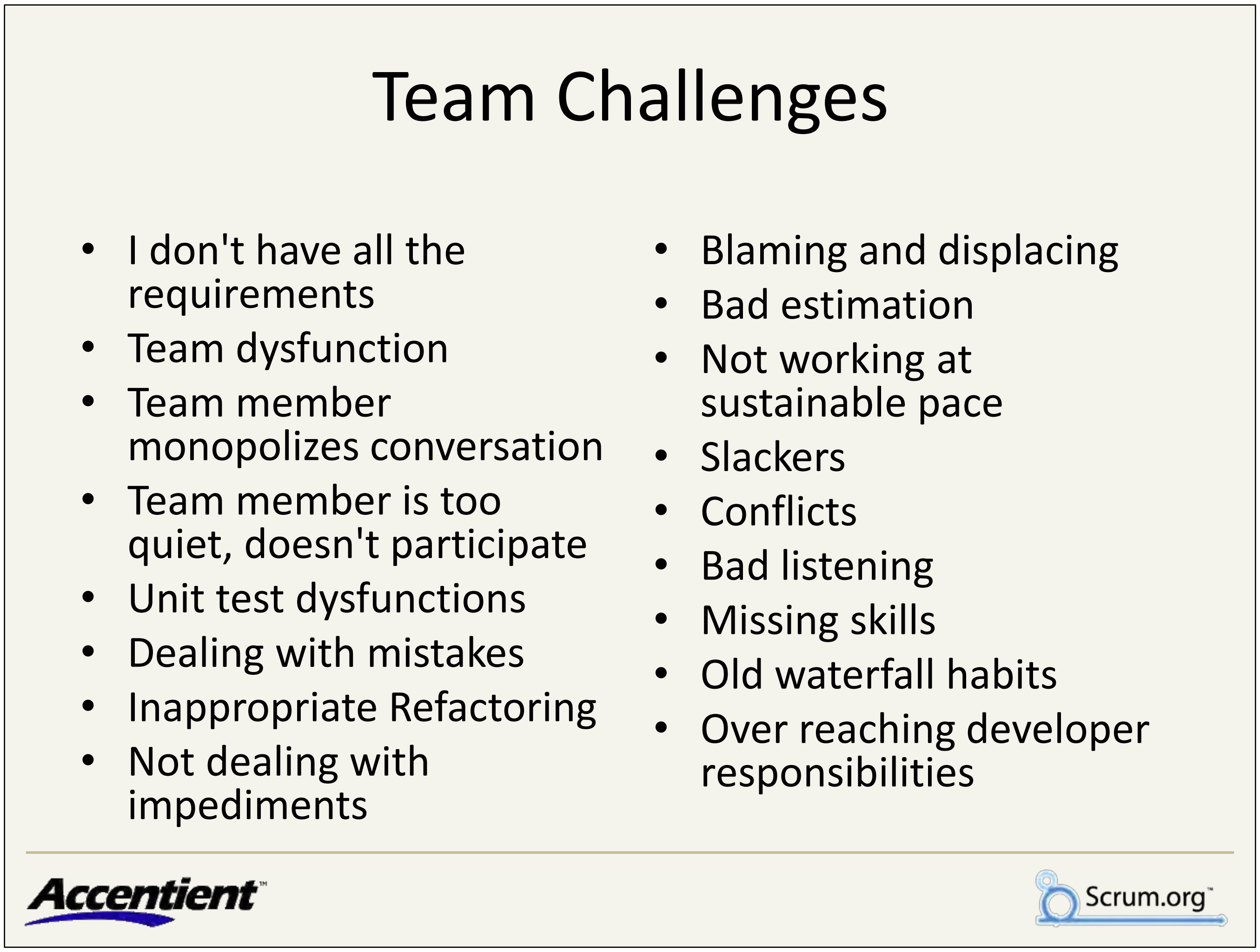
- You’ll likely be spending more time with your team
- Individual time and space is important too
  - Reserve for yourself
  - Respect other’s needs



Accentient™ Scrum.org™

Individual space vs. collaborative dynamic

Scrum enables the creation of self-organizing teams by encouraging co-location of all team members and verbal communication across all team members and disciplines that are involved in the project. This does not mean that all team members should sit in a crowded room talking over each other. Nor does it mean that the team members should disappear to cubicles after the daily Scrum and discontinue collaboration. There needs to be a balance so that ideas and concerns are exchanged as well as focused work accomplished.



Team Challenges

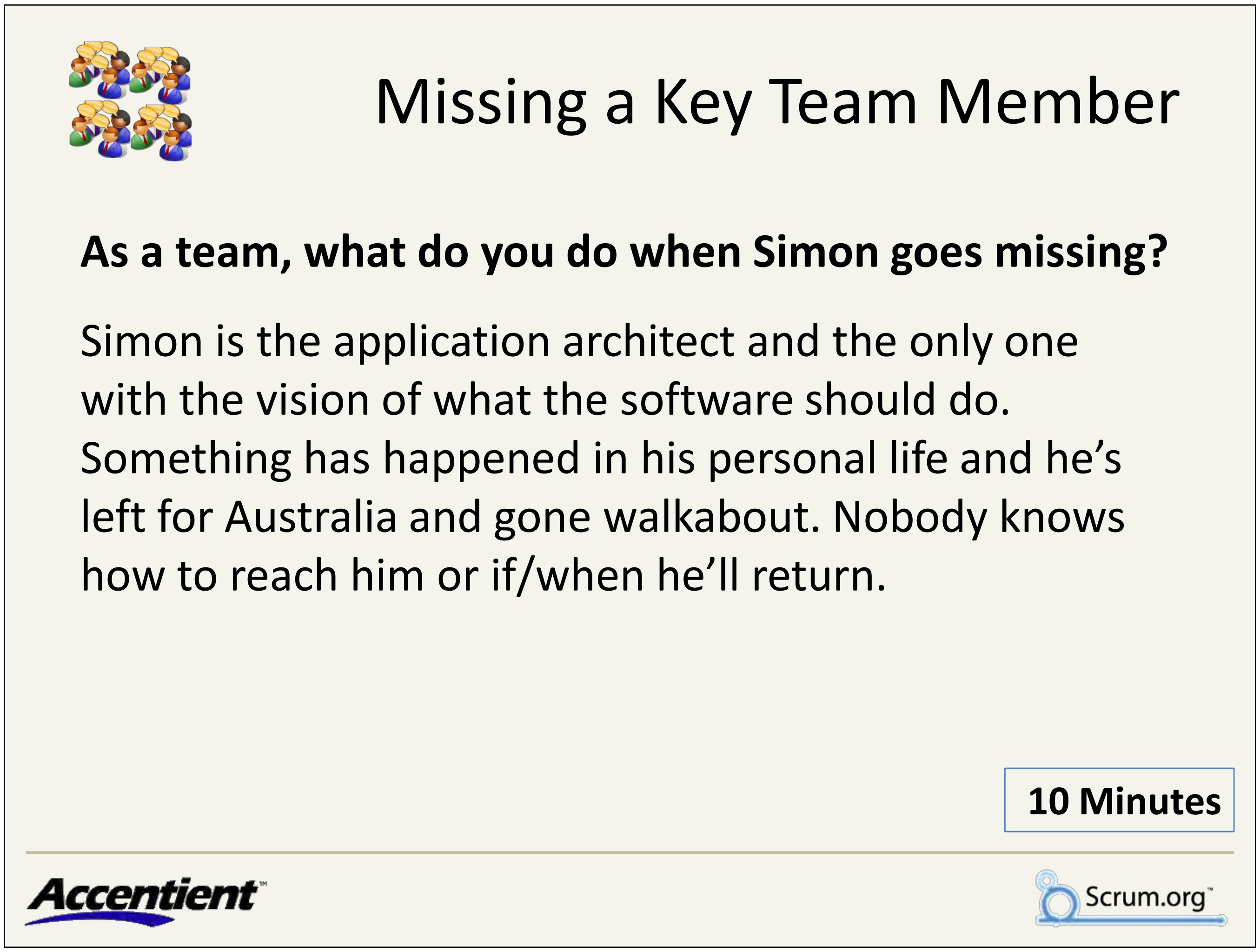
- I don't have all the requirements
- Team dysfunction
- Team member monopolizes conversation
- Team member is too quiet, doesn't participate
- Unit test dysfunctions
- Dealing with mistakes
- Inappropriate Refactoring
- Not dealing with impediments
- Blaming and displacing
- Bad estimation
- Not working at sustainable pace
- Slackers
- Conflicts
- Bad listening
- Missing skills
- Old waterfall habits
- Over reaching developer responsibilities

Accentient™ Scrum.org™

Team challenges

In 1965, Bruce Tuckman identified four necessary phases of group (team) development: forming, storming, norming, and performing. Each of these phases are necessary and inevitable in order for the team to grow, to face up to challenges, to tackle problems, to find solutions, to plan work, and to deliver results. This model has become the basis for subsequent models.

1. Forming – in the first stages of team building, the forming of the team takes place
2. Storming – every group will then enter the storming stage in which different ideas compete for consideration
3. Norming – the team manages to have one goal and come to a mutual plan for the team at this stage
4. Performing – certain teams are able to function as a unit as they find ways to get the job done smoothly and effectively without inappropriate conflict or the need for external supervision



Missing a Key Team Member

As a team, what do you do when Simon goes missing?

Simon is the application architect and the only one with the vision of what the software should do. Something has happened in his personal life and he's left for Australia and gone walkabout. Nobody knows how to reach him or if/when he'll return.

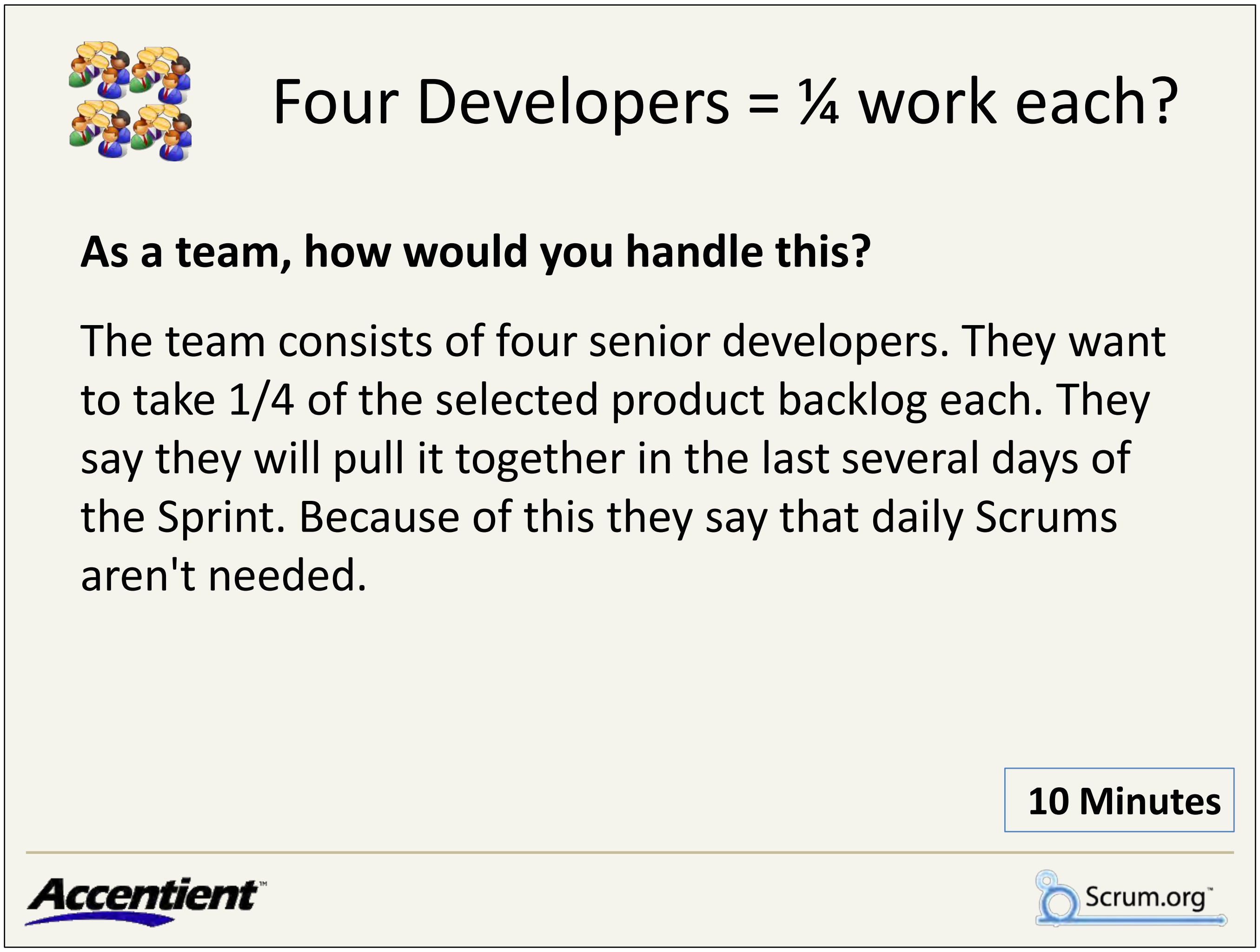
10 Minutes

Accentient™ Scrum.org™

Activity: what would you do when you are missing a key team member?

Questions to consider:

1. Why dysfunctions exist?
2. Is this bad Scrum?
3. If so, how is this bad Scrum?
4. How should the team handle the situation?
5. Does your team have a Simon? Is it you?



Four Developers = ¼ work each?

As a team, how would you handle this?

The team consists of four senior developers. They want to take 1/4 of the selected product backlog each. They say they will pull it together in the last several days of the Sprint. Because of this they say that daily Scrums aren't needed.

10 Minutes

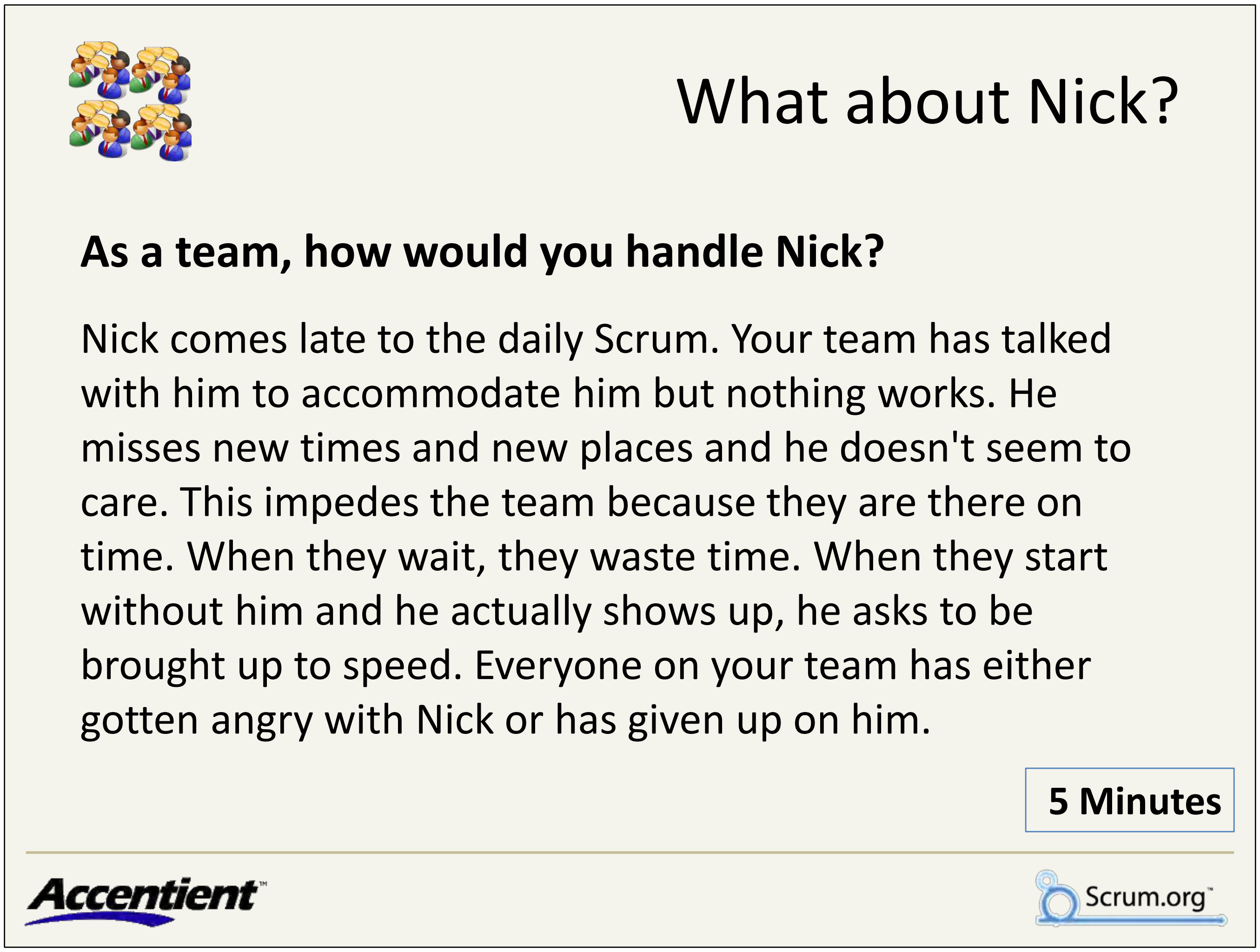
Accentient™

Scrum.org™

Activity: what would you do about these four senior developers?

Questions to consider:

1. Why dysfunctions exist?
2. Is this bad Scrum?
3. If so, how is this bad Scrum?
4. How should the team handle the situation?
5. If they used Team Foundation Server would this be okay?
6. Is it okay for one developer to handle all facets of implementing a story?
7. Do you believe that they can deliver business value?



What about Nick?

As a team, how would you handle Nick?

Nick comes late to the daily Scrum. Your team has talked with him to accommodate him but nothing works. He misses new times and new places and he doesn't seem to care. This impedes the team because they are there on time. When they wait, they waste time. When they start without him and he actually shows up, he asks to be brought up to speed. Everyone on your team has either gotten angry with Nick or has given up on him.

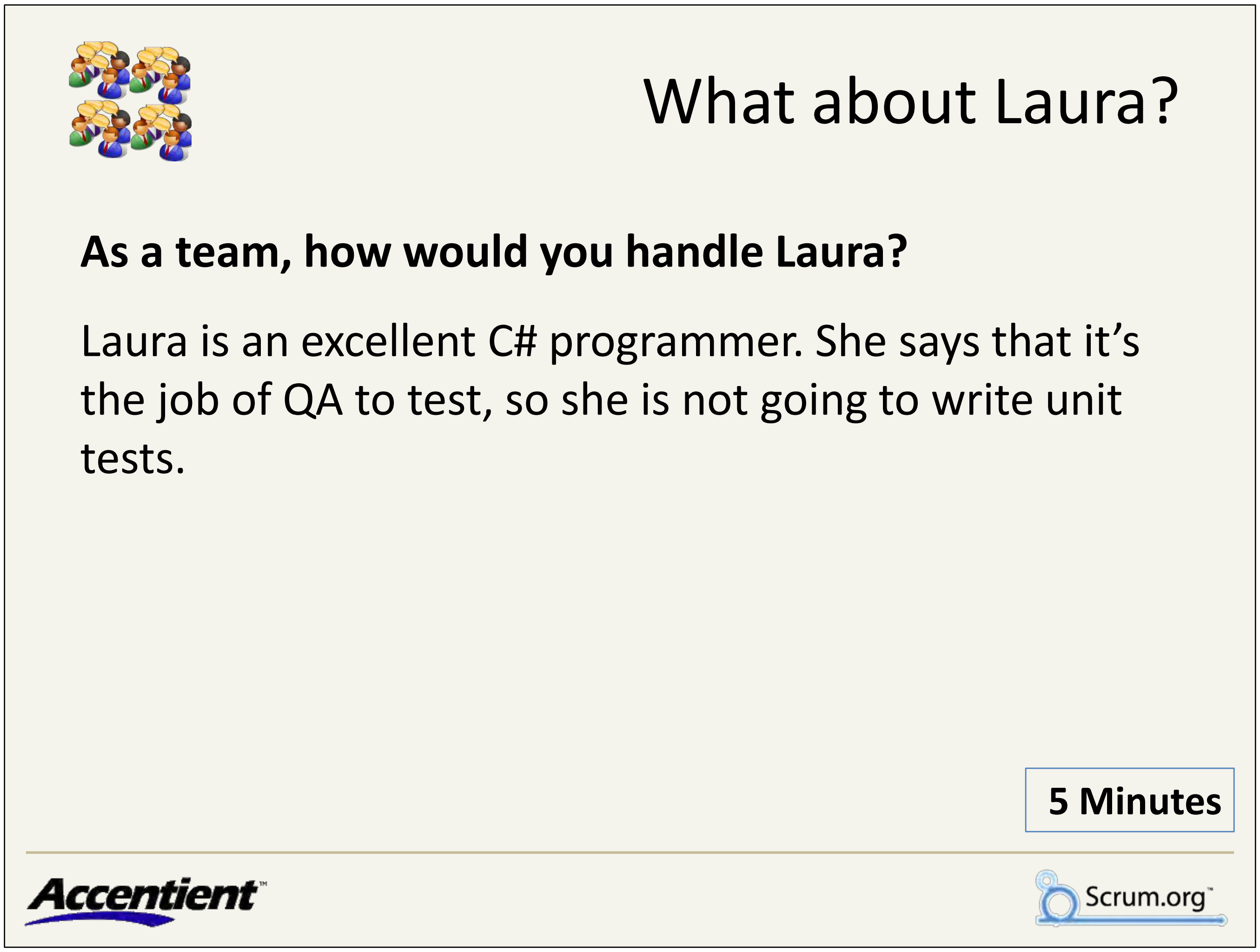
5 Minutes

Accentient™ Scrum.org™

Activity: what would you do about Nick?

Questions to consider:

1. Why dysfunctions exist?
2. Is this bad Scrum?
3. If so, how is this bad Scrum?
4. How should the team handle the situation?
5. What if nick’s shift is midnight to 8am?
6. What if Nick’s capacity (commitment to this team) is only 2 hrs/day?



What about Laura?

As a team, how would you handle Laura?

Laura is an excellent C# programmer. She says that it's the job of QA to test, so she is not going to write unit tests.

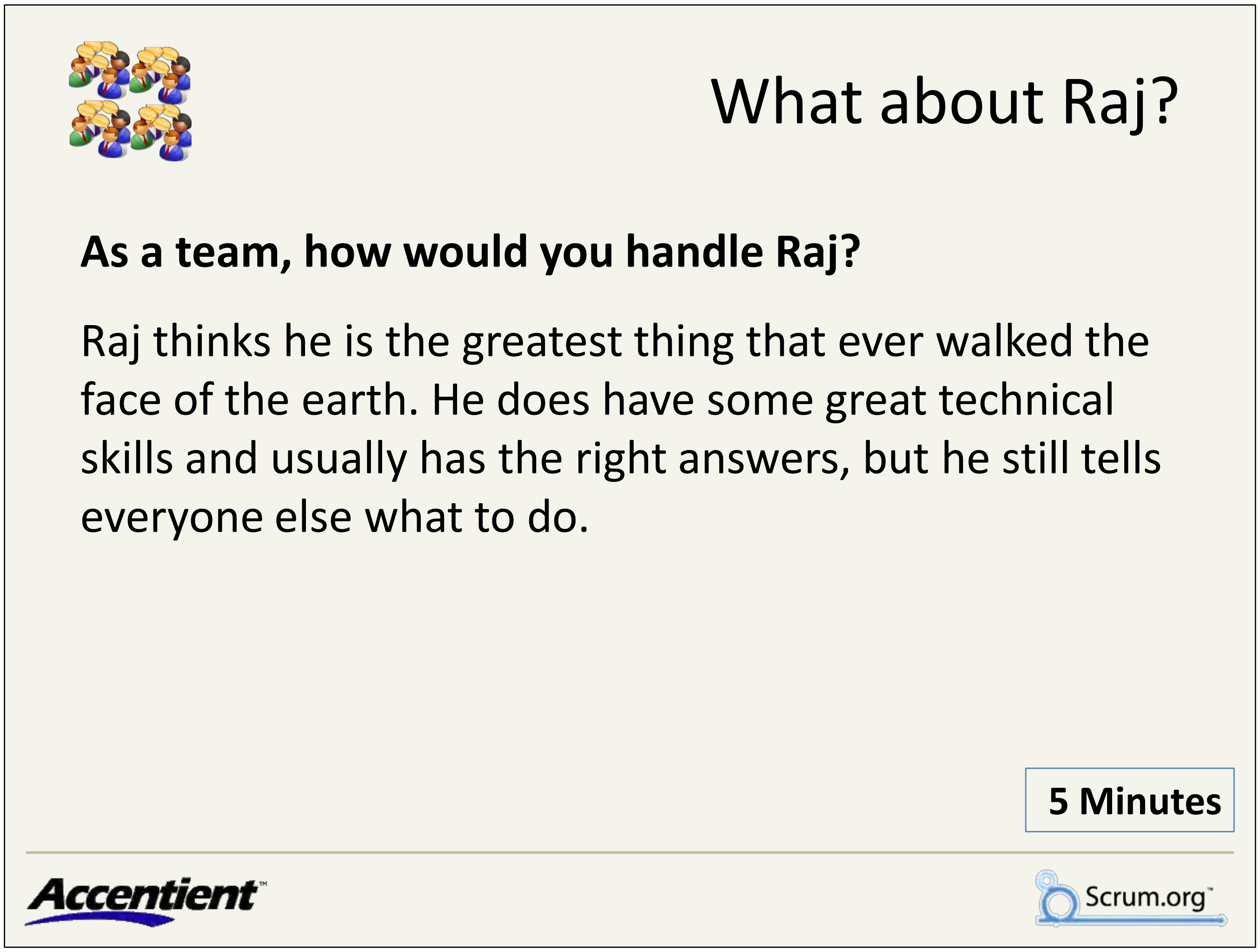
5 Minutes

Accentient™ Scrum.org™

Activity: what would you do about Laura?

Questions to consider:

1. Why dysfunctions exist?
2. Is this bad Scrum?
3. If so, how is this bad Scrum?
4. How should the team handle the situation?
5. Could the product suffer ? if so, how?
6. What if Laura were a DBA writing T-SQL? Should she still be expected to write unit tests?



What about Raj?

As a team, how would you handle Raj?

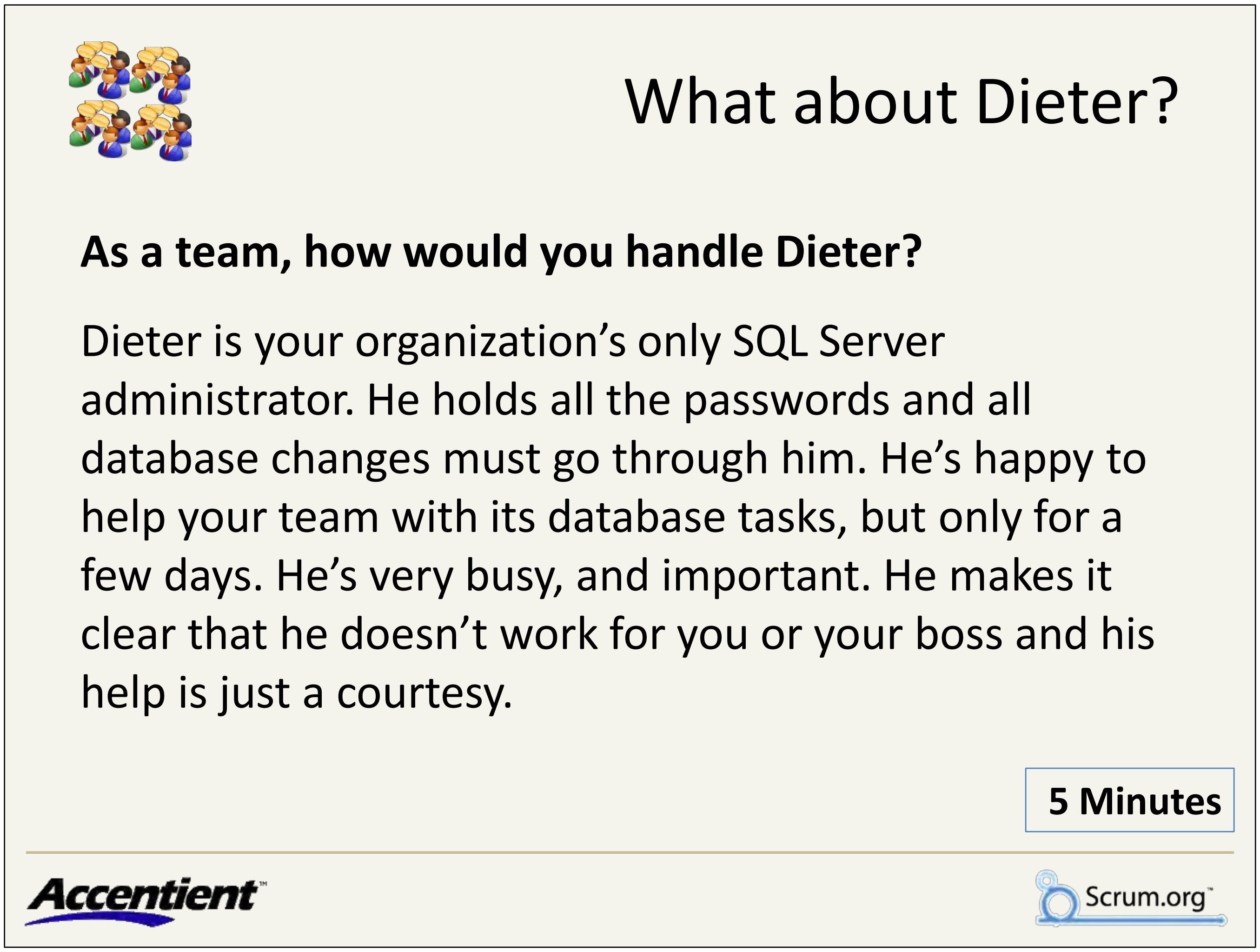
Raj thinks he is the greatest thing that ever walked the face of the earth. He does have some great technical skills and usually has the right answers, but he still tells everyone else what to do.

5 Minutes

Activity: what would you do about Raj?

Questions to consider:

1. Why dysfunctions exist?
2. Is this bad Scrum?
3. If so, how is this bad Scrum?
4. How should the team handle the situation?
5. What if this situation is working for the team?
6. What if Raj becomes a Simon (and goes walkabout)?



What about Dieter?

As a team, how would you handle Dieter?

Dieter is your organization's only SQL Server administrator. He holds all the passwords and all database changes must go through him. He's happy to help your team with its database tasks, but only for a few days. He's very busy, and important. He makes it clear that he doesn't work for you or your boss and his help is just a courtesy.

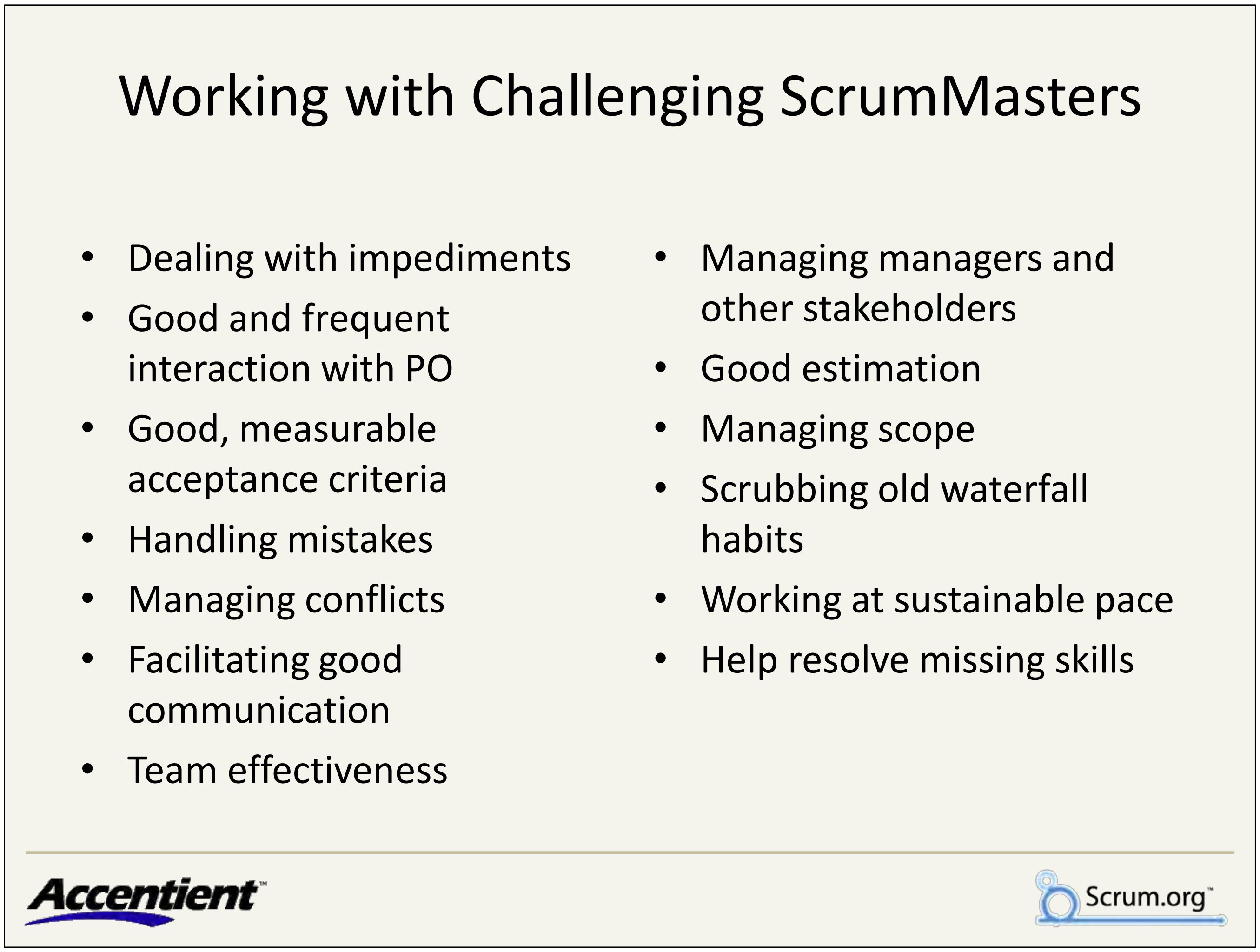
5 Minutes

Accentient™ Scrum.org™

Activity: what would you do about Dieter?

Questions to consider:

1. Why dysfunctions exist?
2. Is this bad Scrum?
3. If so, how is this bad Scrum?
4. How should the team handle the situation?



Working with Challenging ScrumMasters

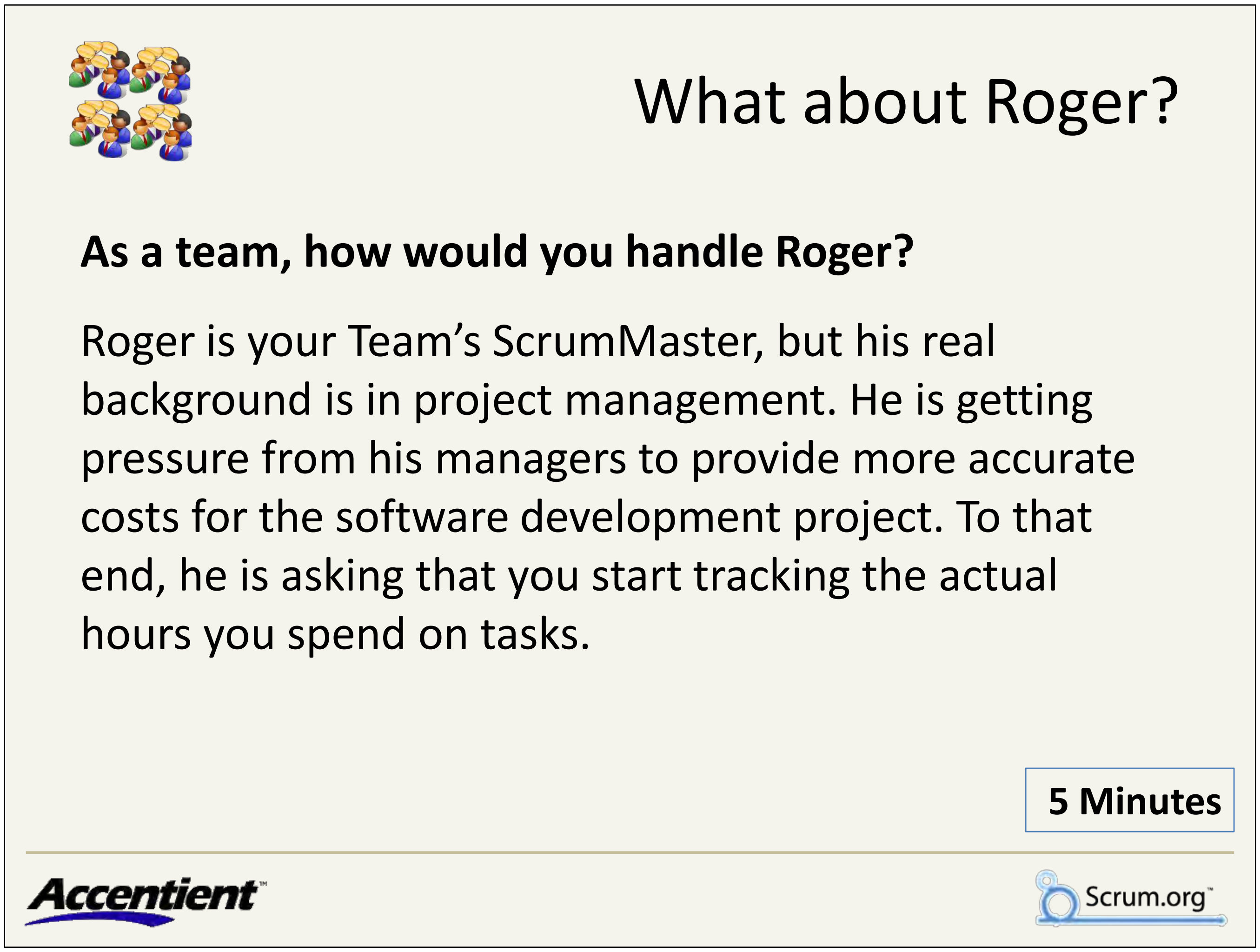
- Dealing with impediments
- Good and frequent interaction with PO
- Good, measurable acceptance criteria
- Handling mistakes
- Managing conflicts
- Facilitating good communication
- Team effectiveness
- Managing managers and other stakeholders
- Good estimation
- Managing scope
- Scrubbing old waterfall habits
- Working at sustainable pace
- Help resolve missing skills

Accentient™ Scrum.org™

Working with challenging ScrumMasters

This used to be the project manager, now it's the person who's responsible for ensuring that the process is used as intended. There are different ways of thinking of the ScrumMaster role. One might be as a parent, because when you get a team together, at first they don't know how to self-manage, they don't know how to work cross-functionally, they don't know how to work with a Product Owner, or how to work within timeboxes. So just like a parent, the ScrumMaster is responsible for teaching them how to do this until they know how, moving them from a relatively immature childlike state, to a mature self-functioning team. Similarly a ScrumMaster is like a coach, responsible for cheering them on, for being their leader, being their guide. The ScrumMaster is also like a referee ensuring that they follow the rules.

The ScrumMaster above and beyond anything has to enforce the rules. Many of the processes that are used in organizations create impediments to the team's progress. It is often easier to simply give in and accept that the organizational processes can't be changed and compromise the efficiency of Scrum and the team's output. So it is vital in order to keep the high productivity of Scrum that the rules are followed precisely and kept running regardless. By doing that, everything in the organization that is dysfunctional and gets in the way of building software regularly becomes obvious. It is the job of the ScrumMaster to remove any impediment, within or external to the team that prevents them reaching their goal of building the software they commit to at the beginning of each Sprint.



What about Roger?

As a team, how would you handle Roger?

Roger is your Team's ScrumMaster, but his real background is in project management. He is getting pressure from his managers to provide more accurate costs for the software development project. To that end, he is asking that you start tracking the actual hours you spend on tasks.

5 Minutes

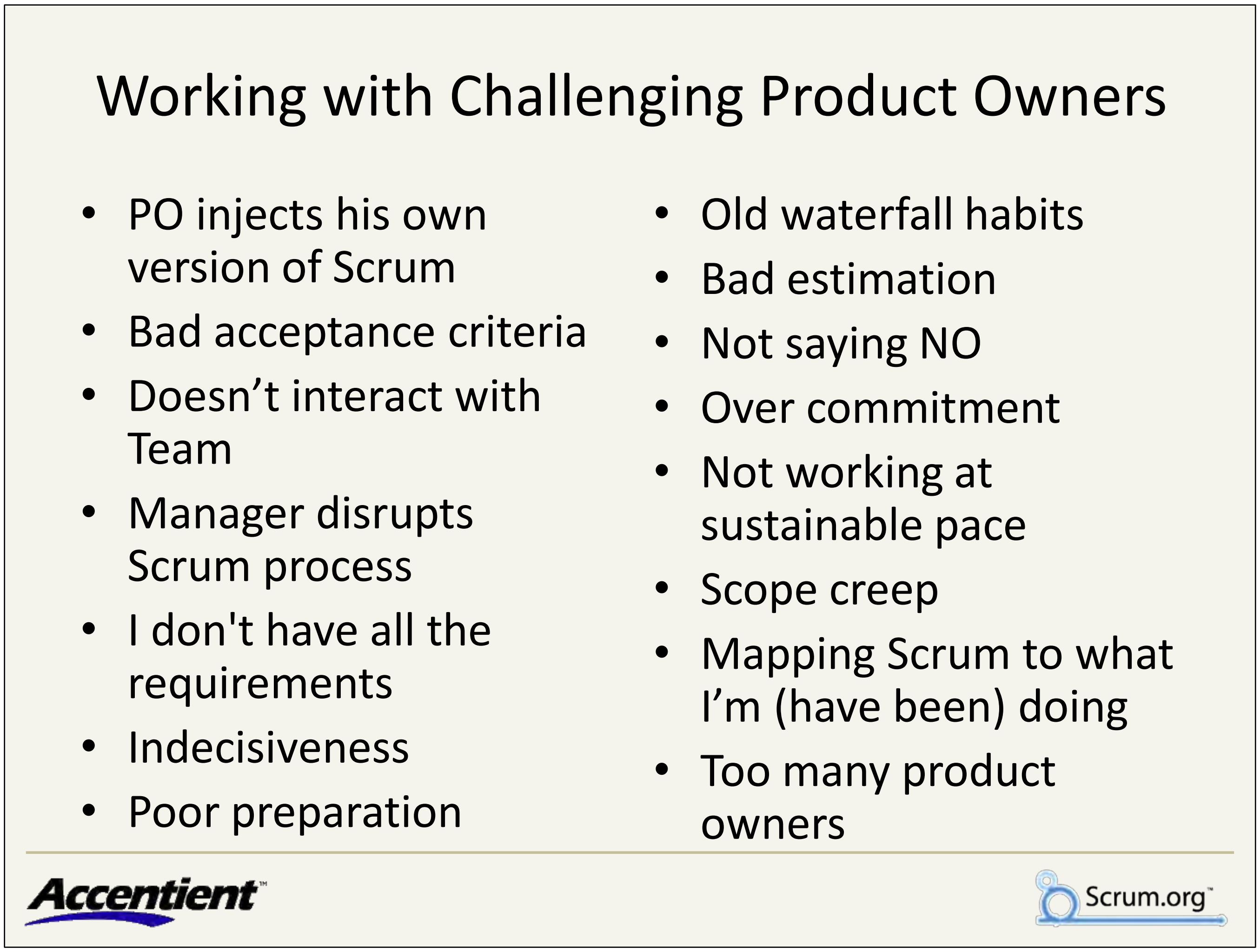
Accentient™

Scrum.org™

Activity: what would you do about Roger?

Questions to consider:

1. Why dysfunctions exist?
2. Is this bad Scrum?
3. If so, how is this bad Scrum?
4. How should the team handle the situation?

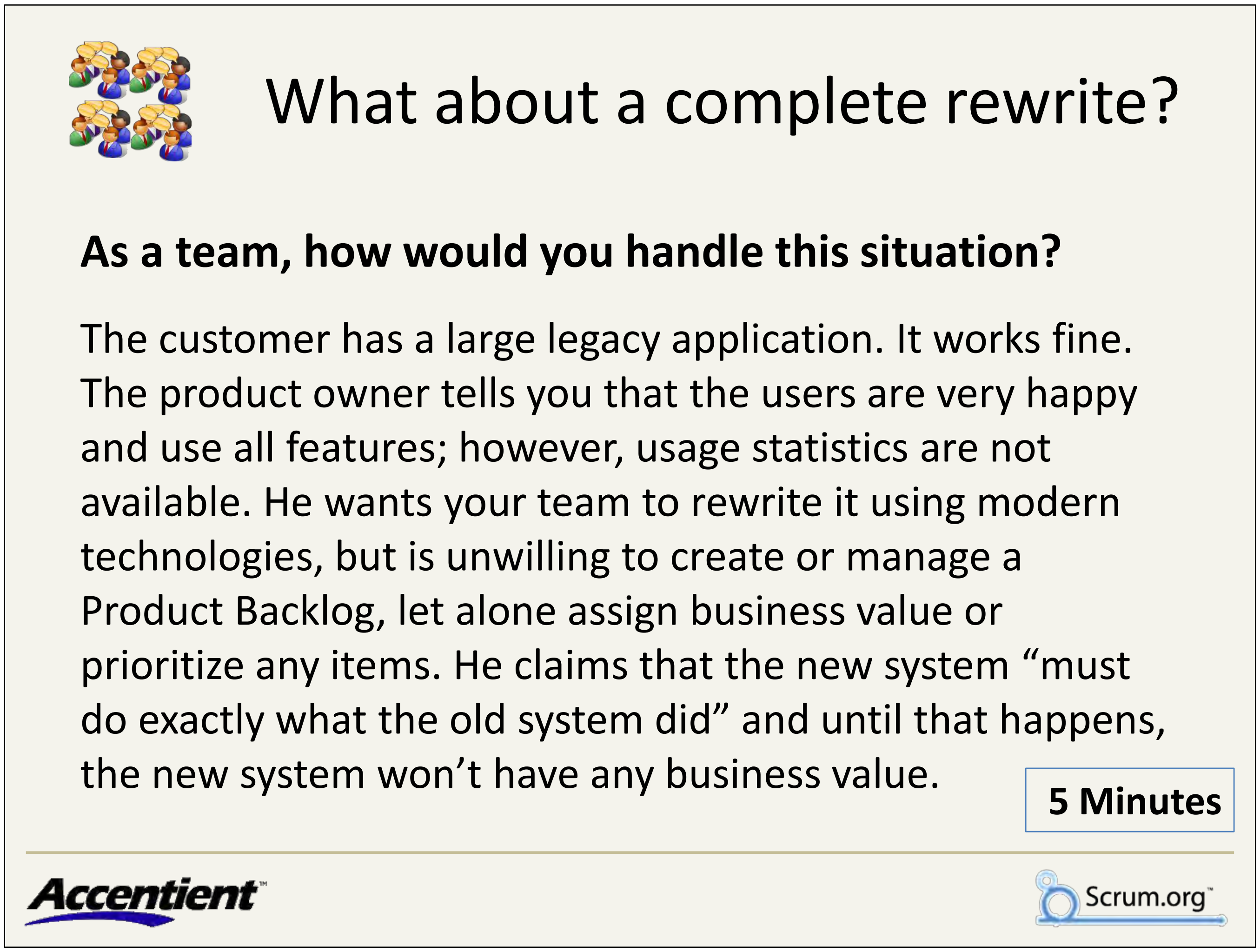


Working with Challenging Product Owners

- PO injects his own version of Scrum
- Bad acceptance criteria
- Doesn’t interact with Team
- Manager disrupts Scrum process
- I don't have all the requirements
- Indecisiveness
- Poor preparation
- Old waterfall habits
- Bad estimation
- Not saying NO
- Over commitment
- Not working at sustainable pace
- Scope creep
- Mapping Scrum to what I’m (have been) doing
- Too many product owners

Working with challenging Product Owners

In addition to all of these challenges, the primary one is if the Product Owner doesn't successfully reflect the real value/priority for the business. This is both the biggest responsibility and biggest potential risk of the Product Owner.



What about a complete rewrite?

As a team, how would you handle this situation?

The customer has a large legacy application. It works fine. The product owner tells you that the users are very happy and use all features; however, usage statistics are not available. He wants your team to rewrite it using modern technologies, but is unwilling to create or manage a Product Backlog, let alone assign business value or prioritize any items. He claims that the new system "must do exactly what the old system did" and until that happens, the new system won't have any business value.

5 Minutes

Accentient™ Scrum.org™

Activity: what about a complete rewrite?

Finally: training, assessment, and certification for the *team* ...

THE PROFESSIONAL SCRUM DEVELOPER PROGRAM

Professional Scrum Developer
a training and certification program by SCRUM.ORG™
Accentient™
Scrum.org™

Professional Scrum Developer program

The Professional Scrum Developer program trains, assesses, and certifies Scrum developers working on a specific technology platform. The program includes a five-day course, an online assessment, and an industry-recognized certification. Each Professional Scrum Developer program targets a specific technology stack. At present, Scrum.org offers programs for Microsoft .NET and for Java.

The Professional Scrum Developer Program is a unique offering for a historically neglected role. The program teaches Scrum directly to the people doing the work. It extends the foundation of Scrum by interleaving industry best practices for software development. The classroom environment simulates real enterprise ALM, leading student teams from “requirements” to “release”. The SD program addresses real-world challenges faced by development teams. It incorporates “soft-skills” typically avoided in technical training, but which have become critical to the success of development teams today.

For more information watch this video of Ken Schwaber and Sam Guckenheimer discussing the Professional Scrum Developer program: http://bit.ly/fNYijc.

SCRUM

February 2010

Scrum: Developed and sustained by Ken Schwaber and Jeff Sutherland

Scrum.org™

Acknowledgements

General

Scrum is based on industry-accepted best practices, used and proven for decades. It is then set in an empirical process theory. As Jim Coplien once remarked to Jeff, "Everyone will like Scrum; it is what we already do when our back is against the wall."

People

Of the thousands of people that have contributed to Scrum, we should single out those that were instrumental in its first ten years. First there were Jeff Sutherland, working with Jeff McKenna, and Ken Schwaber with Mike Smith and Chris Martin. Scrum was first formally presented and published at OOPSLA 1995. During the next five years, Mike Beadle and Martine Devos made significant contributions. And then everyone else, without whose help Scrum wouldn't have been refined into what it is today.

History

The history of Scrum can already be considered long in the world of software development. To honor the first places where it was tried and refined, we honor Individual, Inc., Fidelity Investments, and IDX (now GE Medical).

Purpose

Scrum has been used to develop complex products since the early 1990s. This paper describes how to use Scrum to build products. Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. The role of Scrum is to surface the relative efficacy of your development practices so that you can improve upon them **while providing a framework within which complex products can be developed.**

Scrum Theory

Scrum, which is grounded in empirical process control theory, employs an iterative, incremental approach to optimize predictability and control risk. Three pillars uphold every implementation of empirical process control.

The first leg is transparency

Transparency ensures that aspects of the process that affect the outcome must be visible to those managing the outcomes. Not only must these aspects be transparent, but also what is being seen must be known. That is, when someone inspecting a process believes that something is done; it must be equivalent to their definition of done.

The second leg is inspection

The various aspects of the process must be inspected frequently enough so that unacceptable variances in the process can be detected. The frequency of inspection has to take into consideration that all processes are changed by the act of inspection. A conundrum occurs when the required frequency of inspection exceeds the tolerance to inspection of the process. Fortunately, this doesn't seem to be true of software development. The other factor is the skill and diligence of the people inspecting the work results.

The Third Leg is Adaptation

If the inspector determines from the inspection that one or more aspects of the process are outside acceptable limits, and that the resulting product will be unacceptable, the inspector must adjust the process or the material being processed. The adjustment must be made as quickly as possible to minimize further deviation.

There are three points for inspection and adaptation in Scrum. The Daily Scrum meeting is used to inspect progress toward the Sprint goal, and to make adaptations that optimize the value of the next work day. In addition, the Sprint Review and Planning meetings are used to inspect progress toward the Release Goal and to make adaptations that optimize the value of the next Sprint. Finally, the Sprint Retrospective is used to review the past Sprint and determine what adaptations will make the next Sprint more productive, fulfilling, and enjoyable.

Scrum Content

The Scrum framework consists of a set of **Scrum Teams** and their associated roles; **Time-Boxes**, **Artifacts,** and **Rules**.

Scrum Teams are designed to optimize flexibility and productivity; to this end, they are self-organizing, they are cross-functional, and they work in iterations. Each Scrum Team has three roles: 1) the **ScrumMaster**, who is responsible for ensuring the process is understood and followed; 2) the **Product Owner**, who is responsible for maximizing the value of the work that the Scrum Team does; and 3) the **Team**, which does the work. The Team consists of developers with all the skills to turn the Product Owner's requirements into a potentially releasable piece of the product by the end of the Sprint.

Scrum employs time boxes to create regularity. Elements of Scrum that are time-boxed include the **Release Planning Meeting**, the **Sprint Planning Meeting**, the **Sprint**, the **Daily Scrum**, the **Sprint Review**, and the **Sprint Retrospective**. The heart of Scrum is a **Sprint**, which is an iteration of one month or less that is of consistent length throughout a development effort. All Sprints use the same Scrum framework, and all Sprints deliver an increment of the final product that is potentially releasable. One Sprint starts immediately after the other.

Scrum employs four principal artifacts. The **Product Backlog** is a prioritized list of everything that might be needed in the product. The **Sprint Backlog** is a list of tasks to turn the Product Backlog for one Sprint into an increment of potentially shippable product. A burndown is a measure of remaining backlog over time. A **Release Burndown** measures remaining Product Backlog across the time of a release plan. A **Sprint Burndown** measures remaining **Sprint Backlog** items across the time of a Sprint.

Rules bind together Scrum's time-boxes, roles, and artifacts. Its rules are described throughout the body of this document. For example, it is a Scrum rule that only Team members - the people committed to turning the Product Backlog into an increment – can talk during a Daily Scrum. Ways of implementing Scrum that are not rules but rather are suggestions are described in "Tips" boxes.

TIP

When rules are not stated, the users of Scrum are expected to figure out what to do. Don't try to figure out a perfect solution, because the problem usually changes quickly. Instead, try something and see how it works. The inspect-and-adapt mechanisms of Scrum's empirical nature will guide you.

SCRUM ROLES

The Scrum Team consists of the ScrumMaster, the Product Owner, and the Team. Scrum Team members are called "pigs." The Product Owner is the "pig" of the Product Backlog. The Team is the "pig" of the Sprint work. The ScrumMaster is the "pig" of the Scrum process. Everyone else is a "chicken." Chickens cannot tell "pigs" how to do their work. Chickens and pigs come from the story, *"A chicken and a pig are together when the chicken says, "Let's start a restaurant!"*

The pig thinks it over and says, "What would we call this restaurant?"

The chicken says, "Ham n' Eggs!"

The pig says, "No thanks, I'd be committed, but you'd only be involved!"

THE SCRUMMASTER

The ScrumMaster is responsible for ensuring that the Scrum Team adheres to Scrum values, practices, and rules. The ScrumMaster helps the Scrum Team and the organization adopt Scrum. The ScrumMaster teaches the Scrum Team by coaching and by leading it to be more productive and produce higher quality products. The ScrumMaster helps the Scrum Team understand and use self-organization and cross-functionality. The ScrumMaster also helps the Scrum Team do its best in an organizational environment that may not yet be optimized for complex product development. When the ScrumMaster helps make these changes, this is called "removing impediments." The ScrumMaster's role is one of a servant-leader for the Scrum Team.

TIP

The ScrumMaster works with the customers and management to identify and instantiate a Product Owner. The ScrumMaster teaches the Product Owner how to do his or her job. Product Owners are expected to know how to manage to optimize value using Scrum. If they don't, we hold the ScrumMaster accountable.

TIP

The ScrumMaster may be a member of the Team; for example, a developer performing Sprint tasks. However, this often leads to conflicts when the ScrumMaster has to choose between removing impediments and performing tasks. The ScrumMaster should never be the Product Owner.

THE PRODUCT OWNER

The Product Owner is the one and only person responsible for managing the Product Backlog and ensuring the value of the work the Team performs. This person maintains the Product Backlog and ensures that it is visible to everyone. Everyone knows what items have the highest priority, so everyone knows what will be worked on.

TIP

For commercial development, the Product Owner may be the product manager. For in-house development efforts, the Product Owner could be the manager of the business function that is being automated.

The Product Owner is one person, not a committee. Committees may exist that advise or influence this person, but people who want to change an item's priority have to convince the Product Owner. Companies that adopt Scrum may find it influences their methods for setting priorities and requirements over time.

TIP

The Product Owner can be a Team member, also doing development work. This additional responsibility may cut into the Product Owner's ability to work with stakeholders. However, the Product Owner can never be the ScrumMaster.

For the Product Owner to succeed, everyone in the organization has to respect his or her decisions. No one is allowed to tell the Team to work from a different set of priorities, and Teams aren't allowed to listen to anyone who says otherwise. The Product Owner's decisions are visible in the content and prioritization of the Product Backlog. This visibility requires the Product Owner to do his or her best, and it makes the role of Product Owner both a demanding and a rewarding one.

The Team

Teams of developers turn Product Backlog into increments of potentially shippable functionality every Sprint. Teams are also cross-functional; Team members must have all of the skills necessary to create an increment of work. Team members often have specialized skills, such as programming, quality control, business analysis, architecture, user interface design, or data base design. However, the skills that Team member share – that is, the skill of addressing a requirement and turning it into a usable product – tend to be more important than the ones that they do not. People who refuse to code because they are architects or designers are not good fits for Teams. Everyone chips in, even if that requires learning new skills or remembering old ones. There are no titles on Teams, and there are no exceptions to this rule. Teams do not contain sub-Teams dedicated to particular domains like testing or business analysis, either.

Teams are also self-organizing. No one – not even the ScrumMaster - tells the Team how to turn Product Backlog into increments of shippable functionality. The Team figures this out on its own. Each Team member applies his or her expertise to all of the problems. The synergy that results improves the entire Team's overall efficiency and effectiveness.

The optimal size for a Team is seven people, plus or minus two. When there are fewer than five Team members, there is less interaction and as a result less productivity gain. What's more, the Team may encounter skill constraints during parts of the Sprint and be unable to deliver a releasable piece of the product. If there are more than nine members, there is simply too much coordination required. Large Teams generate too much complexity for an empirical process to manage. However, we have encountered some successful Teams that have exceeded the upper and lower bounds of this size range. The Product Owner and ScrumMaster roles are not included in this count unless they are also pigs, working on tasks in the Sprint Backlog.

Team composition may change at the end of a Sprint. Every time Team membership is changed, the productivity gained from self-organization is diminished. Care should be taken when changing Team composition.

Time-Boxes

The Time-Boxes in Scrum are the **Release Planning Meeting**, the **Sprint**, the **Sprint Planning Meeting**, the **Sprint Review**, the **Sprint Retrospective**, and the **Daily Scrum**.

Release Planning Meeting

The purpose of release planning is to establish a plan and goals that the Scrum Teams and the rest of the organizations can understand and communicate. Release planning answers the questions, "How can we turn the vision into a winning product in best possible way? How can we meet or exceed the desired customer satisfaction and Return on Investment?" The release plan establishes the goal of the release, the highest priority Product Backlog, the major risks, and the overall features and functionality that the release will contain. It also establishes a probable delivery date and cost that should hold if nothing changes. The organization can then inspect progress and make changes to this release plan on a Sprint-by-Sprint basis.

Release planning is entirely optional. If Scrum teams start work without the meeting, the absence of its artifacts will become apparent as an impediment that needs to be resolved. Work to resolve the impediment will become an item in the Product Backlog.

Products are built iteratively using Scrum, wherein each Sprint creates an increment of the product, starting with the most valuable and riskiest. More and more Sprints create additional increments of the product. Each increment is a potentially shippable slice of the entire product. When enough increments have been created for the Product to be of value, of use to its investors, the product is released.

Most organizations already have a release planning process, and in most of these processes most of the planning is done at the beginning of the release and left unchanged as time passes. In Scrum release planning, an overall goal and probable outcomes are defined. This release planning usually requires no more than 15-20% of the time an organization consumed to build a traditional release plan. However, a Scrum release performs just-in-time planning every Sprint Review and Sprint Planning meeting, as well as daily just-in-time planning at every Daily Scrum meeting. Overall, Scrum release efforts probably consume slightly more effort than tradition release planning efforts.

Release planning requires estimating and prioritizing the Product Backlog for the Release. There are many techniques for doing so that lie outside the purview of Scrum but are nonetheless useful when used with it.

The Sprint

A Sprint is an iteration. Sprints are time-boxed. During the Sprint, the ScrumMaster ensures that no changes are made that would affect the Sprint Goal. Both Team composition and quality goals remain constant throughout the Sprint. Sprints contain and consist of the Sprint Planning meeting, the development work, the Sprint Review, and the Sprint Retrospective. Sprints occur one after another, with no time in between Sprints.

Tip

If the Team senses that it has overcommitted, it meets with the Product Owner to remove or reduce the scope of Product Backlog selected for the Sprint. If the Team senses that it may have extra time, it can work with the Product Owner to select additional Product Backlog.

A project is used to accomplish something; in software development, it is used to build a product or system. Every project consists of a definition of what is to be built, a plan to build it, the work done according to the plan, and the resultant product. Every project has a horizon, which is to say the time frame for which the plan is good. If the horizon is too long, the definition may have changed, too many variables may have entered in, the risk may be too great, etc. Scum is a framework for a project whose horizon is no more than one month long, where there is enough complexity that a longer horizon is too risky. The predictability of the project has to be controlled at least each month, and the risk that the project may go out of control or become unpredictable is contained at least each month.

Tip

When a Team begins Scrum, two-week Sprints allow it to learn without wallowing in uncertainty. Sprints of this length can be synchronized with other Teams by adding two increments together.

Sprints can be cancelled before the Sprint time box is over. Only the Product Owner has the authority to cancel the Sprint, although he or she may do so under influence from the stakeholders, the Team, or the ScrumMaster. Under what kind of circumstances might a Sprint need to be cancelled? Management may need to cancel a Sprint if the Sprint Goal becomes obsolete. This could occur if the company changes direction or if market or technology conditions change. In general, a Sprint should be cancelled if it no longer makes sense given the circumstances. However, because of the short duration of Sprints, it rarely makes sense to do so.

When a Sprint is cancelled, any completed and "done" Product Backlog items are reviewed. They are accepted if they represent a potentially shippable increment. All other Product Backlog items are put back on the Product Backlog with their initial estimates. Any work done on them is assumed to be lost. Sprint terminations consume resources, since everyone has to regroup in another Sprint planning meeting to start another Sprint. Sprint terminations are often traumatic to the Team, and they are very uncommon.

Sprint Planning Meeting

The Sprint Planning meeting is when the iteration is planned. It is time-boxed to eight hours for a one month Sprint. For shorter Sprints, allocate proportionately less of the total Sprint length to this meeting (for example, two weeks would be a four-hour Sprint Planning Meeting). The Sprint Planning Meeting consists of two parts. The first part is when what will be done in the Sprint is decided upon. The second part (a four-hour time-box for a monthly Sprint) is when the Team figures out how it is going to build this functionality into a product increment during the Sprint.

There are two parts to the Sprint Planning Meeting: the "What?" part and the "How?" part. Some Scrum Teams combine the two. In the first part, the Scrum Team addresses the question of "What?" Here, the Product Owner presents the top priority Product Backlog to the Team. They work together to figure out what functionality is to be developed during the next Sprint. The input to this meeting is the Product Backlog, the latest increment of product, the capacity of the Team, and past performance of the Team. The amount of backlog the Team selects is solely up to the Team. Only the Team can assess what it can accomplish over the upcoming Sprint.

Having selected the Product Backlog, a Sprint Goal is crafted. The Sprint Goal is an objective that will be met through the implementation of the Product Backlog. This is a statement that provides guidance to the Team on why it is building the increment. The Sprint Goal is a subset of the release goal.

The reason for having a Sprint Goal is to give the Team some wiggle room regarding the functionality. For example, the goal for the above Sprint could also be: "Automate the client account modification functionality through a secure, recoverable transaction middleware capability." As the Team works, it keeps this goal in mind. In order to satisfy the goal, it implements the functionality and technology. If the work turns out to be harder than the Team had expected, then the Team collaborates with the Product Owner and only partially implement the functionality.

In the second part of the Sprint Planning Meeting, the Team addresses the question of "How?" During the second part of the Sprint Planning Meeting (four hour time-box for a monthly Sprint), the Team figures out how it will turn the Product Backlog selected during Sprint Planning Meeting (What) into a done increment. The Team usually starts by designing the work. While designing, the Team identifies tasks. These tasks are the detailed pieces of work needed to convert the Product Backlog into working software. Tasks should have decomposed so they can be done in less than one day. This task list is called the Sprint Backlog. The Team self-organizes to undertake the work in the Sprint Backlog, either during the Sprint Planning meeting or just-in-time during the Sprint.

The Product Owner is present during the second part of the Sprint Planning Meeting to clarify the Product Backlog and to help make trade-offs. If the Team determines that it has too much or too little work, it may renegotiate the Product Backlog with the Product Owner. The Team may also invite other people to attend in order to provide technical or domain advice. A new Team often first realizes that it will either sink or swim as a Team, not individually, in this meeting. The Team realizes that it must rely on itself. As it realizes this, it starts to self-organize to take on the characteristics and behavior of a real Team.

TIP

Usually, only 60-70% of the total Sprint Backlog will be devised in the Sprint Planning meeting. The rest is stubbed out for later detailing, or given large estimates that will be decomposed later in the Sprint.

SPRINT REVIEW

At the end of the Sprint, a Sprint Review meeting is held. This is a four hour time-boxed meeting for one month Sprints. For Sprints of lesser duration, allocate proportionately less of the total Sprint length to this meeting (for example, two weeks would be a two-hour Sprint Review). During the Sprint Review, the Scrum Team and stakeholders collaborate about what was just done. Based on that and changes to the Product Backlog during the Sprint, they collaborate about what are the next things that could be done. This is an informal meeting, with the presentation of the functionality intended to foster collaboration about what to do next.

The meeting includes at least the following elements. The Product Owner identifies what has been done and what hasn't been done. The Team discusses what went well during the Sprint and what problems it ran into, and how it solved these problems. The Team then demonstrates the work that is done and answers questions. The Product Owner then discusses the Product Backlog as it stands. He or she projects likely completion dates with various velocity assumptions. The entire group then collaborates about what it has seen and what this means regarding what to do next. The Sprint Review provides valuable input to subsequent Sprint Planning meeting.

Sprint Retrospective

After the Sprint Review and prior to the next Sprint Planning meeting, the Scrum Team has a Sprint Retrospective meeting. This is a three hour, time-boxed meeting for monthly Sprints (allocate proportionately less of the total Sprint length to this meeting). At this meeting, the ScrumMaster encourages the Scrum Team to revise, within the Scrum process framework and practices, its development process to make it more effective and enjoyable for the next Sprint. Many books document techniques that are helpful to use in Retrospectives.

The purpose of the Retrospective is to inspect how the last Sprint went in regards to people, relationships, process and tools. The inspection should identify and prioritize the major items that went well and those items that-if done differently-could make things even better. These include Scrum Team composition, meeting arrangements, tools, definition of "done," methods of communication, and processes for turning Product Backlog items into something "done." By the end of the Sprint Retrospective, the Scrum Team should have identified actionable improvement measures that it implements in the next

Sprint. These changes become the adaptation to the empirical inspection.

Daily Scrum

Each Team meets daily for a 15-minute inspect and adapt meeting called the Daily Scrum. The Daily Scrum is at the same time and same place throughout the Sprints. During the meeting, each Team member explains:

1. What he or she has accomplished since the last meeting;
2. What he or she is going to do before the next meeting; and
3. What obstacles are in his or her way.

Daily Scrums improve communications, eliminate other meetings, identify and remove impediments to development, highlight and promote quick decision-making, and improve everyone's level of project knowledge.

The ScrumMaster ensures that the Team has the meeting. The Team is responsible for conducting the Daily Scrum. The ScrumMaster teaches the Team to keep the Daily Scrum short by enforcing the rules and making sure that people speak briefly. The ScrumMaster also enforces the rule that chickens are not allowed to talk or in anyway interfere with the Daily Scrum.

The Daily Scrum is not a status meeting. It is not for anyone but the people transforming the Product Backlog items into an increment (the Team). The Team has committed to a Sprint Goal, and to these Product Backlog items. The Daily Scrum is an inspection of the progress toward that Sprint Goal (the three questions). Follow-on meetings usually occur to make adaptations to the upcoming work in the Sprint. The intent is to optimize the probability that the Team will meet its Goal. This is a key inspect and adapt meeting in the Scrum empirical process.

Scrum Artifacts

Scrum Artifacts include the Product Backlog, the Release Burndown, the Sprint Backlog, and the Sprint Burndown.

Product Backlog and Release Burndown

The requirements for the product that the Team(s) is developing are listed in the Product Backlog. The Product Owner is responsible for the Product Backlog, its contents, its availability, and its prioritization. Product Backlog is never complete. The initial cut at developing it only lays out the initially known and best-understood requirements. The Product Backlog evolves as the product and the environment in which it will be used evolves. The Backlog is dynamic in that it constantly changes to identify what the product needs to be appropriate, competitive, and useful. As long as a product exists, Product Backlog also exists.

The Product Backlog represents everything necessary to develop and launch a successful product. It is a list of all features, functions, technologies, enhancements, and bug fixes that constitute the changes that will be made to the product for future releases. Product Backlog items have the attributes of a description, priority, and estimate. Priority is driven by risk, value, and necessity. There are many techniques for assessing these attributes.

Tip

Product Backlog items are usually stated as User Stories. Use Cases are appropriate as well, but they are better for use in developing life- or mission-critical software.

Product Backlog is sorted in order of priority. Top priority Product Backlog drives immediate development activities. The higher the priority, the more urgent it is, the more it has been thought about, and the more consensus there is regarding its value. Higher priority backlog is clearer and has more detailed information than lower priority backlog. Better estimates are made based on the greater clarity and increased detail. The lower the priority, the less the detail, until you can barely make out the item.

As a product is used, as its value increases, and as the marketplace provides feedback, the product's backlog emerges into a larger and more exhaustive list. Requirements never stop changing. Product Backlog is a living document. Changes in business requirements, market conditions, technology, and staffing cause changes in the Product Backlog. To minimize rework, only the highest priority items need to be detailed out. The Product Backlog items that will occupy the Teams for the upcoming several Sprints are fine-grained, having been decomposed so that any one item can be done within the duration of the Sprint.

TIP

Scrum Teams often spend 10% of each Sprint grooming the product backlog to meet the above definition of the Product Backlog. When groomed to this level of granularity, the Product Backlog items at the top of the Product Backlog (highest priority, greatest value) are decomposed so they fit within one Sprint. They have been analyzed and thought through during the grooming process. When the Sprint Planning meeting occurs, these top priority Product Backlog items are well understood and easily selected.

TIP

Acceptance tests are often used as another Product Backlog item attribute. They can often supplant more detailed text descriptions with a testable description of what the Product Backlog item must do when completed.

Multiple Scrum Teams often work together on the same product. One Product Backlog is used to describe the upcoming work on the Product. A Product Backlog attribute that groups items is then employed. Grouping can occur by feature set, technology, or architecture, and it is often used as a way to organize work by Scrum Team.

The Release Burndown graph records the sum of remaining Product Backlog estimated effort across time. The estimated effort is in whatever unit of work the Scrum Team and organization have decided upon. The units of time are usually Sprints.

Product Backlog item estimates are calculated initially during Release Planning, and thereafter as they are created. During Product Backlog grooming they are reviewed and revised. However, they can be updated at any time. The Team is responsible for all estimates. The Product Owner may influence the Team by helping understand and select trade-offs, but the final estimate is made by the Team. The Product Owner keeps an updated Product Backlog list Release Backlog Burndown posted at all times. A trend line can be drawn based on the change in remaining work.

TIP

In some organizations, more work is added to the backlog than is completed. This may create a trend line that is flat or even slopes upwards. To compensate for this and retain transparency, a new floor may be created when work is added or subtracted. The floor should add or remove only significant changes and should be well documented.

TIP

The trend line may be unreliable for the first two to three Sprints of a release unless the Teams have worked together before, know the product well, and understand the underlying technology.

SPRINT BACKLOG AND SPRINT BURNDOWN

The Sprint Backlog consists of the tasks the Team performs to turn Product Backlog items into a "done" increment. Many are developed during the Sprint Planning Meeting. It is all of the work that the Team identifies as necessary to meet the Sprint goal. Sprint Backlog items must be decomposed. The decomposition is enough so changes in progress can be understood in the Daily Scrum. One day or less is a usual size for a Sprint Backlog item that is being worked on.

The Team modifies Sprint Backlog throughout the Sprint, as well as Sprint Backlog emerging during the Sprint. As it gets into individual tasks, it may find out that more or fewer tasks are needed, or that a given task will take more or less time than had been expected. As new work is required, the Team adds it to the Sprint Backlog. As tasks are worked on or completed, the estimated remaining work for each task is updated. When tasks are deemed unnecessary, they are removed. Only the Team can change its Sprint Backlog during a Sprint. Only the Team can change the contents or the estimates. The Sprint Backlog is a highly visible, real time picture of the work that the Team plans to accomplish during the Sprint, and it belongs solely to the Team.

Sprint Backlog Burndown is a graph of the amount of Sprint Backlog work remaining in a Sprint across time in the Sprint. To create this graph, determine how much work remains by summing the backlog estimates every day of the Sprint. The amount of work remaining for a Sprint is the sum of the work remaining for all of Sprint Backlog. Keep track of these sums by day and use them to create a graph that shows the work remaining over time. By drawing a line through the points on the graph, the Team can manage its progress in completing a Sprint's work. Duration is not considered in Scrum. Work remaining and date are the only variables of interest.

TIP

Whenever possible, hand draw the burndown chart on a big sheet of paper displayed in the Team's work area. Teams are more likely to see a big, visible chart than they are to look at Sprint burndown chart in Excel or a tool.

One of Scrum's rules pertains to the purpose of each Sprint, which is to deliver increments of potentially shippable functionality that adheres to a working definition of "done."

Done

Scrum requires Teams to build an increment of product functionality every Sprint. This increment must be potentially shippable, for Product Owner may choose to immediately implement the functionality. To do so, the increment must be a complete slice of the product. It must be "done." Each increment should be additive to all prior increments and thoroughly tested, ensuring that all increments work together.

In product development, asserting that functionality is done might lead someone to assume that it is at least cleanly coded, refactored, unit tested, built, and acceptance tested. Someone else might assume only that the code has been built. If everyone doesn't know what the definition of "done" is, the other two legs of empirical process control don't work. When someone describes something as *done*, everyone must understand what *done* means.

Done defines what the Team means when it commits to "doing" a Product Backlog item in a Sprint. Some products do not contain documentation, so the definition of "done" does not include documentation. A completely "done" increment includes all of the analysis, design, refactoring, programming, documentation and testing for the increment and all Product Backlog items in the increment. Testing includes unit, system, user, and regression testing, as well as non-functional tests such as performance, stability, security, and integration. Done includes any internationalization. Some Teams aren't yet able to include everything required for implementation in their definition of done. This must be clear to the Product Owner. This remaining work will have to be done before the product can be implemented and used.

Tip

"Undone" work is often accumulated in a Product Backlog item called "Undone Work" or "Implementation Work." As this work accumulates, the Product Backlog burndown remains more accurate than if it weren't accumulated.

FINAL THOUGHTS

Some organizations are incapable of building a complete increment within one Sprint. They may not yet have the automated testing infrastructure to complete all of the testing. In this case, two categories are created for each increment: the "done" work and the "undone" work. The "undone" work is the portion of each increment that will have to be completed at a later time. The Product Owner knows exactly what he or she is inspecting at the end of the Sprint because the increment meets the definition of "done" and the Product Owner understands the definition. "Undone" work is added to a Product Backlog item named "undone work" so it accumulates and correctly reflects on the Release Burndown graph. This technique creates transparency in progress toward a release. The inspect and adapt in the Sprint Review is as accurate as this transparency.

For instance, if a Team is not able to do performance, regression, stability, security, and integration testing for each Product Backlog item, the proportion of this work to the work that can be done (analysis, design, refactoring, programming, documentation, unit and user testing) is calculated. Let's say that this proportion is six pieces of "done" and four pieces on "undone." If the Team finishes a Product Backlog item of six units of work (the Team is estimating based on what it knows how to "do"), four is added to the "undone work" Product Backlog item when they are finished.

Sprint by Sprint, the "undone" work of each increment is accumulated and must be addressed prior to releasing the product. This work is accumulated linearly although it actually has some sort of exponential accumulation that is dependent on each organization's characteristics. Release Sprints are added to the end of any release to complete this "undone" work. The number of Sprints is unpredictable to the degree that the accumulation of "undone" work is not linear.