Build Agile Knowledge - Participate in a sprint!

Presenters:
Almir Drugovic and **Terri Spratt**

About the Presenters

Almir Drugovic (adrugovic@gmail.com; LinkedIn) has over fifteen years of professional and technical experience, with over ten years experience managing large organizational initiatives and technology projects. Currently, Almir is responsible for enterprise adoption, scaling and continuous improvement of Agile at Nokia Location & Commerce, and is focused on helping the company transition to Agile. Almir is one of the few Agile practitioners to have hands-on experience in a large scale Agile adoption effort, and his specialties include Waterfall to Lean and Agile transition, Business Process Management, large scale program management, business process reengineering, and maximizing value delivery by implementing Engineering Practices.

Terri Spratt (LinkedIn) is a professional Agile coach, trainer and facilitator, and is also Agile Transition Manager at NOKIA in Chicago. She has over 15 years of software development experience, and specializes in helping organizations transition from waterfall to Agile/Scrum, including implementing the many cultural changes necessary to make a successful transition. Terri is experienced in coaching teams to implement, continually improve and sustain Agile best practices for the long term. She is also a professional facilitator and trainer, leading workshops and classes that help teams and organizations move to a higher level of Agility.

Our Backlog

- Why Agile *
- How Agile is Quality and Risks
- Measuring Value *
- Where to Start
- How to Scale Agile to the Enterprise Level *
- Role of Engineering Practices in Agile Success
- Co-located Teams in a Global Organization
- The Role of Managers in an Agile Environment
- Evolutionary and Emergent Architecture
- How Coaching Evolves as Agile Scales
- Evolving the Culture of an Organization
- Implementing Agile in Non-IT Areas
- Sustaining Agile for the Long Term
- Agile & Lean *
- Sprint Types *
- More about Objective, Epic and Sprint *
- The story behind Agile Manifesto *

Agile and Lean

Lean Principles

Agile has its roots in the Lean Manufacturing Principles developed by Toyota

Lean Principles

- Eliminate Waste. Build Quality In
- Create Knowledge
- Defer Commitment
- Deliver Fast
- Respect People
- Improve the System

Seven Wastes

1. Partially done work
2. Extra features
3. Relearning
4. Handoffs
5. Task switching
6. Delays
7. Defects

The story behind Agile Manifesto

Unification Under the Agile Manifesto

Many Adaptive Methods unified under the term "Agile" in 2001 when seventeen people involved in such practices as XP, Scrum, DSDM, FDD, and other methods gathered to create the Agile Manifesto

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

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.

What Is Agile: The Manifesto’s Twelve Values

The Agile Manifesto also contains twelve detailed values

1. **Our highest priority is to satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. **Business people and developers must work together** daily throughout the project.
5. **Build projects around motivated individuals.** Give them the environment and support they need, and trust them to get the job done.
6. **The most efficient and effective method of conveying information** to and within a development team is face-to-face conversation.
7. **Working software** is the primary measure of progress.
8. **Agile processes promote sustainable development.** The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. **Continuous attention to technical excellence** and good design enhances agility.
10. **Simplicity**--the art of maximizing the amount of work not done--is essential.
11. **The best architectures, requirements, and designs** emerge from self-organizing teams.
12. **At regular intervals, the team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

Sprint Types

Sprint Types

Although the Sprint was originally defined to result in an increment of software, the model can be applied to all stages of enterprise software development

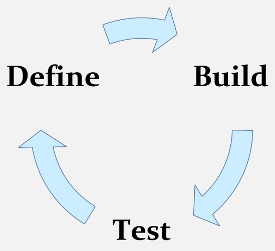
Setup Sprint

The Setup Sprint includes activities which need to happen between the completion of N5/N4 and the start of the first N3 Feature Sprint:

- **Spikes** - Research, testing, or a proof-of-concept of an unknown, in order to reduce risk and refine estimates
- **Environment** – Activities to set up the environment for N3 such as procuring hardware or setting up servers
- **Schedule** – Securing team resources required for N3

Feature Sprint (focus of this module)

- The Feature Sprint contains all activities to define in detail, build, and test a unit of software which can be demoed or released for hardening



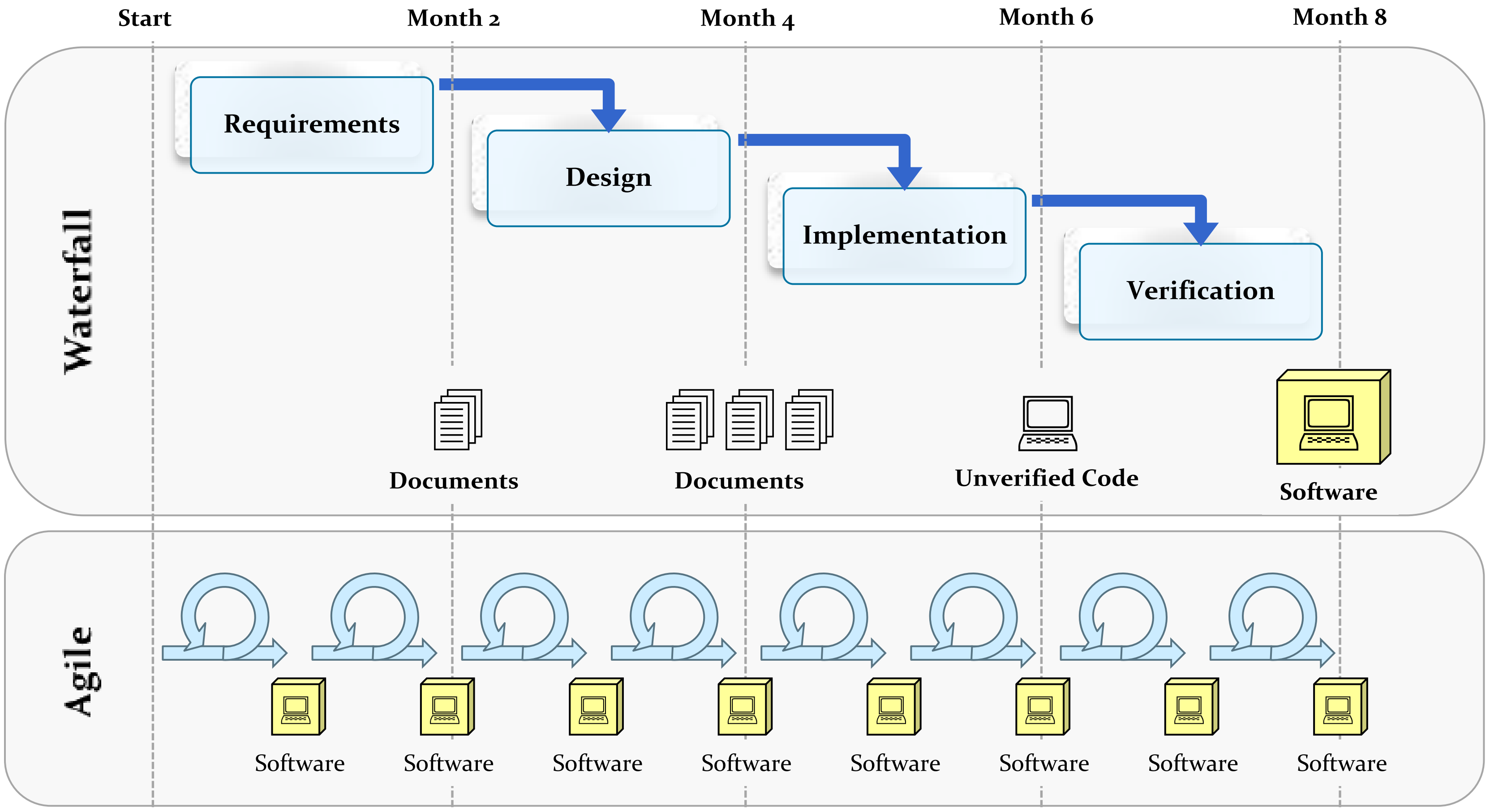
Hardening Sprint

- The Hardening Sprint consists of two major activities:
- Integration of components, architecture, and infrastructure
- Preparation of software for an internal Release (light hardening) or production Release (heavy hardening)

Why Agile

Why Agile

The Agile methodology strives to continuously deliver small increments of functionality with business value. This gives continuous visibility and continuous delivery of value.



Why Agile (2 of 2)

Agile has many benefits beyond speed-to-market

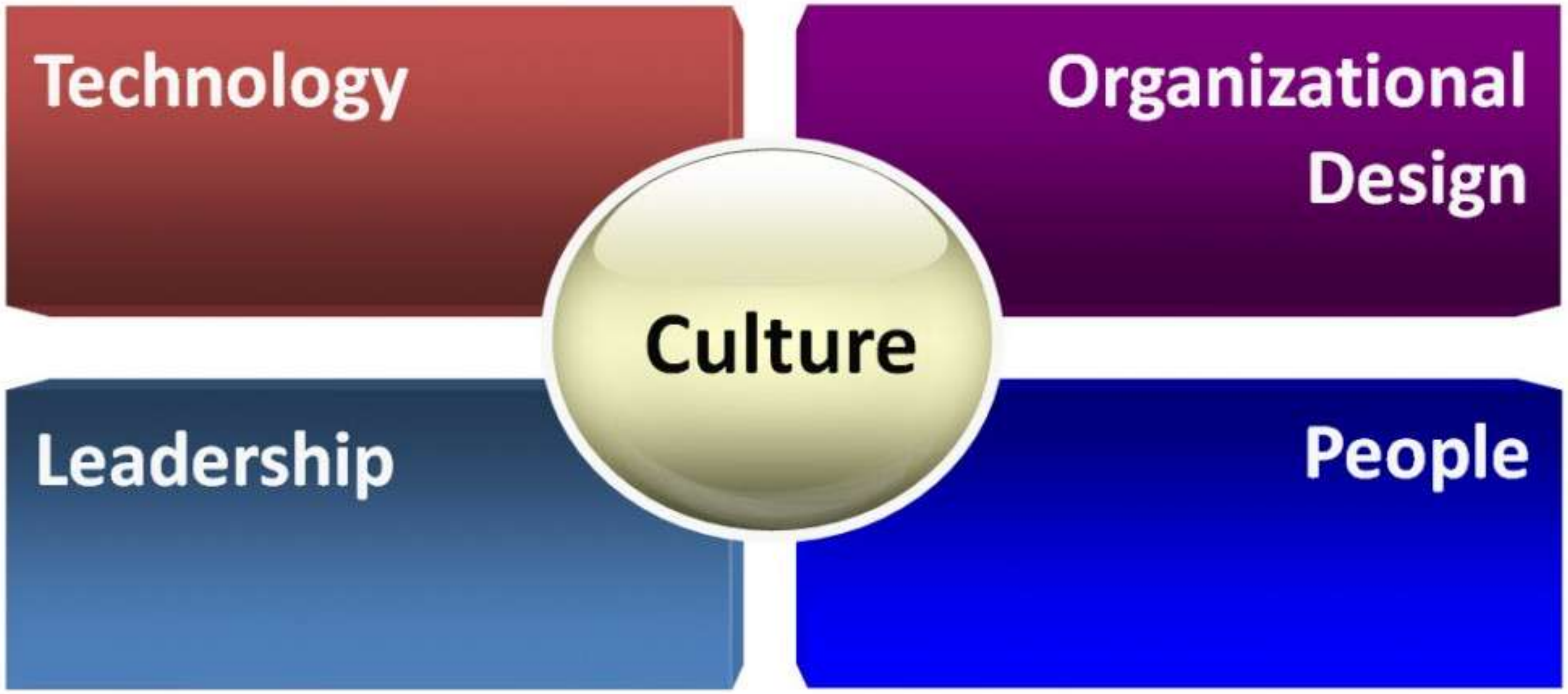
Key Benefits

- Delivery of working increments of software in short, frequent iterations ("Sprints")
- Greater adaptability to changing market conditions through re-evaluation of requirements at the beginning of each Sprint
- Higher quality through early, frequent testing
- Heavy user involvement and frequent demos to confirm that Product Management, R&D, and DMO are aligned
- Greater visibility throughout the development process

Value (per industry standards)

1. Greater ROI with Incremental Return
2. Speed to Market with Greater Adaptability
3. Lower Cost
4. Reduced Risk
5. Higher Quality
6. Higher Customer Satisfaction
7. Greater Visibility
8. Higher Employee Satisfaction

Agile Transition Success Factors



All five success factors are essential to a successful Agile transition

- Courtesy of Jorgen Hesselberg

How to Scale Agile in Large Companies

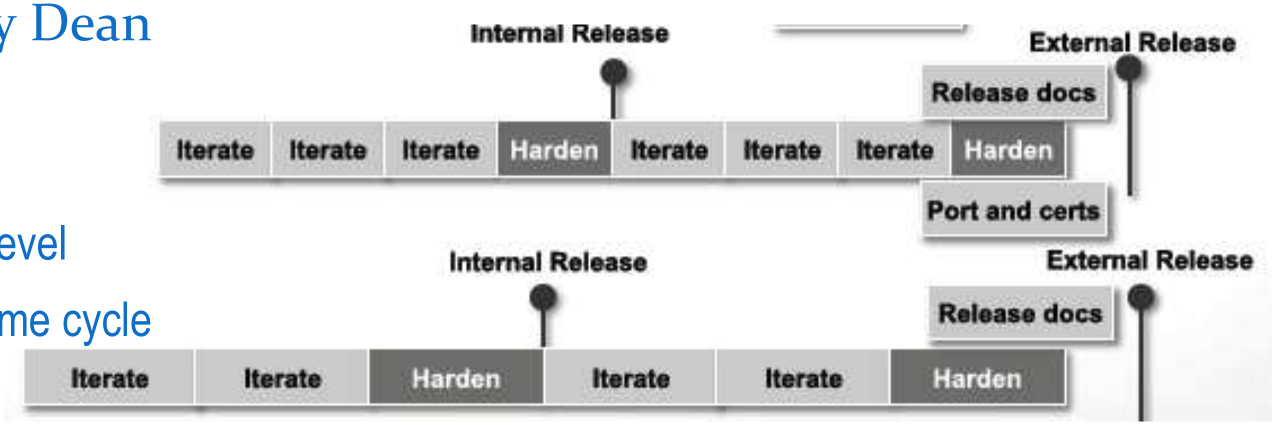
What Problem Are We Trying to Solve?

- Teams are finding it difficult to manage dependencies across teams.
- Teams working on dependent products or features are sometimes receiving conflicting priorities from different Product Owners.
- Teams are working at different cadences, making it difficult to integrate and test products developed by multiple teams.
- System level visibility not always accessible outside of teams, making longer term planning and cross-team coordination difficult.

Solution Release Train (by Dean Leffingwell):

Create Release Programs:

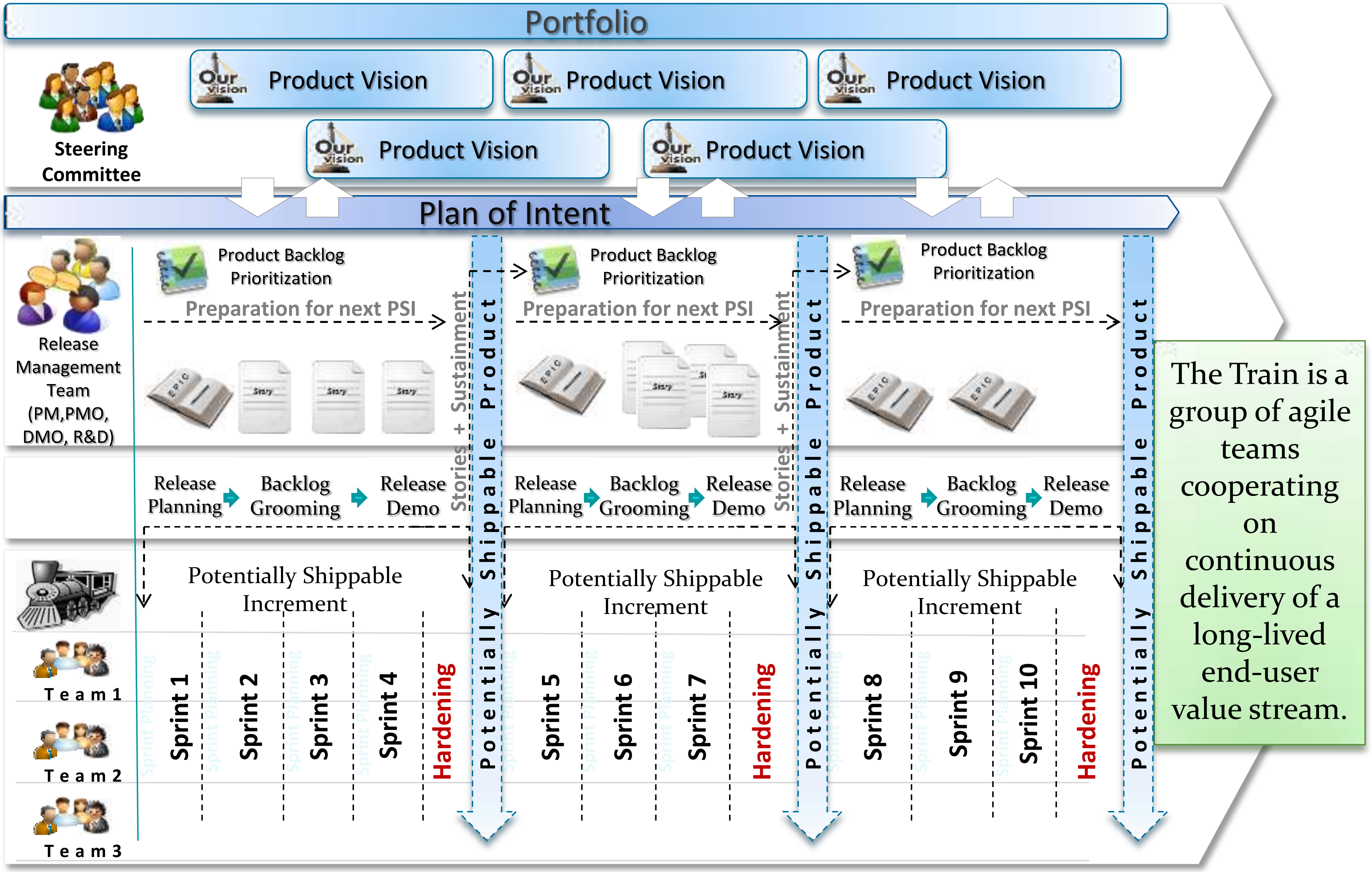
- Plan releases at program level
- Iterate and integrate on same cycle across dependent teams



Need for Two Levels of Planning

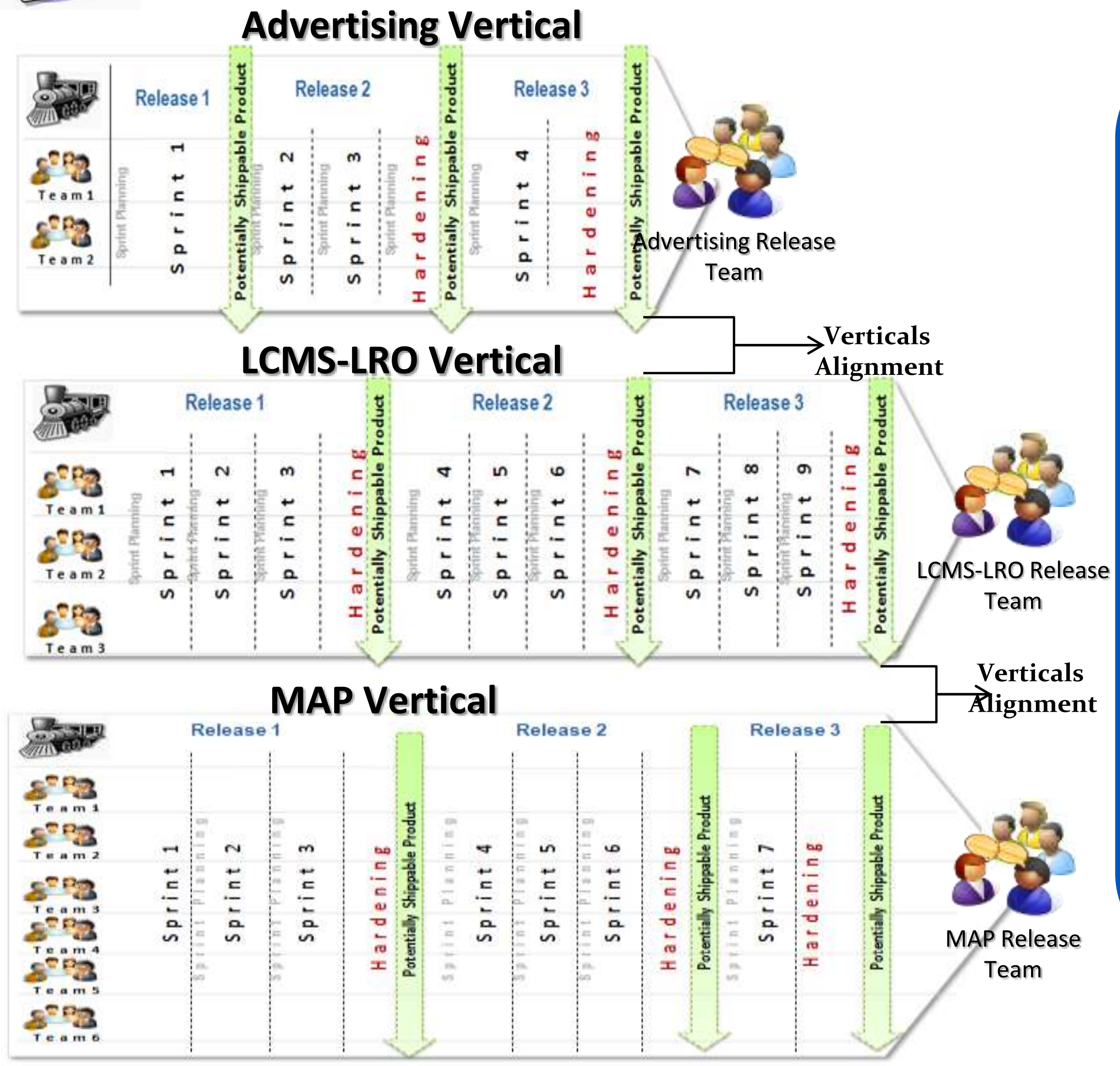
- Plan Releases based on Value Stream
  - Release theme sets overall objective
  - Global alignment and prioritization
  - Three to six months planning horizon; "Plan of Intent"
  - Prioritized Epic sets define proposed/estimated content
  - High visibility and confidence near term (Release "next" and "next +1").
  - Lower visibility longer term
- Plan Iterations at the Story Level
  - Alignment and prioritization
  - Four to six weeks planning horizon
  - Teams identify user stories for iterations
  - Adapt and adjust at each iteration

Agile Framework



- Courtesy of Dean Leffingwell (www.leffingwell.org)

Planning Release Plan – Program Alignment



- Release Management Team is responsible for managing the Release Plan.
- Cadence and timing of the Release Train are determined based on business needs, technical complexity and size of the vertical.
- Release Team is responsible for aligning goals across verticals.

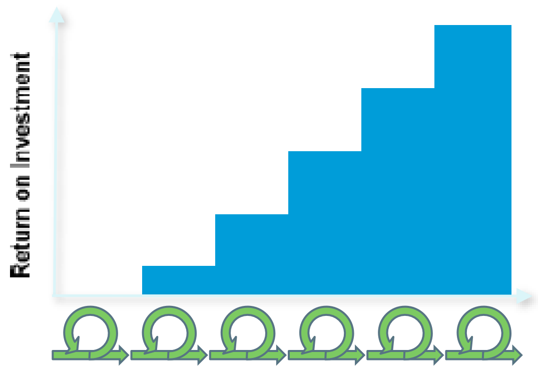
The Value Proposition

The Agile Value Proposition

Greater ROI with Incremental Return

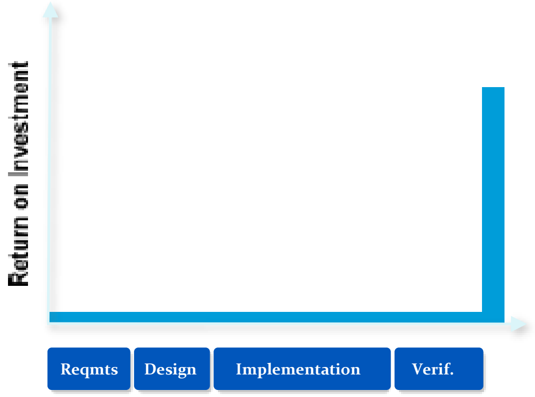
- Agile has a greater ROI by delivering features of greater value through frequent customer feedback rather than less accurate long-term forecasts
- Likewise, features which are of lower value are not built
- Agile has incremental return, which allows value to be realized even if further development is cancelled or put on hold
- See upcoming section on lower cost...

Agile ROI



- In Agile, value is realized incrementally.
- The overall value may be higher because requirements are re-evaluated after each Sprint to deliver the highest priority ones.

Waterfall ROI



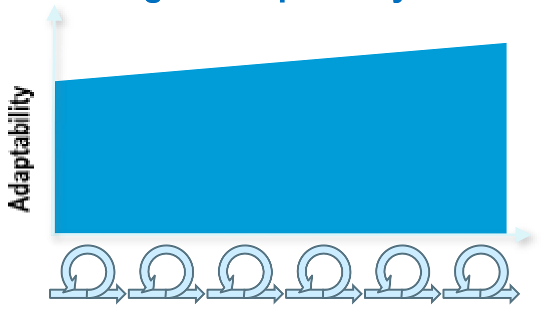
- In Waterfall, value is realized at the completion of the development process.
- The final value may be lower due to requirements "aging."

The Agile Value Proposition

Speed to Market with Greater Adaptability

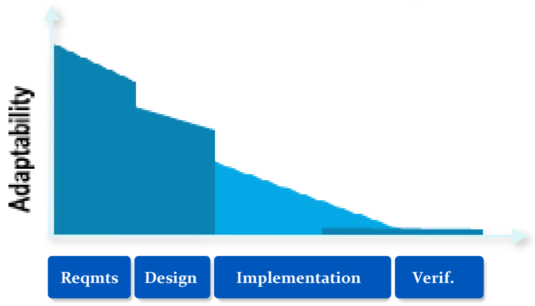
- Agile brings software to market faster with greater adaptability by delivering software incrementally through short Sprints. This gains a competitive advantage, greater responsiveness to threats, and better adaptability to changing market conditions.
- Agile brings software to market faster with greater adaptability by allowing code to be more easily added through XP practices like simple design, refactoring, a system metaphor, coding standards, pair programming, and collective code ownership.
- Agile brings software to market faster with greater adaptability by finding and fixing defects early in the process (when they can be corrected faster) using XP practices like test-driven development, automated testing, and continuous integration.

Agile Adaptability



- Scrum practices are employed to re-evaluate requirements at the beginning of each Sprint.
- XP practices are used to create code which is clear, simple, and adaptable.

Waterfall Adaptability



- With fixed requirements and gated processes, the ability to adapt decreases significantly at the end of each phase.

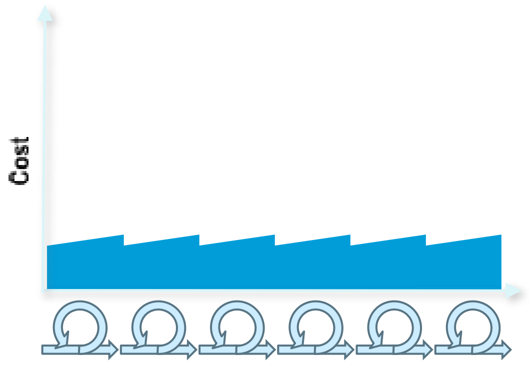
[1] This can be further broken down into "Competitive Advantage" and "Better Responsiveness to Threats" and "Changing Market Conditions."

The Agile Value Proposition

Lower Cost

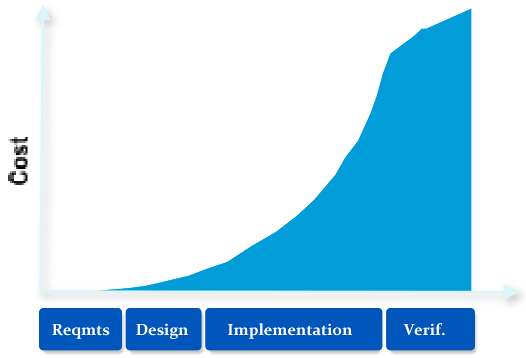
- Agile lowers cost by minimizing the need for heavy documentation, handoffs, and change control processes because teams are aligned by software product rather than functional silo, and collaborate continuously.
- Agile lowers cost through early testing, which detects defects when they are significantly less expensive to fix.
- Agile lowers cost by investing in detail Just-In-Time. Features which have a lower likelihood of being built receive little investment beyond tracking and prioritizing at a high level.

Agile Cost of Fixing an Early Defect



- By using short Sprints, also known as "Define-Build-Test" cycles, errors in requirements, designs, and code are detected and fixed early in the process.

Waterfall Cost of Fixing an Early Defect



- Estimates vary, but a common rule of thumb is that as a defect in specs or code moves from one phase to the next, the cost of correcting it increases by a factor of 10.

The Agile Value Proposition

Reduced Risk

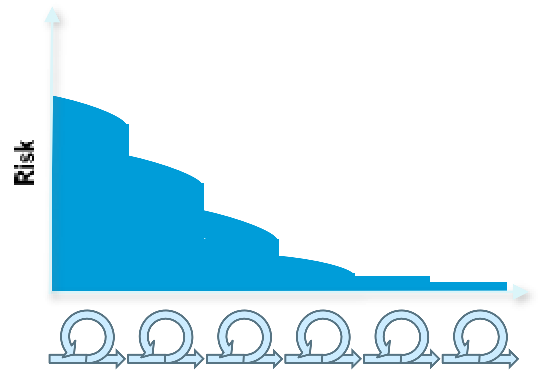
- Agile reduces the risk of building low or no-value features by re-prioritizing them at the beginning of each planning cycle.
- Agile reduces the risk of making a large investment in a product which will ultimately have low value. Delivering faster means "failing faster" to know when to cut your losses.
- Agile reduces the risk of disconnects between what is needed and what is built through frequent demos.
- Agile reduces the risk of missing a delivery date through the typical bottlenecks which result from testing late in the process.
- Agile reduces the risk of building upon incorrect forecasts and assumptions through continuous requirements inspection and validation.
- Agile reduces risk resulting from employee turnover with such XP practices as simple design, a system metaphor, coding standards, pair programming, and collective code ownership.
- Agile reduced risk associated with new technology, unknowns, and assumptions through XP "Spike Solutions."

Continued

The Agile Value Proposition

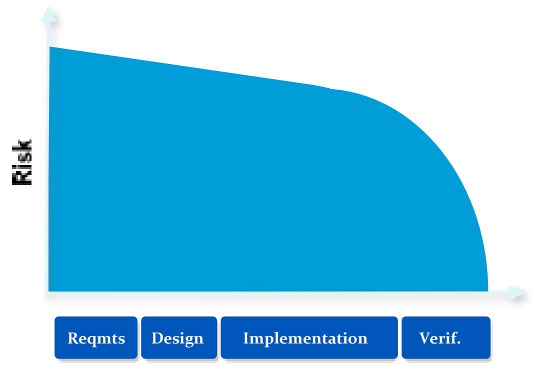
Reduced Risk (continued)

Agile Risk



- Through up-front testing, functional and technical defects are discovered early, lowering risks.

Waterfall Risk



- By freezing requirements early in the process and testing late, a greater number of functional and technical defects are not discovered until the end.

The Agile Value Proposition

Higher Quality

- Agile brings higher quality by incorporating testing throughout the development process to confirm that requirements were properly defined and code was written without defects.
- Agile brings higher quality through continuous integration to detect code integration issues early.
- Agile brings higher quality through additional XP practices such as simple design, refactoring, a system metaphor, coding standards, and pair programming.

Higher Customer Satisfaction

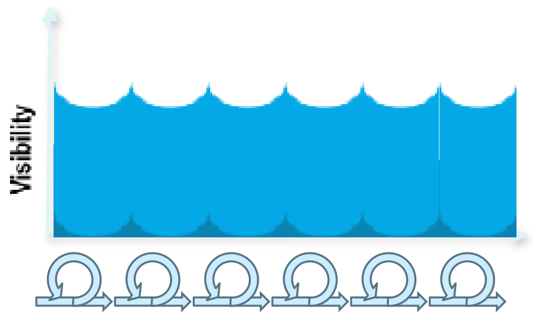
- Agile leads to higher customer satisfaction because customers are able to change their minds as a system moves from the abstract to the real.
- Agile leads to higher customer satisfaction because customers receive more of what they most want but are unable to articulate or predict up front.
- Agile leads to higher customer satisfaction because they feel engaged through frequent demos and releases.

The Agile Value Proposition

Greater Visibility

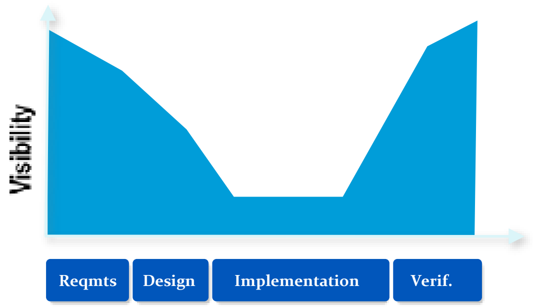
- Agile brings greater visibility through short release cycles and frequent demos of working software. The development process is less of a "black box" to customers and the business.
- Agile brings greater visibility through continuous business involvement. They revisit the big picture frequently, reprioritize requirements during the Release and Sprint planning meetings, and stay in touch with daily progress and issues through the Scrum meetings.
- Agile brings greater visibility of progress and risks through "Information radiators" such as the Scrum Board and Burndown Charts.

Agile Visibility



- Agile has relatively consistent visibility.
- Because of its lower ceremony and self-organizing nature, visibility is not quite as high as the start and end of Waterfall projects.

Waterfall Visibility



- Waterfall projects have high visibility and ceremony up front.
- Visibility drops off as the project enters the implementation "black box." Typically, the business disengages.
- Visibility picks back up once there is working software and perceived risk to the delivery date.

The Agile Value Proposition

Higher Employee Satisfaction

- Employees are more satisfied and motivated when they receive a general plan with reasonable goals and the authority to use their creativity and collective skills to meet those goals.
- Employees are more satisfied when they are able to use their “in the trenches” experience to improve their processes.
- Employees are more satisfied when their own skills are developed to become leaders rather than those positions being filled externally.

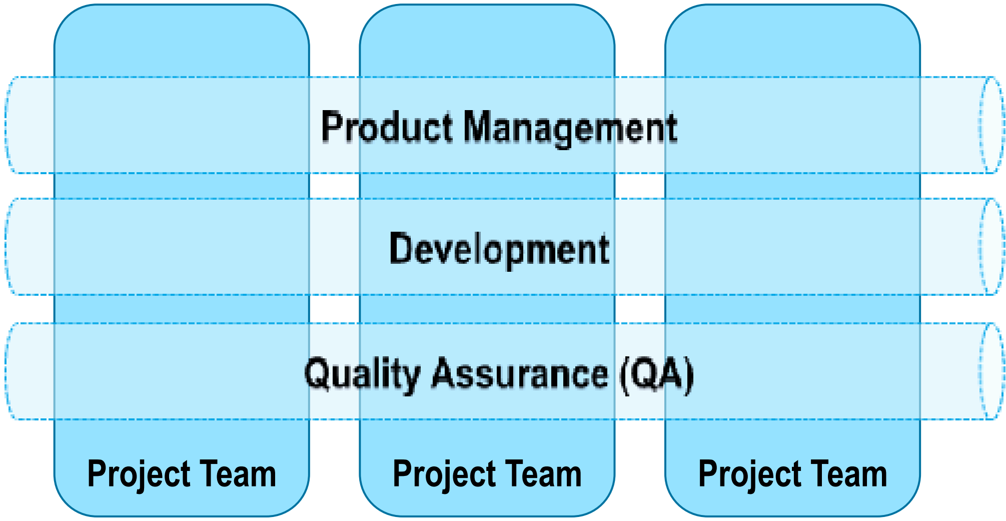
Agile Methodology

Characteristics of an AgileTeam

The Agile Team is a small, cross functional team that is created at the beginning of the project and remains intact till the end.

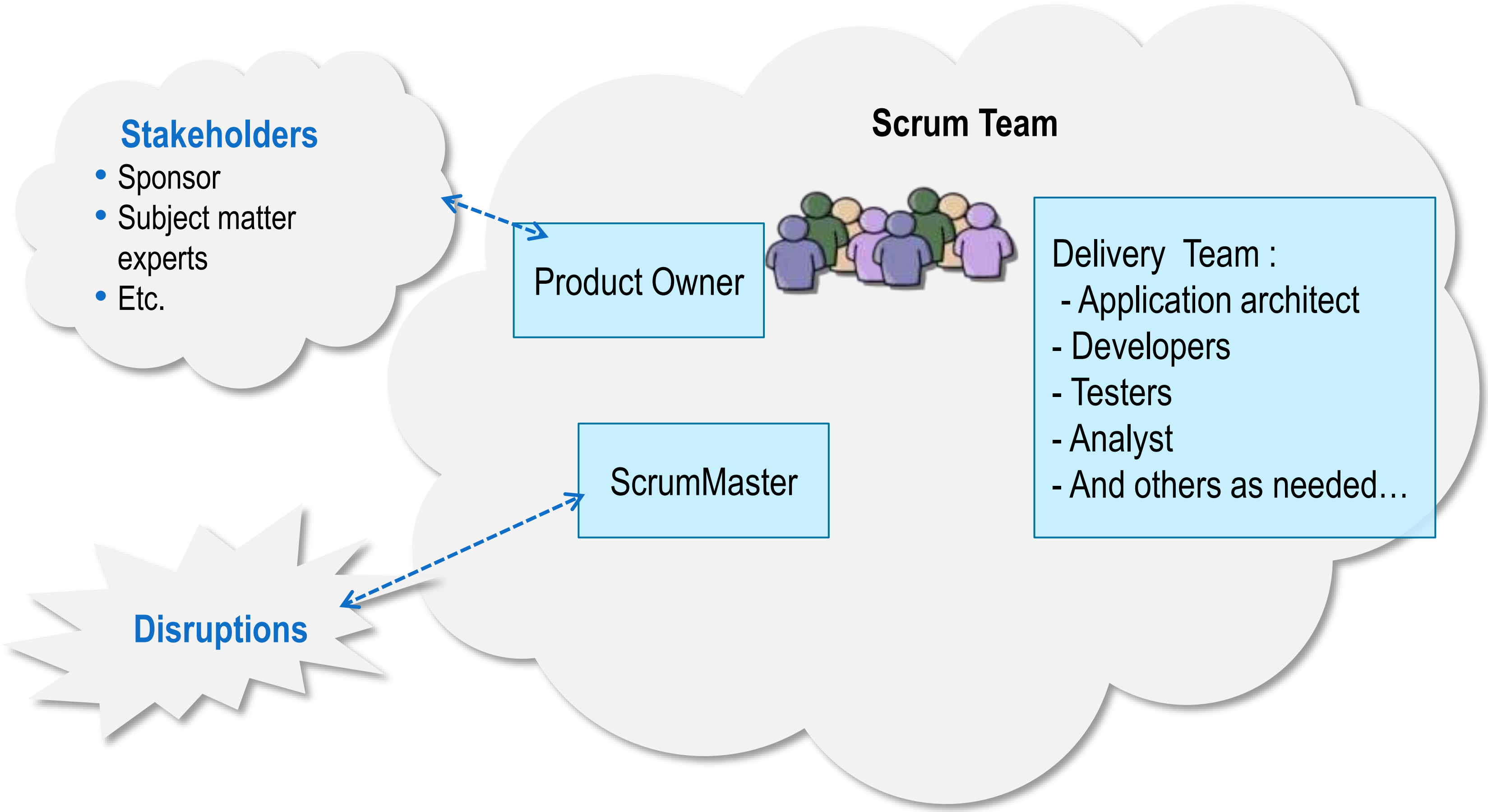
<u>Agile Teams are</u>

- Small (5-9 people)
- 100% dedicated
- Cross-functional
- Cross-trained ("T-shaped" skills)
- Self-empowered
- Motivated
- Collaborative
- Cohesive
- Collocated
- Consistent
- Rewarded as a team



The Agile Team

There are 7 to 9 people full time on a team. They include everyone required to take a product from start to finish.



The User Story

The User Story is the basic building block of the system. It is the initial requirement specification.

As a *<user>*

I can *<do something>*

so that *<user value received>*

The **User Story** specifies ***who***, ***what***, and ***why*** from the **user's** perspective.

Because it **does not** tell how, it opens a conversation with the Delivery Team about how the functionality can best be implemented, leveraging the team's collective expertise.

Epics

Epics are large stories, stories that are too big to be implemented in one Sprint. Epics define large blocks of functionality. Epics are generally defined before Stories.

Epic Description:	____________
User(s):	____________________
Value to NAVTEQ:	______________
Business Value Points:	_________

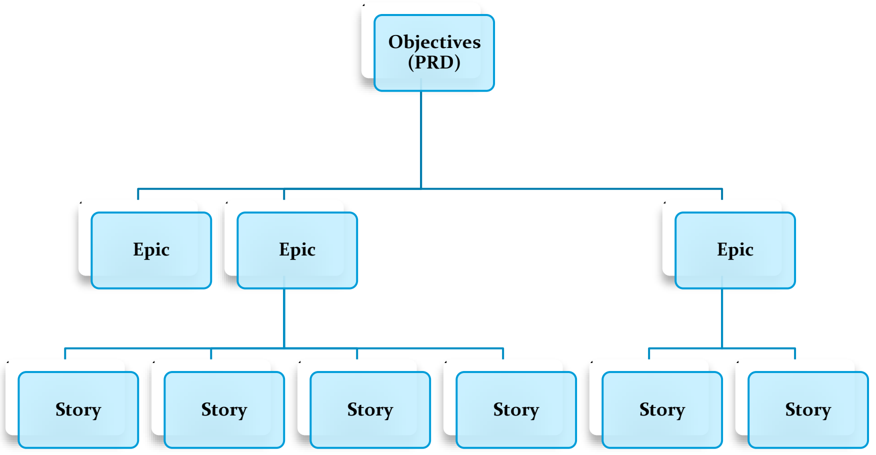
The **Epic** specifies ***what*** (Description), ***for whom*** (Users), and ***why*** (Value) and can be tied directly to business ***value*** (Business Value Points).

Tips

- If you require a reference application or data visualizer to present or validate data, don't forget to include it as an Epic.

Requirements Breakdown Structure

The Objective is broken down Epics. Epics are broken down into Stories.



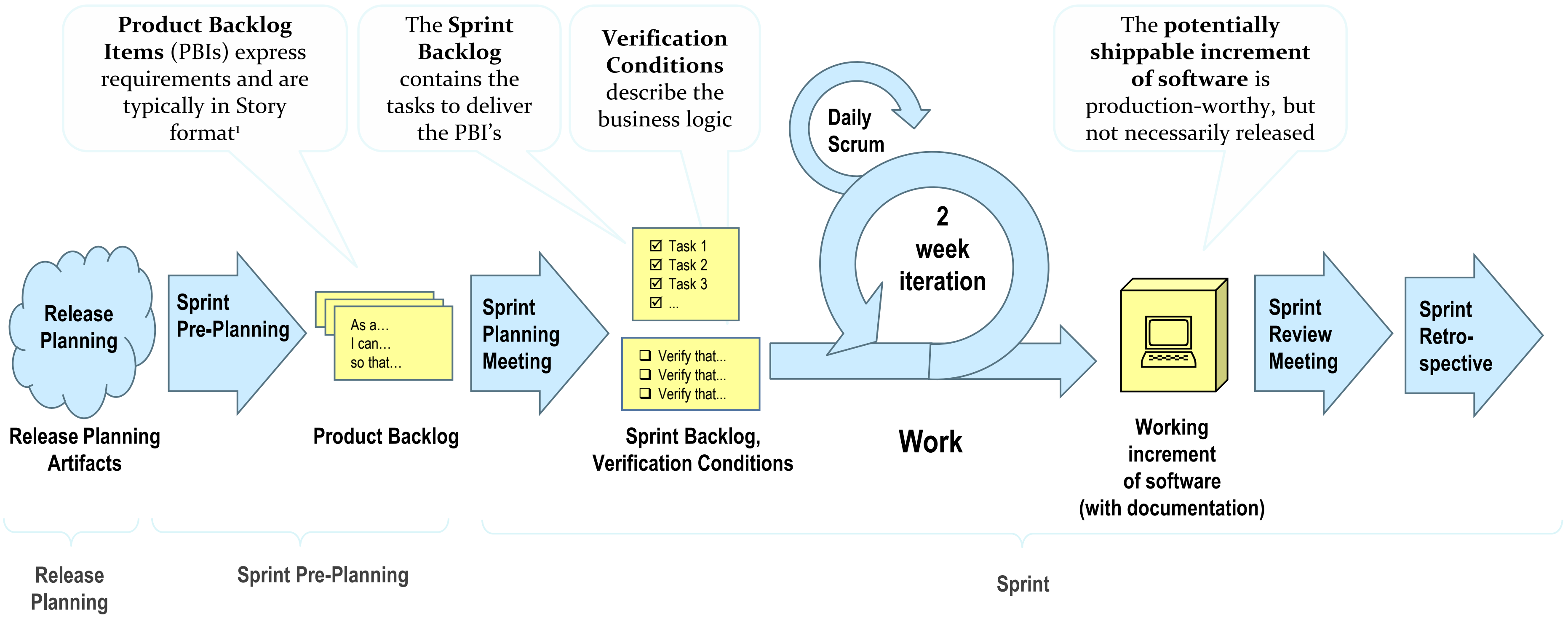
- Epics and Stories are organized in backlogs.
- Backlogs are prioritized lists.
- Epics and Stories are implemented according to their priority.

Rules of Thumb:

- Epics should not span releases. If an Epic is too big to fit in one release, it should be broken down and offer clear business value.
- **Remember: if the project were to stop at the end of any release, all Epics developed should be "Done" and shippable**.
- A Story should take no more than half a Sprint to complete. Otherwise, it should be broken down further into smaller Stories.

The Sprint Cycle

The Sprint cycle is the fundamental time management block of Agile.



Product Backlog Items (PBIs) express requirements and are typically in Story format[1]
The **Sprint Backlog** contains the tasks to deliver the PBI's
Verification Conditions describe the business logic
The **potentially shippable increment of software** is production-worthy, but not necessarily released
Release Planning
Sprint Pre-Planning
As a... I can... so that...
Sprint Planning Meeting
☑ Task 1
☑ Task 2
☑ Task 3
☑ ...
❑ Verify that...
❑ Verify that...
❑ Verify that...
Daily Scrum
2 week iteration
Sprint Review Meeting
Sprint Retro-spective
Release Planning Artifacts
Product Backlog
Sprint Backlog, Verification Conditions
Work
Working increment of software (with documentation)
Release Planning
Sprint Pre-Planning
Sprint

[1] *The Story format originated with Extreme Programming and has gained popularity with Scrum as the expression of Product Backlog Items (PBIs)*