Practical Agile Requirements Engineering

Presented to the
13th Annual Systems Engineering Conference
10/25/2010 – 10/28/2010
Hyatt Regency Mission Bay, San Diego, CA

Presented by:

Dick Carlson (richard.carlson2@boeing.com)

Philip Matuzic (philip.j.matuzic@boeing.com)

Agenda

- Introduction
- Classic Requirements engineering challenges
  - How Agile techniques address these challenges
- Overview / background of Agile software practices
- History and evolution of Agile software Requirements Engineering
- Work products and work flow of Agile Requirements Engineering
- Integration of Agile software Requirements Engineering in teams using Scrum
- Current status of the work and next steps

Introduction

- A large, software-centric program applied Agile techniques to requirements definition using the Scrum approach
- This presentation shows how systems engineering effectively applies Agile practices to software requirements definition and management
- An experience model created from the program illustrates how failures on a large software program evolved into significant process improvements by applying specific Agile practices and principles to practical requirements engineering

Requirements Reality

"The hardest single part of building a software system is deciding precisely what to build.

No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, machines, and other software systems.

No other part of the work so cripples the resulting system if done wrong.

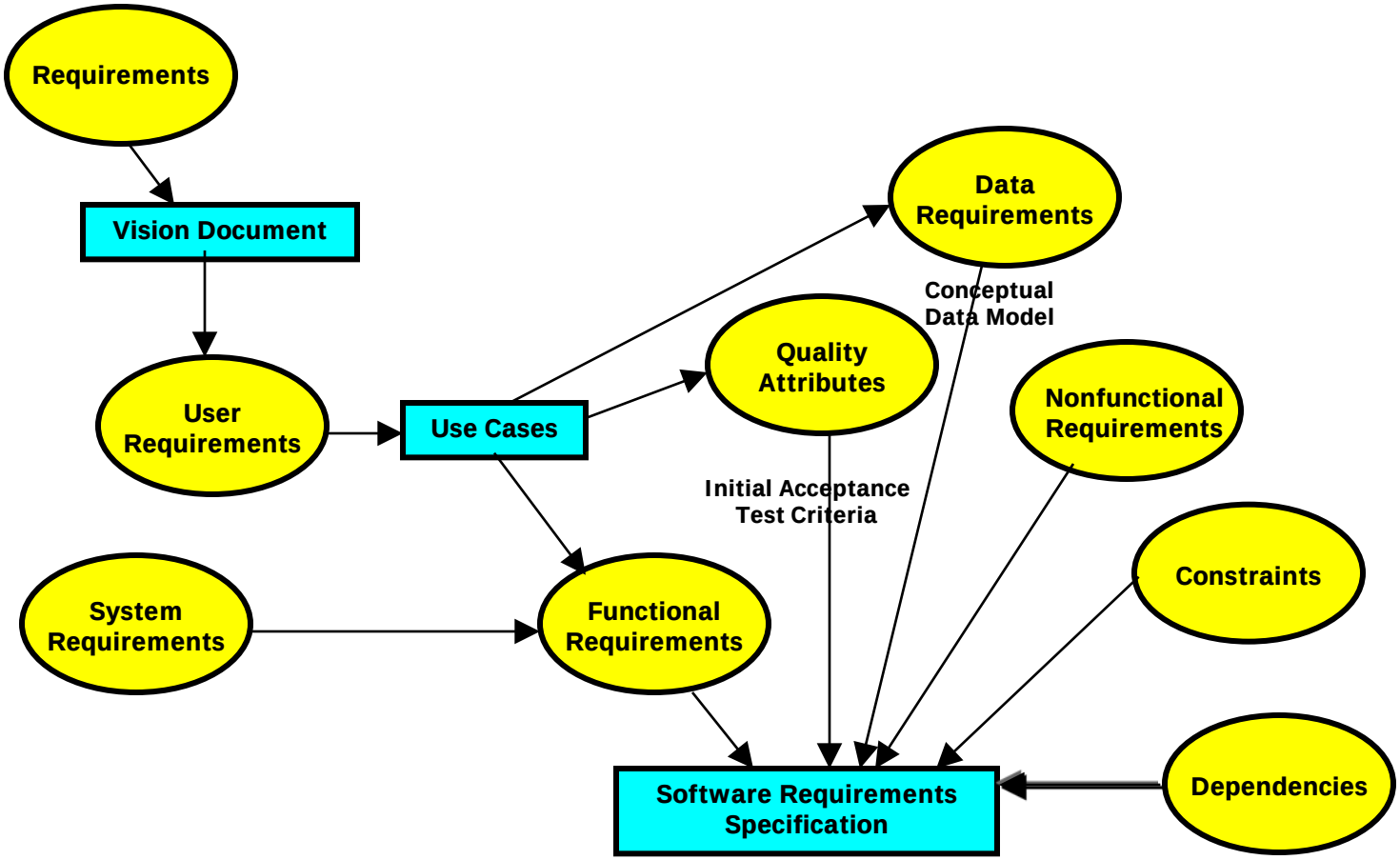
No other part is more difficult to rectify later."

-Fredrick P. Brooks (1986),
"No Silver Bullet – Essence and Accident in Software Engineering"
Proceedings of the IFIP Tenth World Computing Conference: 1069-1076

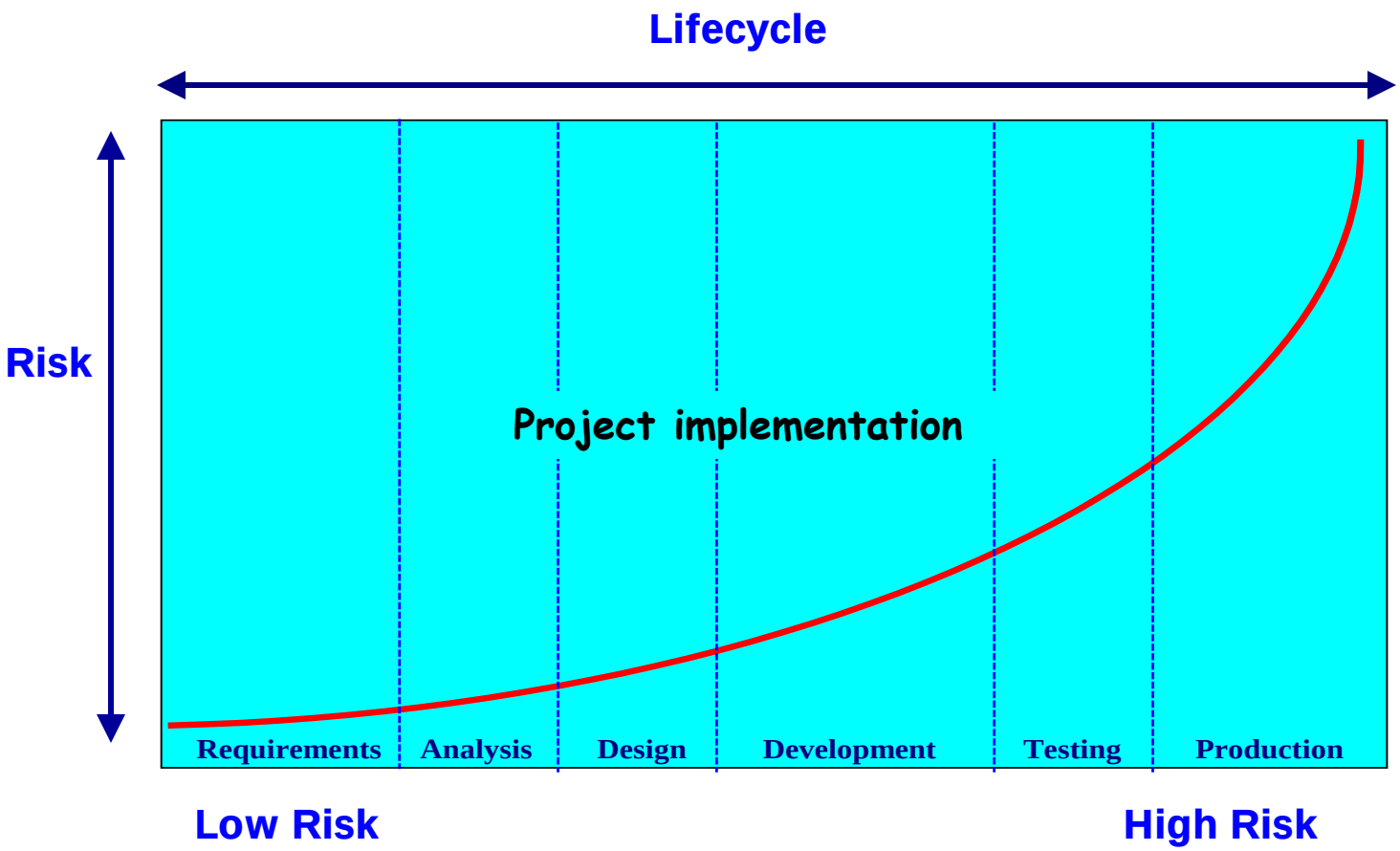
Traditional Requirements Problem

- **Requirements development is a communication problem**
- **Stakeholders and the development team typically do not have a clear understanding of the requirements as written**
- **Validation happens late in the life cycle**
- **Users see the product late in the process**

Traditional Requirements Engineering



Requirements Risk Abatement



Agile Principles and Practices

Principles:

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

Practices:

- **Close customer collaboration**
- **Daily stand-up meetings**
- **Planning and estimating**
- **Frequent feedback**
- **Short, daily standup meetings**
- **Short iterations**
- **Prioritized requirements**
- **Artifact reviews**
- **Self-organized teams**
- **Retrospectives**

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

Agile Requirements Development

- **Communications problem**
- **Requirements are developed and written as user stories (sometimes supported by Agile use cases)**
- **User stories shift the focus from writing to talking**
  - Written in a common language
  - Support iterative development
  - Brief description of functionality as viewed by the user
  - Written to convey functional requirements
  - Support participatory design
  - Emphasize user goals
  - Focus on the “what” – not the “how”

Requirements vs. User Stories

- **Traditional requirement "shall" statements**
  - "The system shall provide a user configurable interface for all user and system manager functions"
  - "The user interface shall be configurable in the areas of:
    - Screen layout
    - Font
    - Background and text color"

- **Corresponding User Story**
  - "As a system user or system manager,
  - I want be able to configure the user interface for screen layout, font, background color, and text color,
  - So that I can use the system in the most efficient manner"

Agile Systems Engineering

Agile SE Defined

- Definition: Agile Systems Engineering (SE) is 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 Product Backlog is prioritized by the Product Owner, and made available to the development team

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 (or working sessions)

- A Scrum project is a series of 4 week iterations where fully developed requirements, functional analyses, and system-level architecture incorporating system decisions are demonstrated at each iteration completion

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

More Scrum Practices

- 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

Systems Engineering Activities

- **Systems engineering includes consistent processes that produce specific and supportive artifacts —**
  - **To ensure integration with other engineering disciplines and domains**
  - **To ensure 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 operations and disposal**

Key Focus of Systems Engineering

- **Two key thrusts in program-level systems engineering domains:**
  1. **Engineering the system to define the technical solution, and**
  2. **Planning and control supporting program management**

- **Engineering the system requires:**
  1. **Requirements Analysis / Definition / Validation**
  2. **Functional Analysis and Allocation**
  3. **Design Synthesis**
  4. **Evaluation of Alternatives**
  5. **Requirements Verification**

Key SE Planning & Control

- **Planning and control requires:**
  - **Organizing and Planning (e.g. Organizing the program, and development of the SEMP, IMP/IMS)**
  - **Requirements Management**
  - **Interface Management**
  - **Baseline Definition and Management**
  - **Affordability**
  - **Decision Making (e.g. Risk Management, Trade Studies, TPMs)**
  - **Metrics Management**
  - **Reviews**

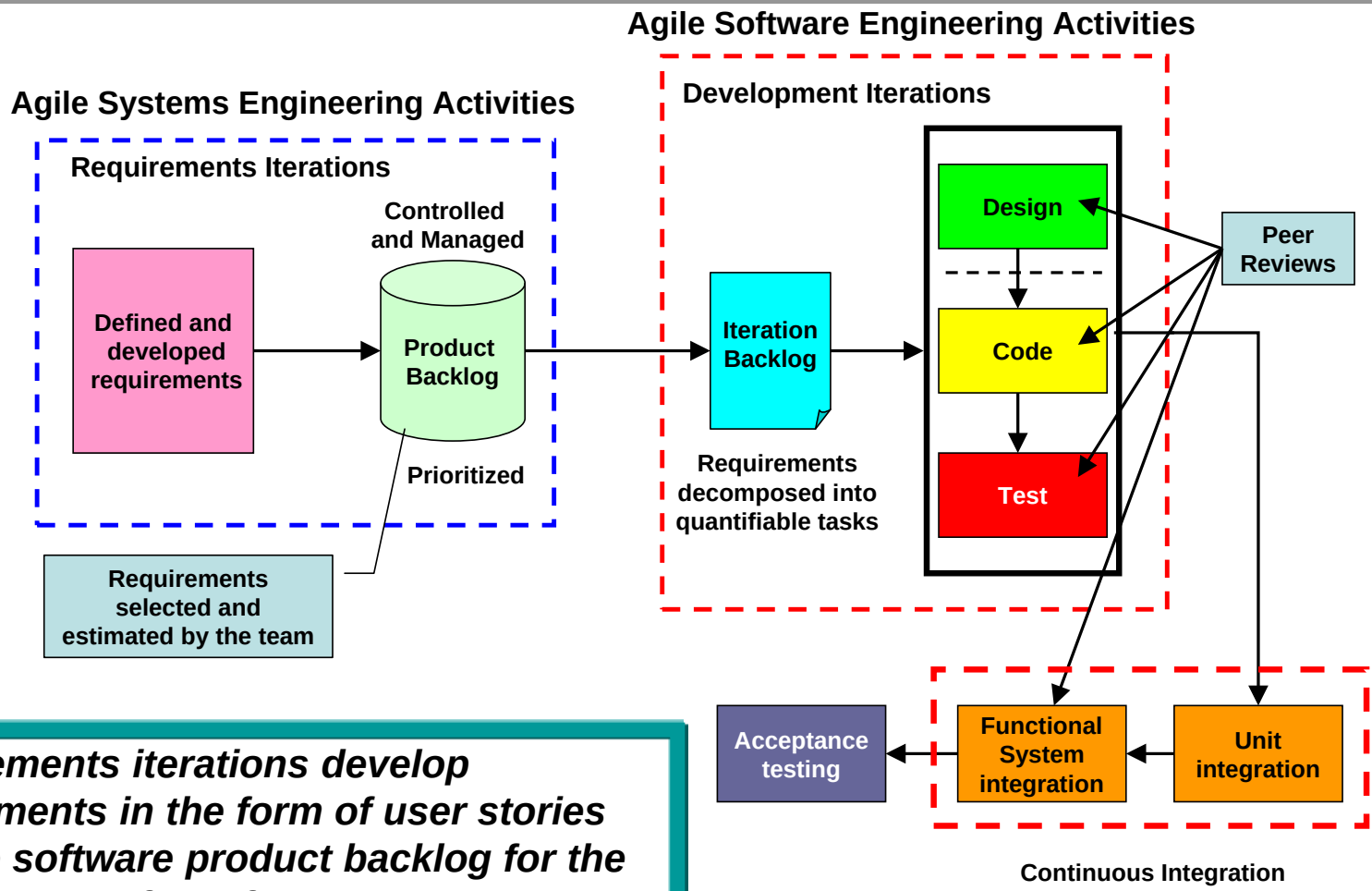
Bridging Agile Practices to SE

- **The benefits experienced by applying common Agile practices apply well to systems engineering activities.**
- **These Agile practices include:**
  - **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**

Agile SE for Integrated Engineering

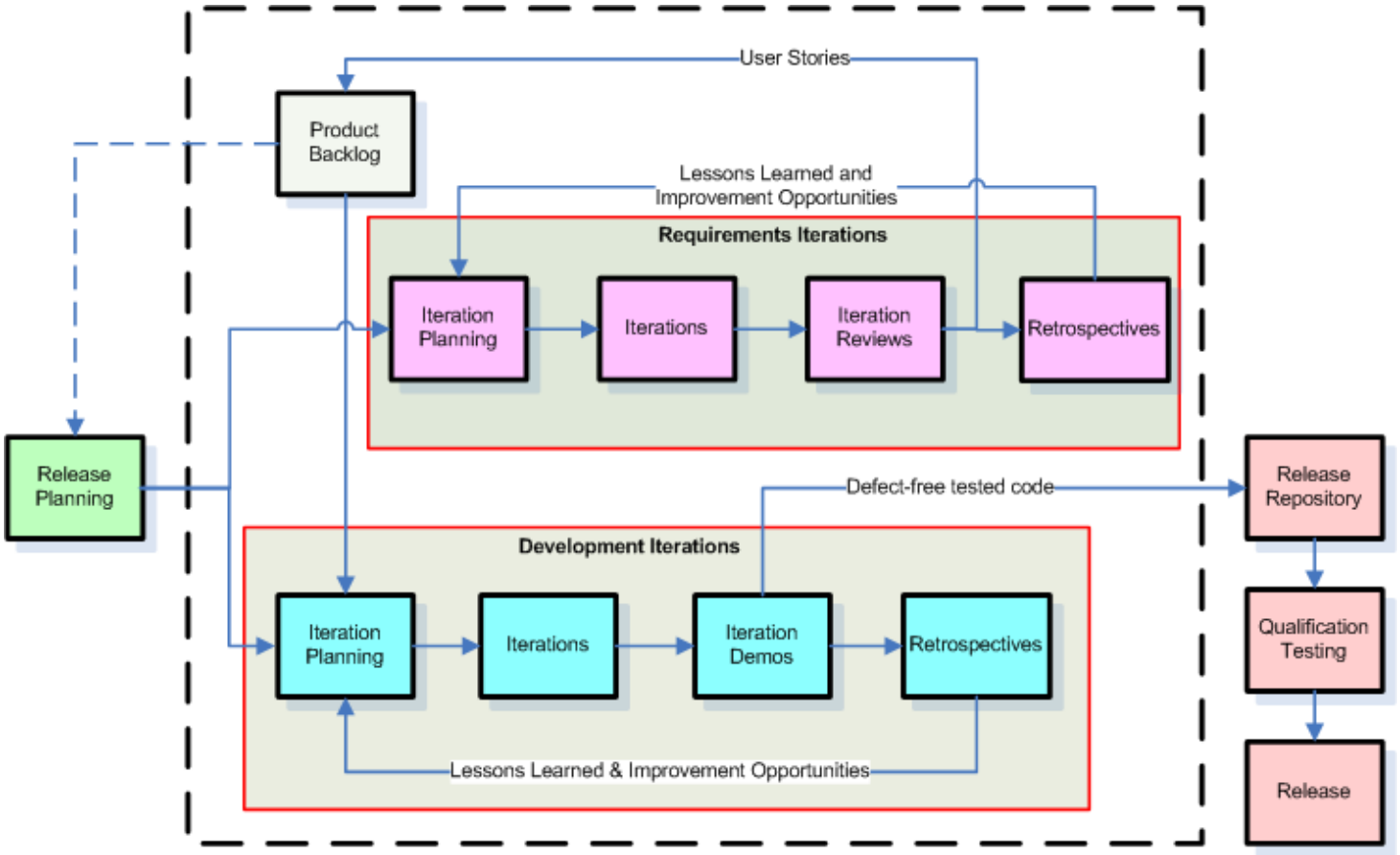
- **The Agile systems engineering approach leverages integrated engineering by employing the same vocabulary and artifacts in an evolving, iterative approach**
  - **Software and systems engineers work collaboratively, consider alternatives, and take actions that lead to process change and improvement**

Boeing SE/SW Experience Model



Requirements iterations develop requirements in the form of user stories into the software product backlog for the development of a software system

SE/SW Collaboration Model



Systems and software engineers collaborate on release planning and the development of requirements written as user stories

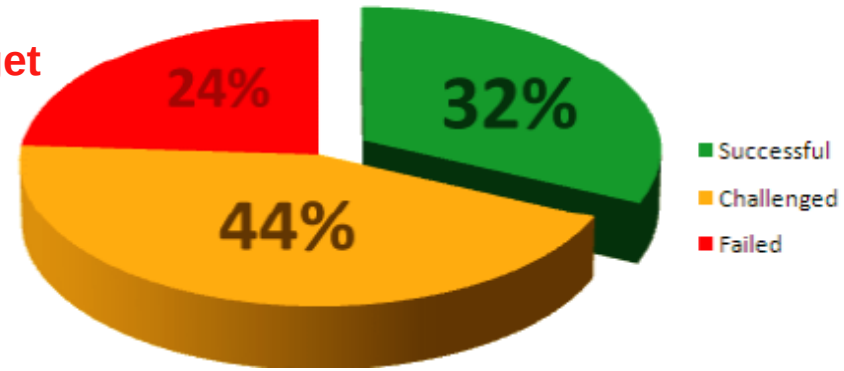
Questions?

Back-up slides

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

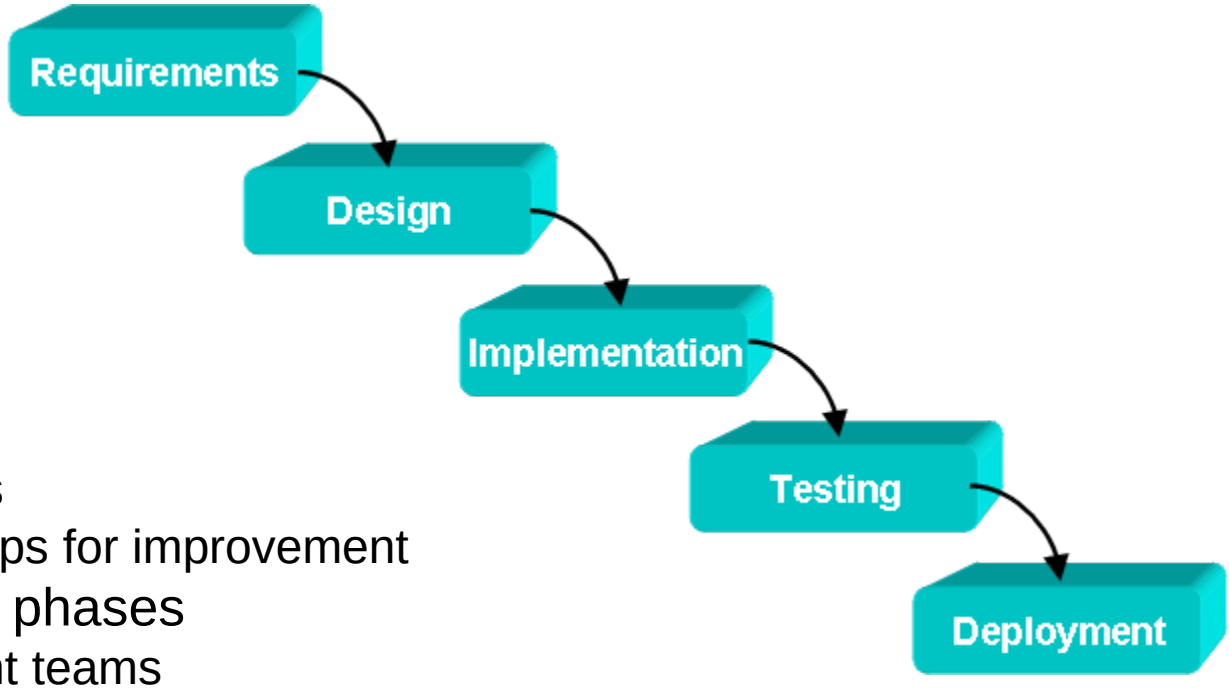
- 2009 CHAOS Standish Chaos Report
  - **32% of software development projects were successful**
  - **44% challenged**
  - **24% failed to meet schedule or budget**



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

Why Aren't the Successes Higher?

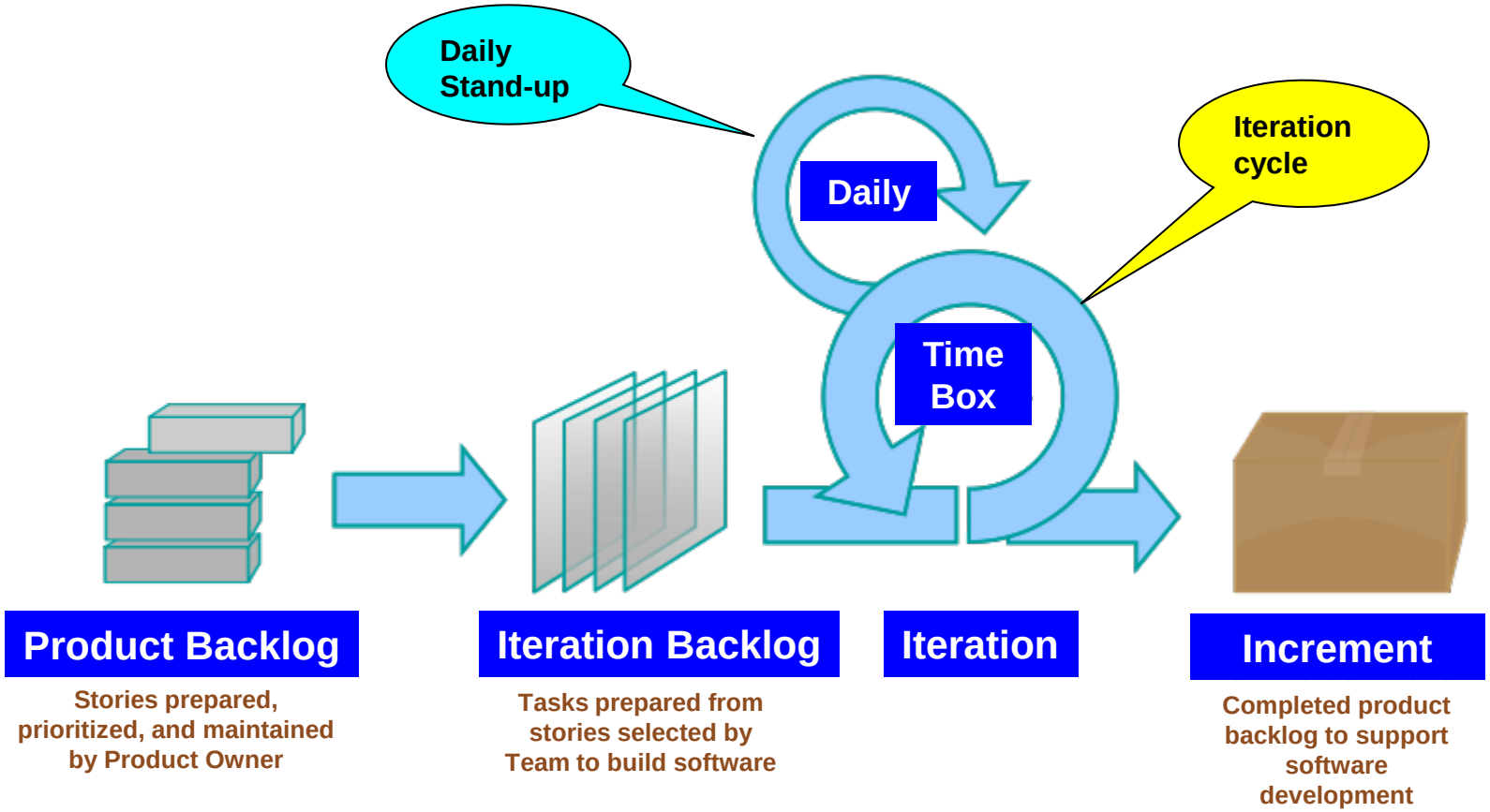
Historic processes were subjected to the weaknesses of the Waterfall model….



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

Scrum Framework



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/)