A Viable Systems Engineering Approach

Presented by:
Dick Carlson (richard.carlson2@boeing.com)
Philip Matuzic (philip.j.matuzic@boeing.com)

Introduction

- This presentation addresses systems engineering applying Agile practices to non software-centric projects

- An experience model created from a case study shows how the failures of a large program was able to show significant improvements applying specific Agile practices

Problem Statement

- **The program was running over budget and schedules were not being met – program management demanded rapid development techniques – the current methodology was not meeting the needs of the sponsors**

- **The Agile approach *Scrum* was selected as the only viable solution for managing project execution and implementation to improve efficiency and productivity. This approach worked well, but not without some daunting problems.**

- **Requirements were slowly evolving and a *product backlog* did not exist, so the core team decided to field teams trained in Agile practices staffed with self-organized and motivated individuals in a series of short iterations to create a product backlog consisting of fully-developed and prioritized requirements**

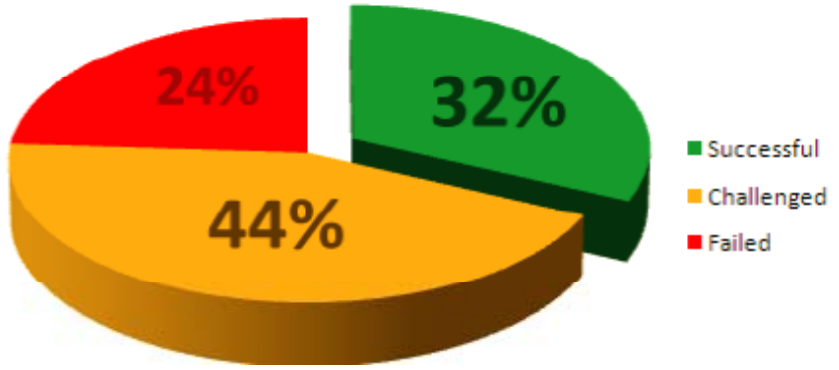
The Approach

- Teams were staffed with functional and domain analysts, and if available, one developer
- Requirements were written directly into a configuration management tool by the team
- Every requirement included one or more success criteria and failure conditions to establish a validation method
- Iterations were conducted in 4-week (30-calendar day) durations and ended with the team reviewing requirements with the Product Owner, key stakeholders, and domain experts

Industry-wide High Project Failure Rates

- **Backward Trend in Software Project Success**
  - Failed and challenged projects hover around 70%
  - High failure rate due to inability to cope with change
  - Big projects exacerbate challenge and failure potential

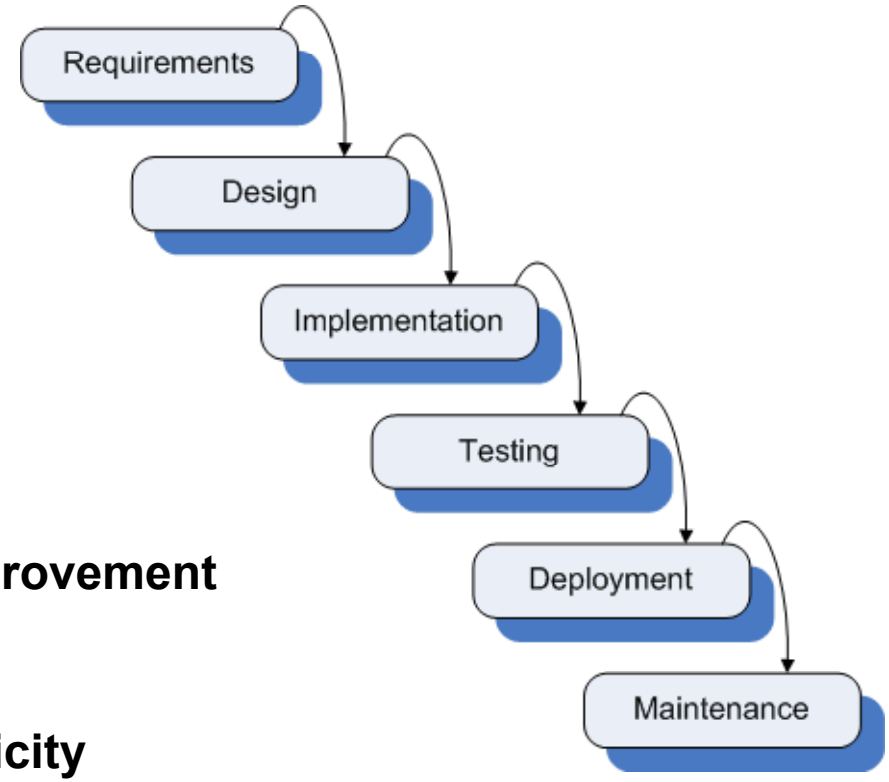
- **2009 CHAOS Standish Chaos Report**
  - **32% of software development projects were successful**
  - **44% challenged**
  - **24% failed to meet schedule or budget**



Johnson, J., et al. (2009), *Chaos Summary 2009*, Boston, MA: Standish Group International

Why Aren't the Successes Higher?

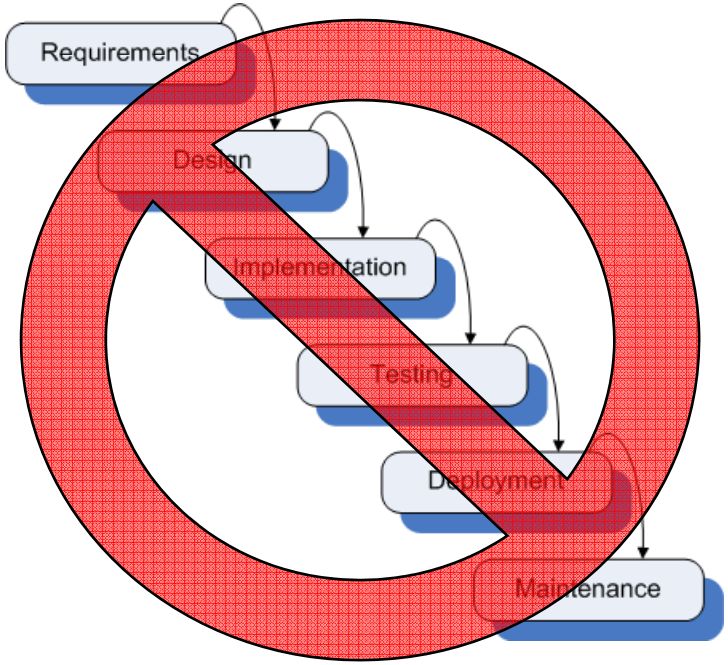
Historic processes were subjected to the weaknesses of the Waterfall model….

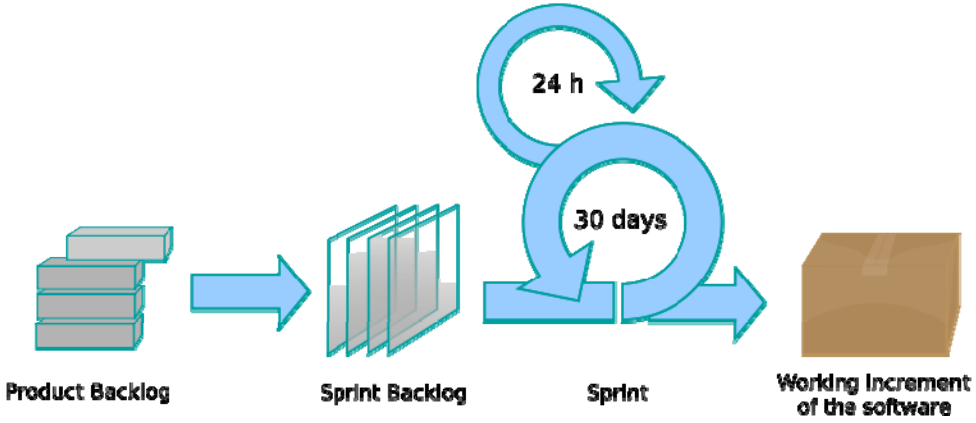


The Waterfall Model

- **Has distinct phases**
- **Lacks feedback loops for improvement**
- **Includes sequential phases**
- **Handoffs to different teams**
- **Has an appealing air of simplicity**
- **Project managers like the easily tracked milestones**

The Need for Lean Systems Development





- **The Waterfall Model relies on up-front requirements and designs that are:**
  - Complete
  - Mature
  - Sufficient
  - Stable

- **A more effective model is needed for:**
  - Changing requirements
  - Incremental development
  - Emerging needs
  - Uncertainty

A Viable Alternative

Manifesto for Agile Software Development

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

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

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

Source: Agile Alliance (http://www.agilealliance.org/)

The Essence of Agile

- **Agile is an iterative and incremental approach to the development of any work product**
  - **Agile is normally focused on building software products, but some practices can be applied to any project**
  - **We have Boeing Systems Engineering experience in developing requirements and a Product Backlog**

- **Agile is a highly collaborative activity conducted by self-organizing teams with "just enough" ceremony to**
  - **produce high quality products in a cost-effective and timely manner**
  - **meet the changing needs of customers and key stakeholders**

Agile Principles and Practices

- **Agile is a philosophy defined by principles and practices that drive quick delivery of quality software and encourage user feedback**
  - **Most of which are easily applied to Agile Systems Engineering**

Principles:

- Customer Satisfaction
- Frequent Delivery/Deployment
- Motivated Team
- Working Software
- Technical Excellence
- Emergent Design
- Embrace Change
- Collaboration
- High Bandwidth
- Sustainable Pace
- Simplicity
- Continuous Improvement

Practices:

- Close customer collaboration
- Daily stand-up meetings
- Continuous integration
- Automated testing
- Planning and estimating
- Short iterations
- Test-driven development
- Prioritized requirements
- Product demonstrations/reviews
- Self-organized teams

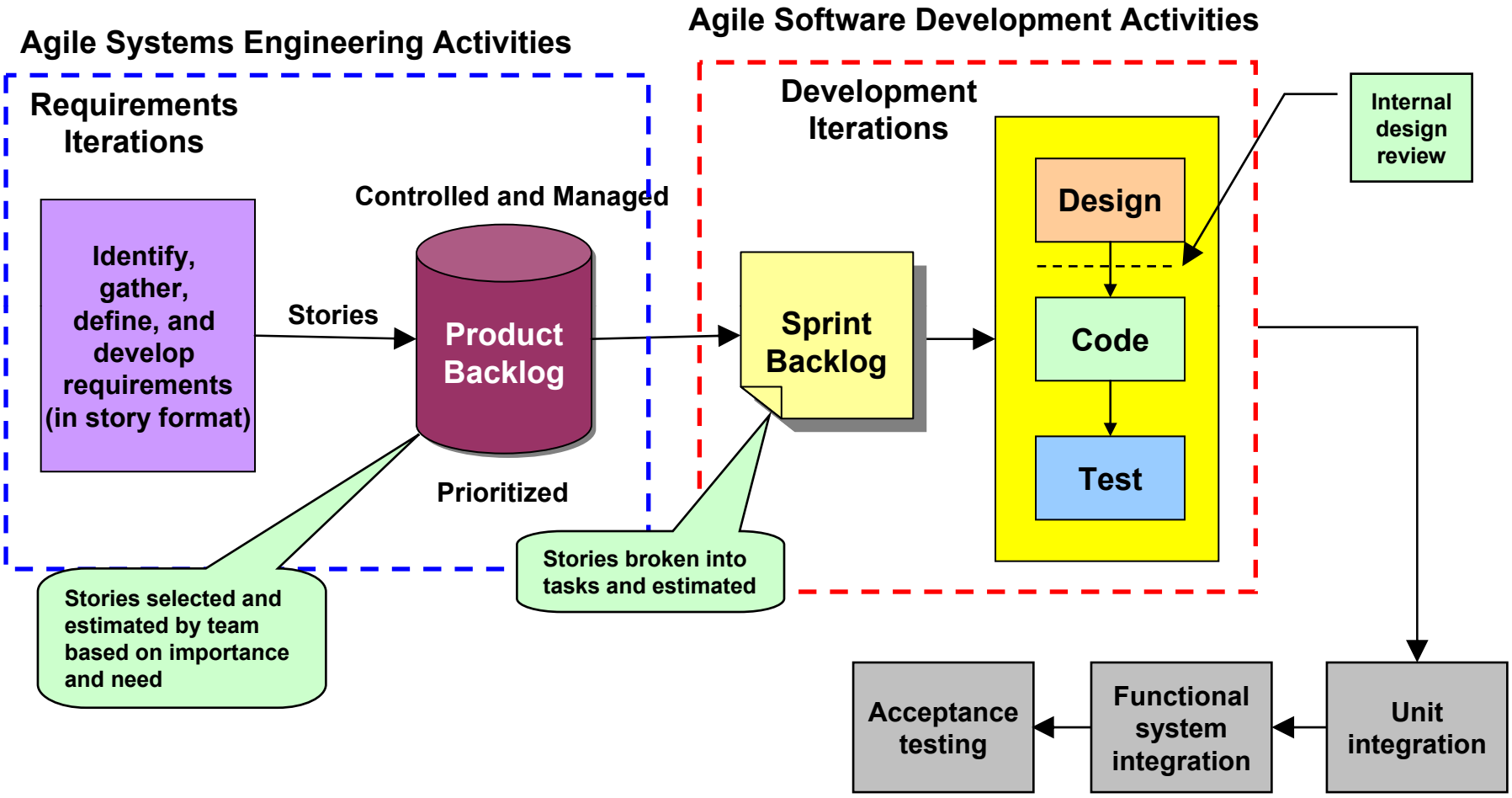
Source: Agile Alliance (http://www.agilealliance.org/)

Scrum Applied to Systems Engineering

- **Scrum is a framework for managing a project that focuses on delivering the highest business value in the shortest time through the use of simple roles, artifacts, and ceremonies**
- **A Scrum project is a series of iterations or Sprints where every 2-4 weeks produces fully developed requirements, functional analyses, and system-level architecture where decisions are made**
- **Teams are self-organized and fully empowered to do whatever it takes to complete all iteration work**
- **The customer, users, and/or business needs set the priorities**
- **Scrum is simple and straightforward**
  - **Practices, artifacts, and rules are few and easy to learn**
  - **No complicated process descriptions**
  - **No individual assignments – team selects all work from the prioritized backlog**

Source: Scrum Alliance (http://www.scrumalliance.org/)

Systems/Software Engineering Experience Model



Scrum Values

- **Scrum is based on a set of fundamental values that make up the backbone of its practices**
  - **Commitment** – Be willing to commit to a goal
    - Scrum provides people all the authority they need to meet their commitments
  - **Focus** – Do what you said you will do
    - Focus efforts and skills on doing the work committed
  - **Openness** – Everything you do can be seen by everyone
    - Scrum is transparent by keeping everything about a project visible
  - **Respect** – Individuals are shaped by their experiences
    - It's important to respect the diversity of people who comprise a team
  - **Courage** – Have the courage to commit, act, be open, and expect respect
    - Celebrate and enjoy the journey

Source: Scrum Alliance (http://www.scrumalliance.org/)

Why Use Scrum?

- Increases team productivity and reduces cost and cycle times
- Leverages chaos
  - Products are built through a series of manageable chunks
  - Progress is made – even when requirements are not stable
  - Everything is visible to everyone!
  - Improves Team communication
  - Enables continuous improvement
  - Customers and stakeholders see on-time delivery of increments and obtain frequent feedback on how the product actually works
- Establishes a relationship with customers and stakeholders that builds trust and stimulates knowledge growth
- Creates a culture where everyone expects the project to succeed

Source: Scrum Alliance (http://www.scrumalliance.org/)

BOEING

Scrum Works!

- Uses an iterative, incremental approach to product development
- Relies on interactions between customers, users, and team
- Because team is committed and focused – multi-tasking is drastically reduced and bad multi-tasking is eliminated
- Easy to apply and implement on any project and discipline – for example, Agile Systems Engineering

Agile Systems Engineering

Systems Engineering Activities

- Systems engineering includes defined and repeatable processes that produce specific and supportive artifacts
  - That ensures integration with other Engineering disciplines and domains
  - And ensures integration among disciplines/functions, design, manufacturing, supply chain, test, product support, etc.
  - To produce a system integrated among all systems and components
  - Applied over and addressing the entire lifecycle of the product from requirements development through disposal

Key Focus of Systems Engineering

- Two key thrusts in program-level systems engineering domains:
  - Engineering the system to define the technical solution, and
  - Planning and control supporting program management
- Engineering the system requires:
  - Requirements Analysis / Definition / Validation
  - Functional Analysis and Allocation
  - Synthesis of Designs
  - Evaluation of Alternatives
  - Requirements Verification
- Planning and control requires:
  - Organizing and Planning (e.g. Organizing the program, and development of the SEMP, IMP/IMS)
  - Requirements Management
  - Interface Management
  - Baseline Management
  - Affordability
  - Decision Making (e.g. Risk Management, Trade Studies, TPMs)
  - Metrics Management
  - Reviews

Bridging Agile Principles and Practices to Systems Engineering

- The benefits of experience through the application of common Agile practices apply well to systems engineering activities. That is:
  - Small, self-organized teams producing work products incrementally through a series of short iterations
  - Commitment by all team members and sponsors
  - Intense iteration planning sessions that identify what will be completed and how the team plans on completing it
  - Time-boxed daily standup meetings
  - Developed requirements and product backlog are reviewed
  - Team reflects what brought the highest value during the iteration through a retrospective that includes the entire team

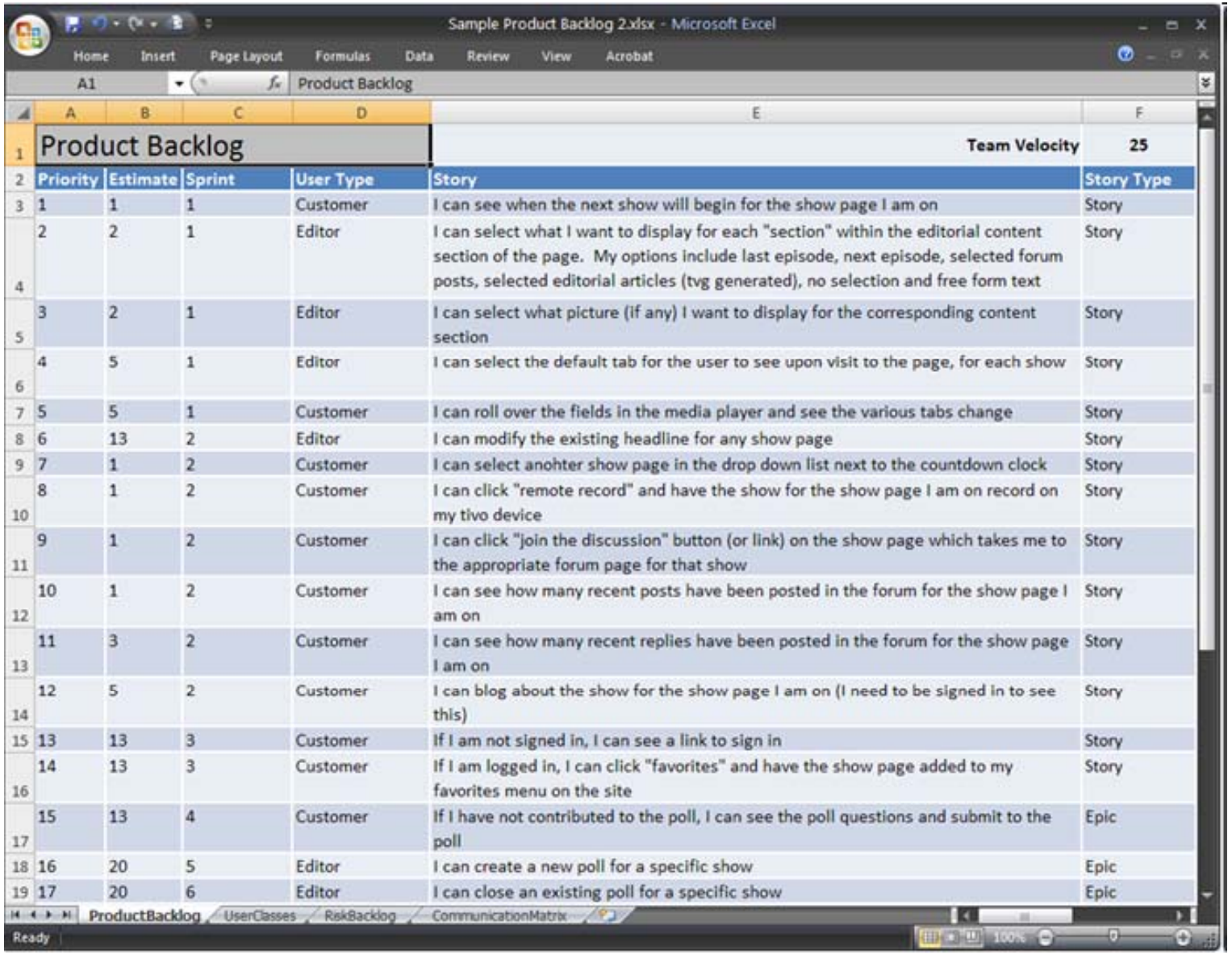
- The Agile systems engineering approach leverages integrated engineering by employing the same vocabulary and artifacts in an evolving, iterative approach
  - Software and systems engineers become collaborators that sincerely consider alternatives and take actions that lead to change

BOEING

Agile Systems Engineering

- An Agile project activity that defines and develops requirements, and creates a Product Backlog for an Agile Software Development project
  - The Scrum framework is used to manage project teams
  - The major difference is the finished product
    - At the end of each iteration, the Product Backlog "baseline" has evolved into a clearer and more complete set of prioritized functionality
    - Acceptance tests are written
    - Attributes and constraints are identified
    - Requisite documentation is created incrementally
    - All data is configuration managed and controlled
  - The Product Backlog is prioritized by the Product Owner and other domain stakeholders, and made available to the development team

Product Backlog Example



	A	B	C	D	E	F
1	Product Backlog				Team Velocity	25
2	Priority	Estimate	Sprint	User Type	Story	Story Type
3	1	1	1	Customer	I can see when the next show will begin for the show page I am on	Story
4	2	2	1	Editor	I can select what I want to display for each "section" within the editorial content section of the page. My options include last episode, next episode, selected forum posts, selected editorial articles (tvg generated), no selection and free form text	Story
5	3	2	1	Editor	I can select what picture (if any) I want to display for the corresponding content section	Story
6	4	5	1	Editor	I can select the default tab for the user to see upon visit to the page, for each show	Story
7	5	5	1	Customer	I can roll over the fields in the media player and see the various tabs change	Story
8	6	13	2	Editor	I can modify the existing headline for any show page	Story
9	7	1	2	Customer	I can select anohter show page in the drop down list next to the countdown clock	Story
10	8	1	2	Customer	I can click "remote record" and have the show for the show page I am on record on my tivo device	Story
11	9	1	2	Customer	I can click "join the discussion" button (or link) on the show page which takes me to the appropriate forum page for that show	Story
12	10	1	2	Customer	I can see how many recent posts have been posted in the forum for the show page I am on	Story
13	11	3	2	Customer	I can see how many recent replies have been posted in the forum for the show page I am on	Story
14	12	5	2	Customer	I can blog about the show for the show page I am on (I need to be signed in to see this)	Story
15	13	13	3	Customer	If I am not signed in, I can see a link to sign in	Story
16	14	13	3	Customer	If I am logged in, I can click "favorites" and have the show page added to my favorites menu on the site	Story
17	15	13	4	Customer	If I have not contributed to the poll, I can see the poll questions and submit to the poll	Epic
18	16	20	5	Editor	I can create a new poll for a specific show	Epic
19	17	20	6	Editor	I can close an existing poll for a specific show	Epic

Example only – contains no content
Screen shot of an example product backlog in Excel

BOEING

Getting Started

- **Management, sponsors, and entire Team must commit**
  - Commitments made are expected to be fulfilled
  - Broken commitments guarantee project failure

- **A good Product Owner is vital**
  - Must have strong domain knowledge
  - Must be Scrum trained
  - Must be available and willing to commit

BOEING

What Else is Needed?

- **Scrum Master and Team must be trained in Scrum**
  - **Scrum Master facilitates meetings, reviews, and retrospectives**
  - **Development Teams build software; Requirements Teams write *stories***
  - **Training is vital to the successful execution of Agile practices**
  - **Insufficient or no training guarantees project failure**

- **Development Teams need a Product Backlog**
  - **Requirement development teams build the Product Backlog**
    - **Consists of fine-grained requirements**
    - **Developers estimate the size of each requirement**

Requirements Expressed in Story Format

- A story describes functionality that is valuable to a customer or users of the system. A story:
  - Provides a clear and concise look at what is needed
  - Is a unit of development "work"
  - Expresses a need in a common language
  - Is a brief discussion about the need that helps flesh out details
  - Includes tests that will validate the need
  - Focuses only on the size of the work – not how long it will take

Story Template

As a < Role >

I want to < Goal >

So that < Reason >

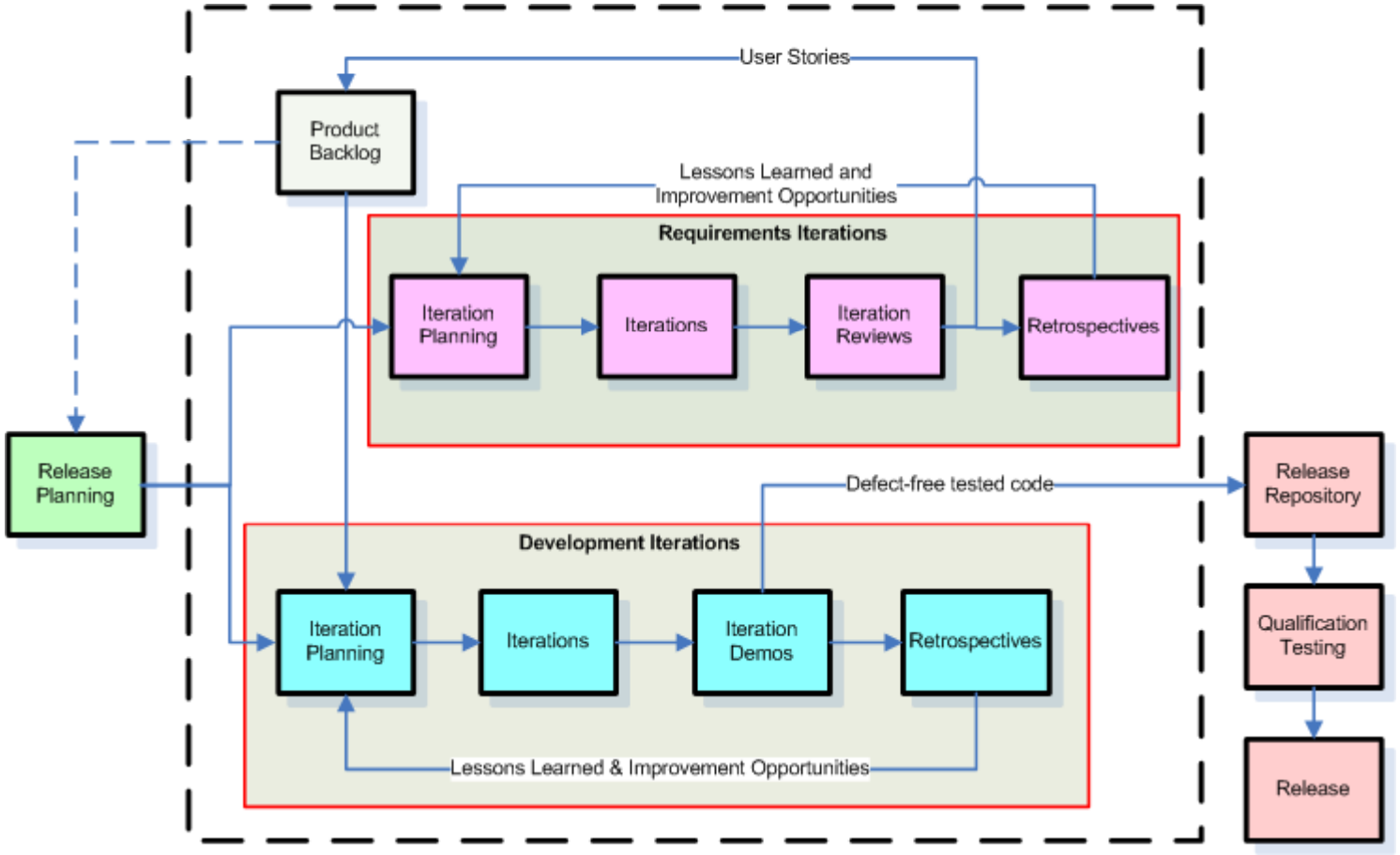
Story Example

As a member of an Agile team,

I want to learn how to write clear and concise stories,

So that they possess the "INVEST" attributes

Software/Systems Engineering Collaboration Model



BOEING

Mapping Scrum Practices to CMMI Level 3

- **Most of Scrum maps well to CMMI Level 2 process areas:**
  - **CM:** In an Agile environment, it is easy to add a layer of CM to protect work products using common tools and technologies.
  - **PPQA:** Some basic PPQA activities are being done naturally when the Scrum Master checks or when QA monitors that the Scrum process is being followed. Other PPQA activities are completed when a team performs peer reviews and validation.
  - **SAM:** There are no practices in Scrum that deal with the selection and management of suppliers.

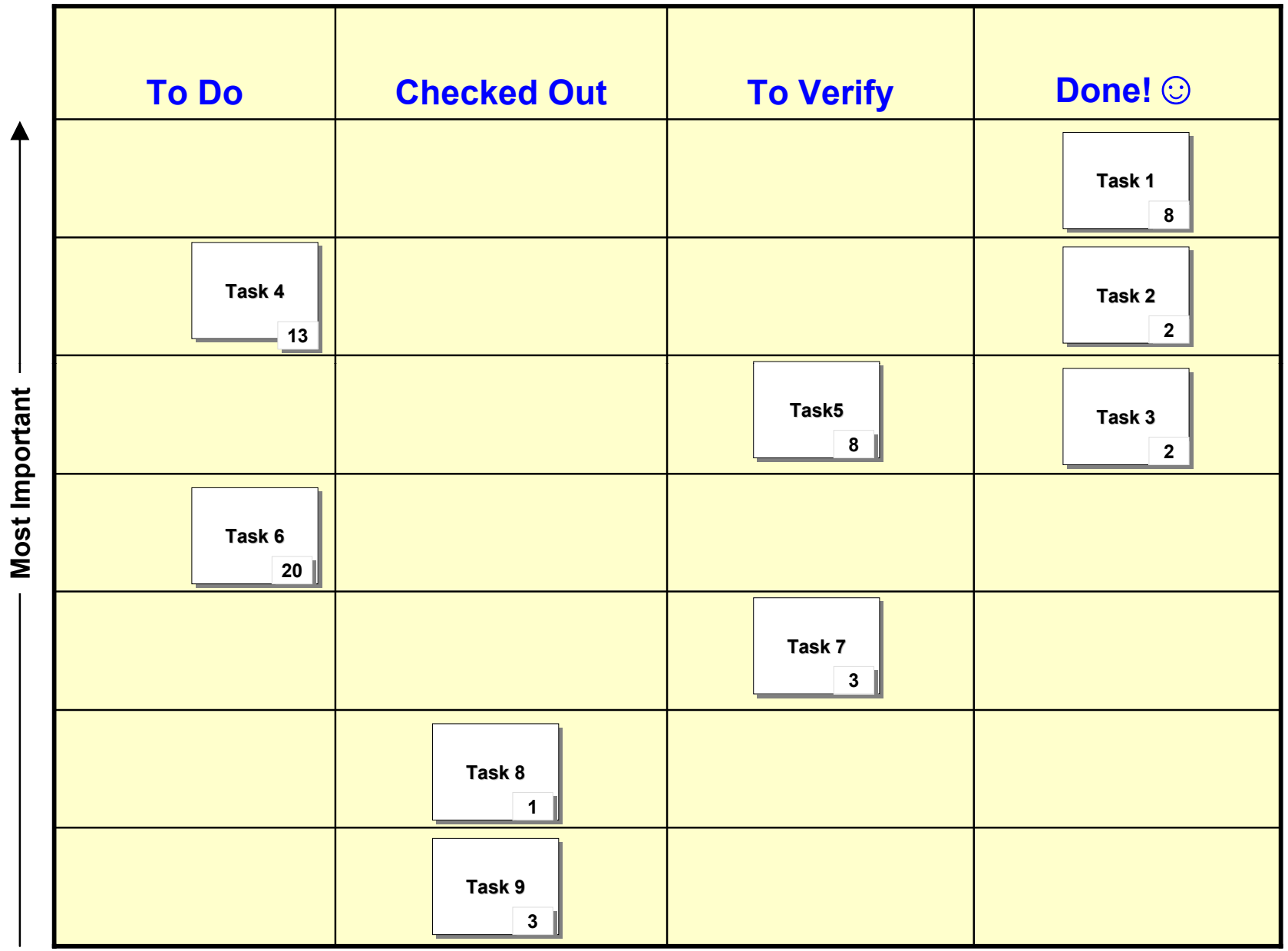
- **There are some limited applications of CMMI Level 3 used in Scrum including:**
  - **RSKM:** Risk management practices implemented at the organization or program level can be applied without much difficulty
  - **VAL:** Validation is confirmed through formal reviews
  - **VER:** Verification is performed through team and iteration reviews

CMMI Levels 2 and 3 source: http://www.sei.cmu.edu/cmmi/tools/peoplecmm/

2-Week Iteration Battle Rhythm

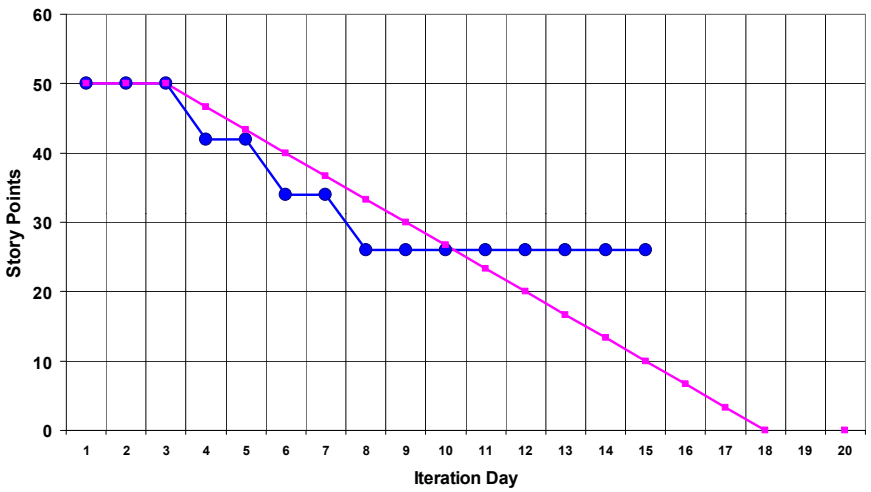
- 10 work-day iterations
- Daily meetings/work sessions decided by the team – Day 2 through 9
- Iteration Review – early Day 10
- Retrospective – after Iteration Review on Day 10

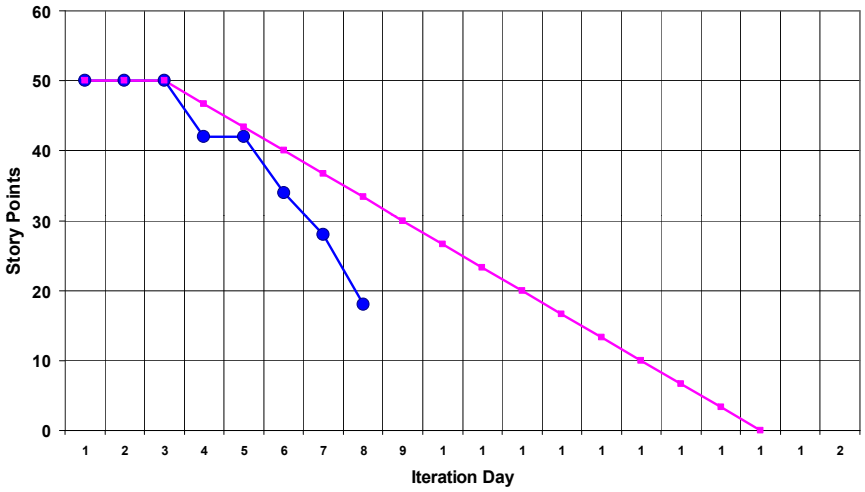
Project Taskboard



Agile Team Metrics

Provides a transparent view of iteration progress and health





- Projects that show consistent straight lining are in danger of not completing work by iteration's end
- Work completed significantly ahead of planned completion may indicate additional work can be added to the iteration

BOEING

Some Lessons Learned

- Focused, face-to-face iteration sessions improve team synergy
- Appoint someone to track time-boxed activities
- Scrum Master should work with management to resolve unplanned organizational impediments that affect team productivity
- Avoid side discussions that disrupt team momentum
- Attend all daily stand-ups (best way to know health and progress of team)
- Working as a team helps everyone understand tasks
- Team consultations with domain experts help clarify ambiguity
- Ensure requisite architecture, infrastructure, technologies, and tools are in place before starting team activities – and all must be trained

BOEING®

Questions?