System Development and Project Organization (BSUP)

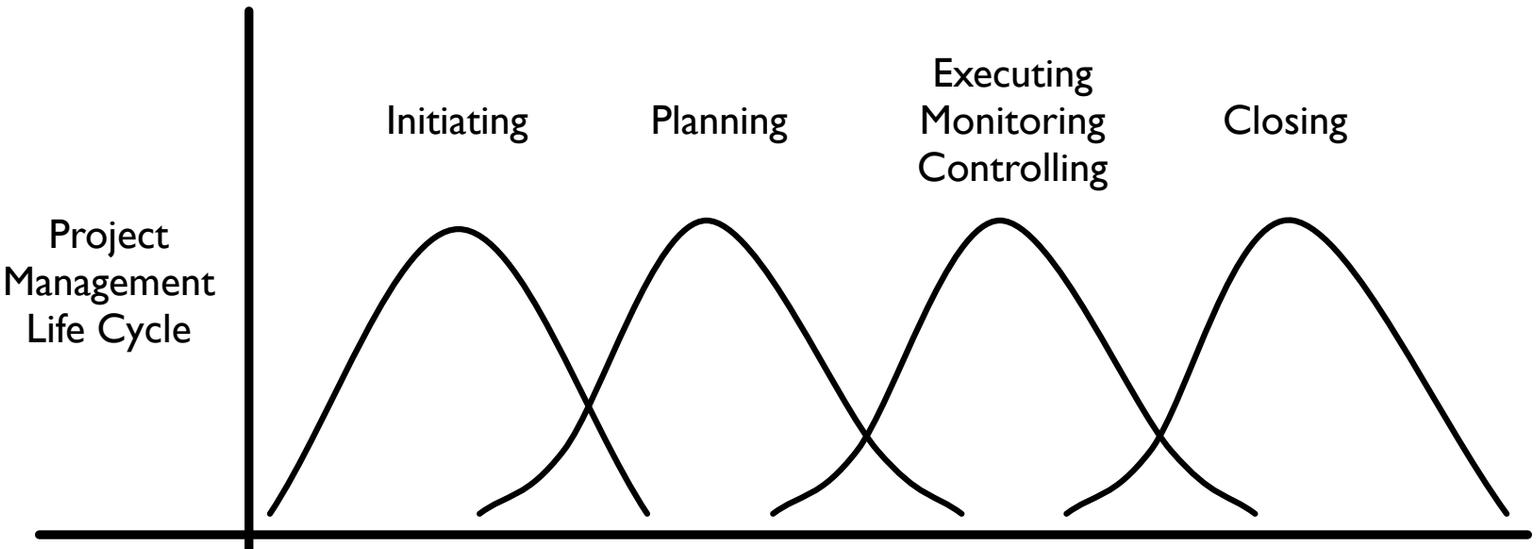
Paolo Tell

Software Processes

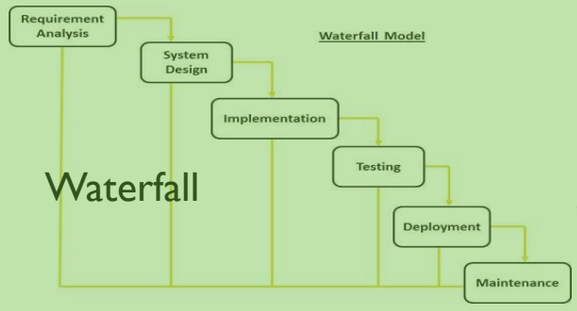
IT University
of Copenhagen

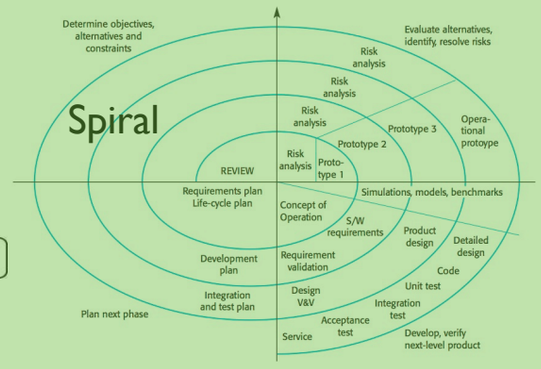
Outline

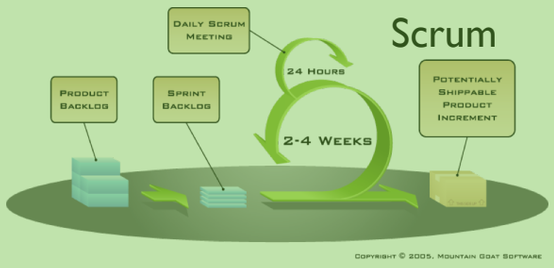
Project Management Life Cycle



Software Development Life Cycle







- Rational Unified Process (RUP)
- eXtreme Programming (XP)
- Feature-Driven Development
- ...

Software Process:
Definition and Basic Models

IT University
of Copenhagen

Motivation

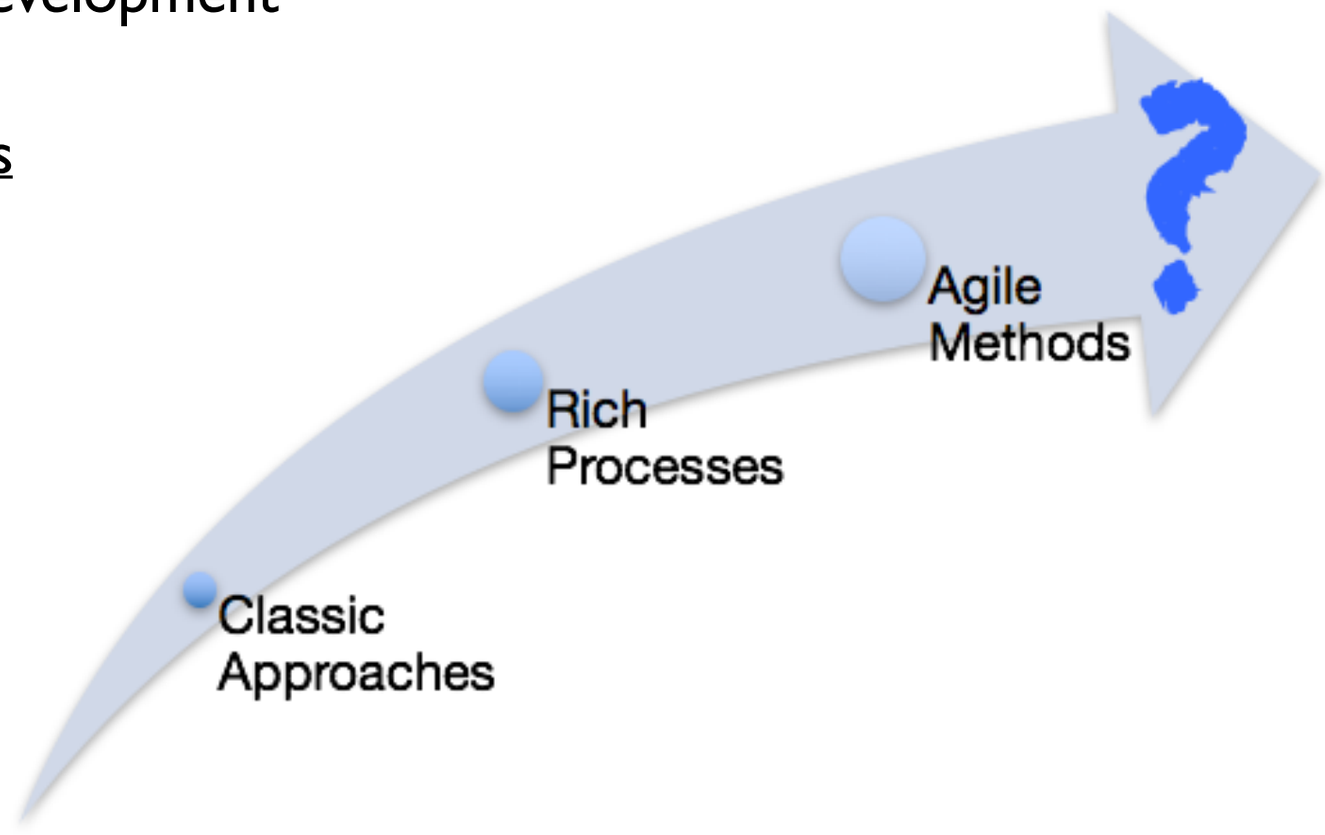
- Software processes exist since the early 1970's

- Goal: "Get out of the software crisis"
  - ➡ Birth of Software Engineering

- Goal: systematization of software development

- Development of Software Processes
  - Inflation
  - "Religious wars"

- State-of-the-art = understanding of interplay between process selection, project success, and project parameters.

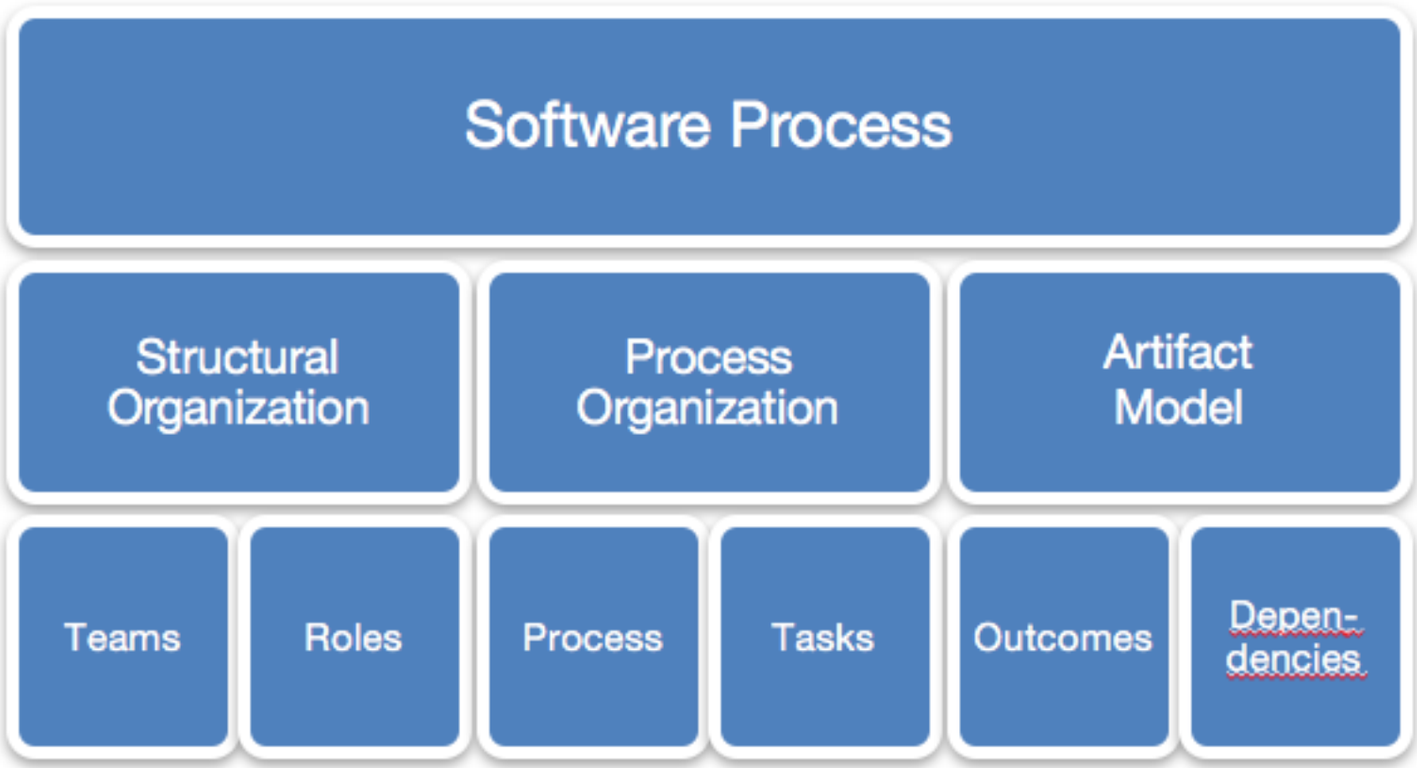


Definition: Software Process

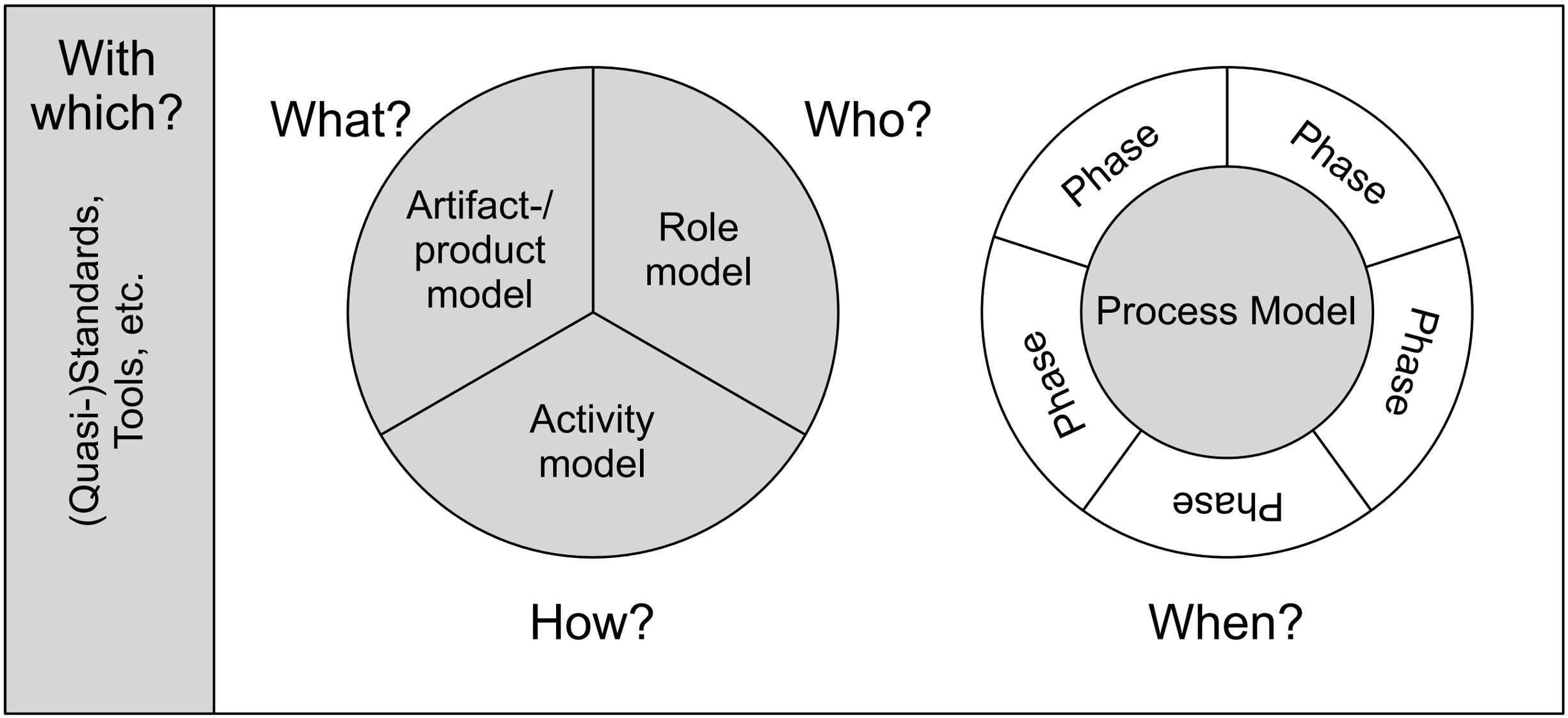
- A software process (model) describes:
  - Systematic,
  - Engineering, and
  - Quantifiable approaches

  to solve task of a particular class in a repeatable manner.

- Software processes address:
  - Structural organization
  - Process organization
  - Artifact models



What is a software process (model)?



Concrete software process

- There are few basic models (on which everything else is based):
  - Phase model
  - Spiral model
  - Prototyping
  - Agile models

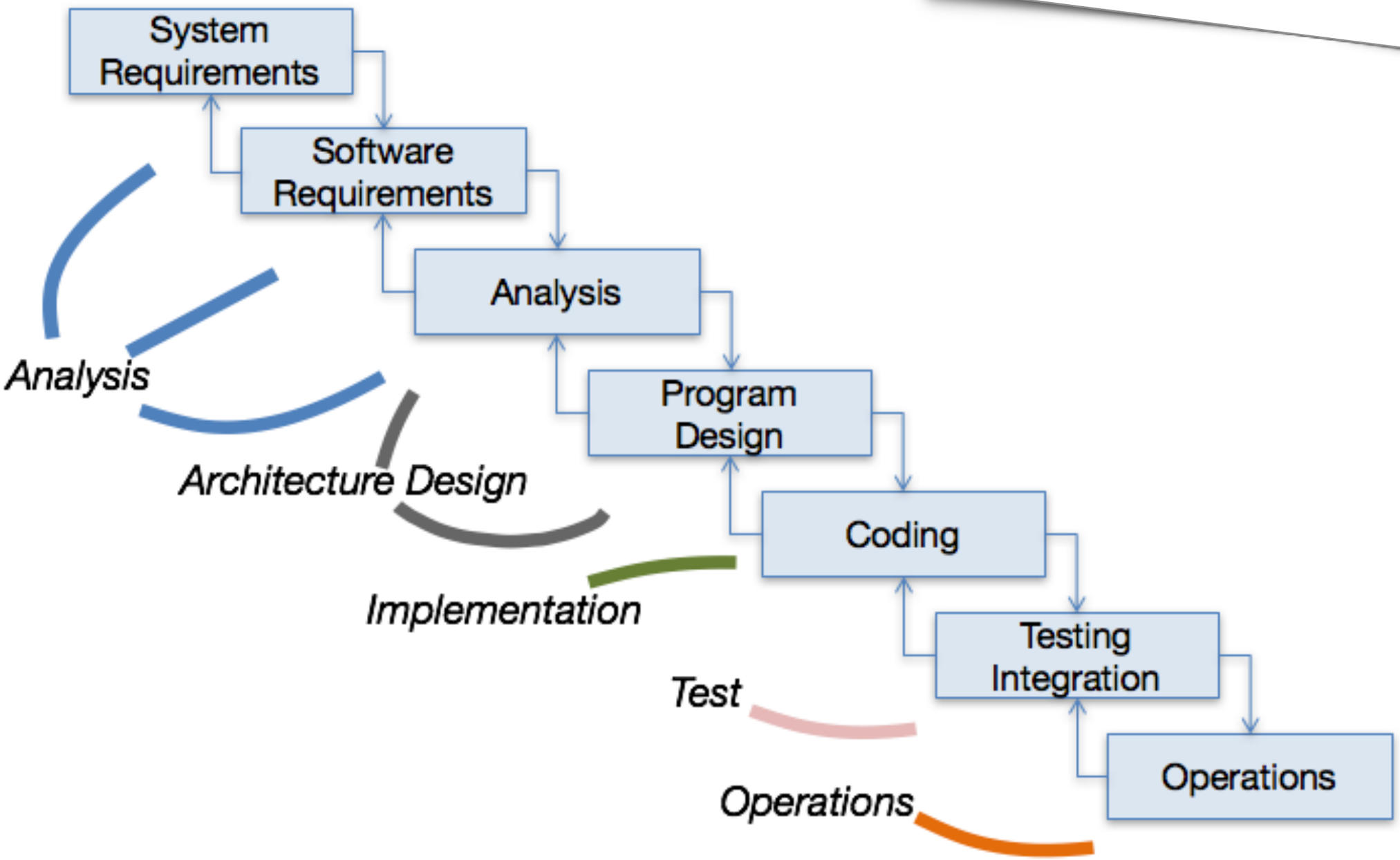
- And there is a plethora of concrete software processes (nobody really knows how many…), e.g.:
  - Rational Unified Process
  - V-Modell (XT)
  - Scrum
  - Kanban
  - eXtreme Programming
  - Feature-driven development
  - Test-driven development
  - Crystal
  - …

Definition and Basic Models

Waterfall Model

Overall Philosophy

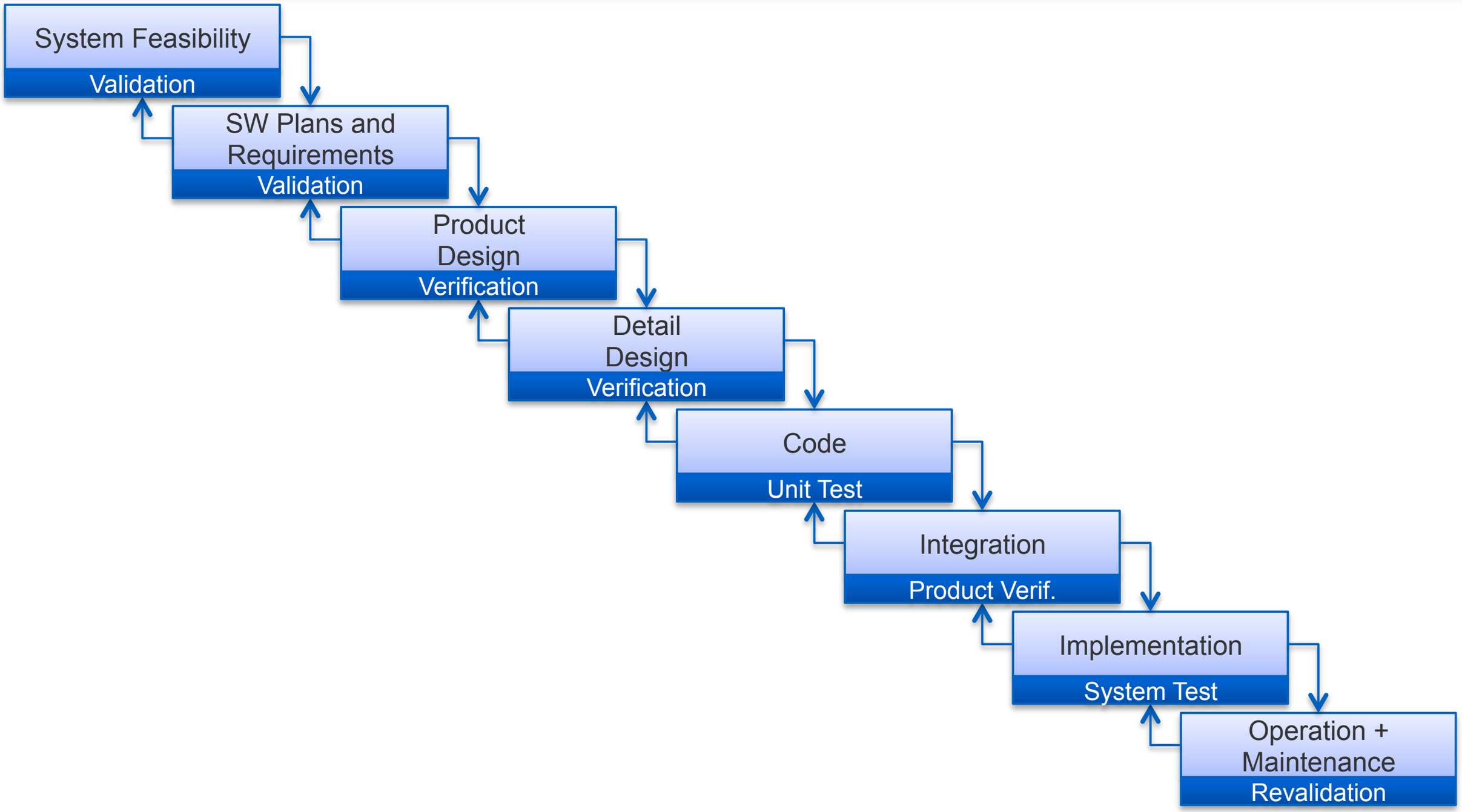
- Structure follows the sequence of the development activities



Waterfall Model

- **"Most famous" implementation of the classic phase model**
  - Aka: conventional approach
  - 1st time mentioned by Royce (1970)
  - Strict sequence
  - Each phase
    - ➡ one milestone on which the next phase is built
    - Bundling activities in the "right" order
    - Expectation: finished results
  - Feedback possible, but only between "neighbored" phases

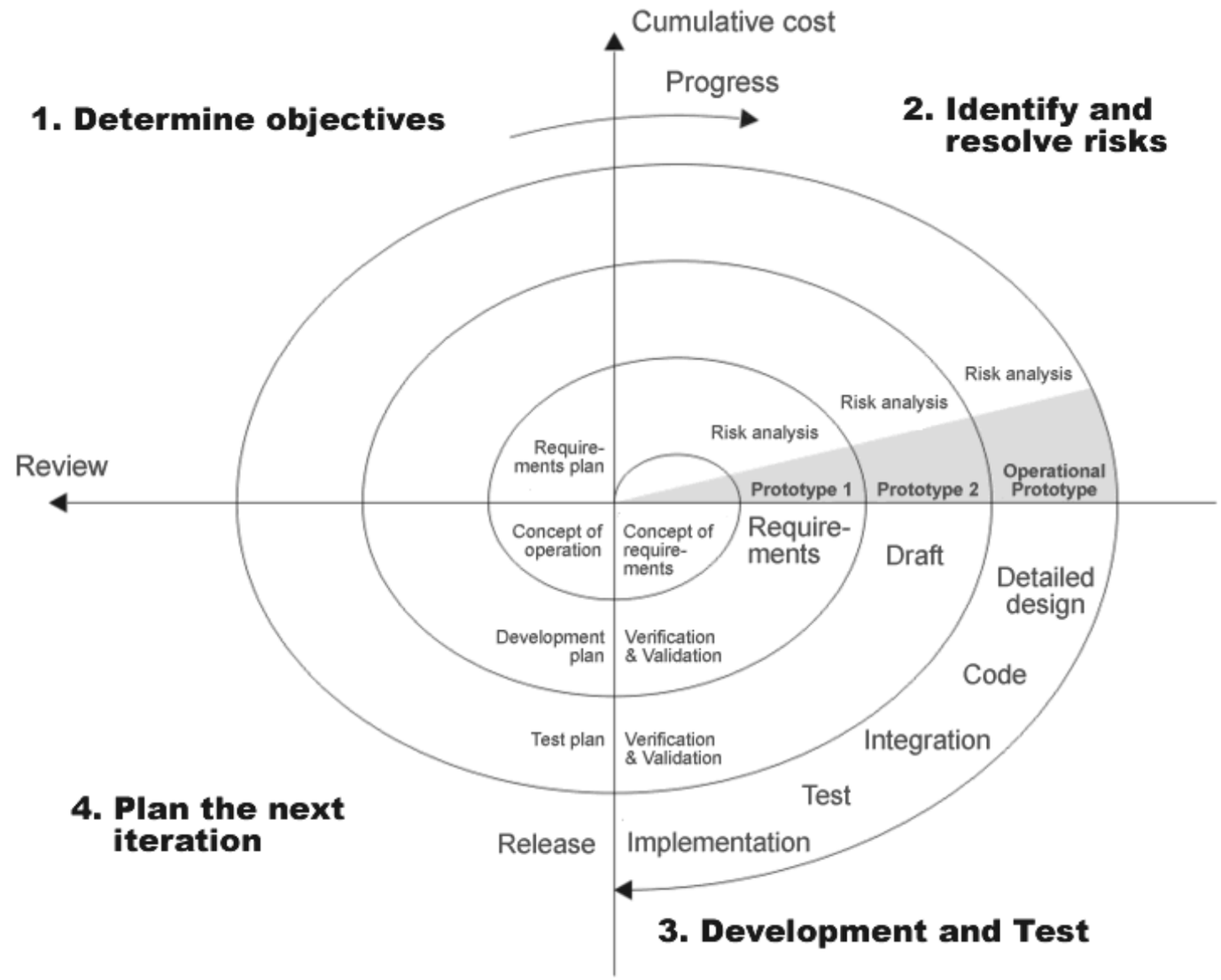
Waterfall Model
Barry's variant: integrate quality assurance



Spiral Model

Overall Philosophy

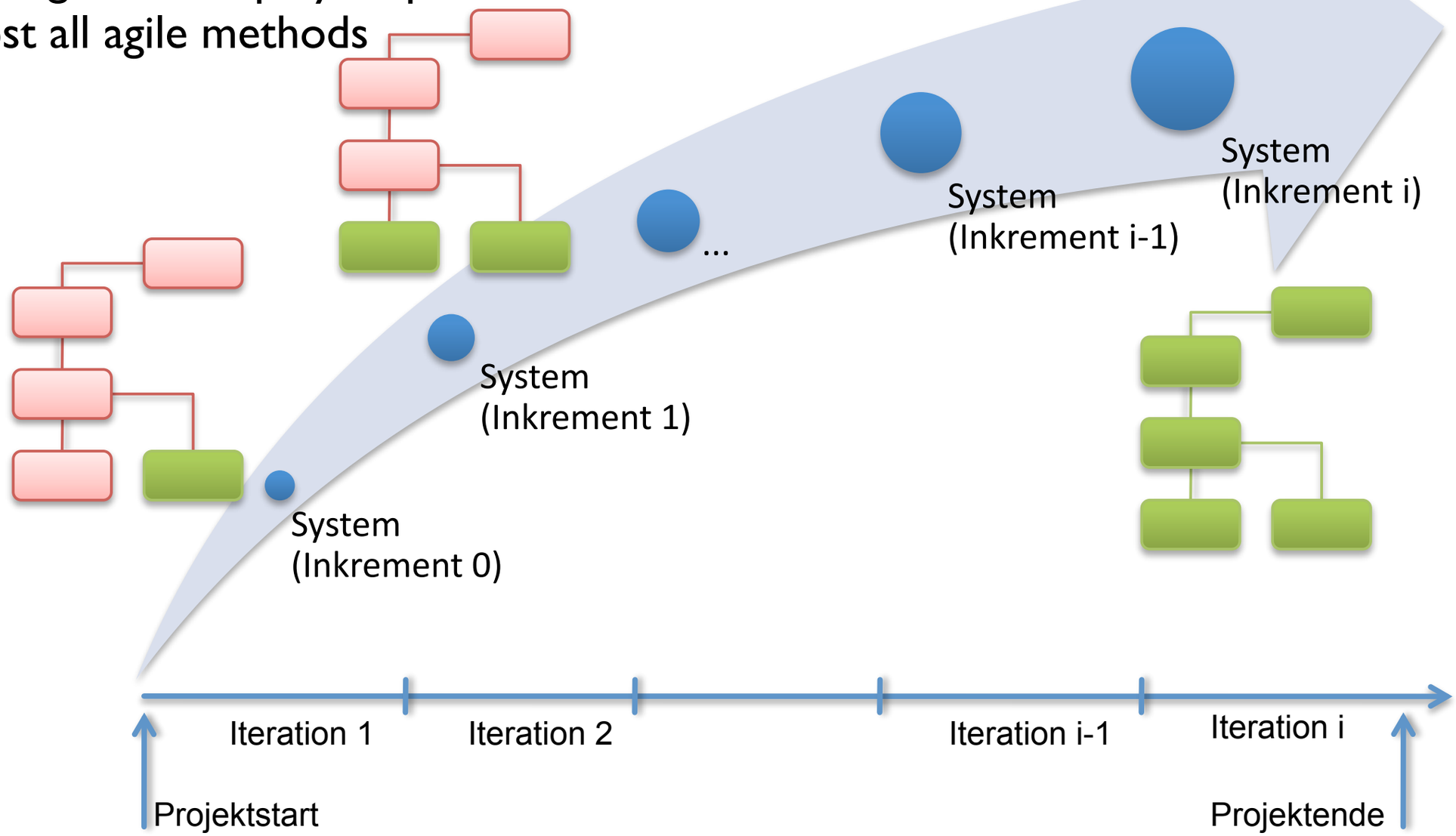
- Repeat the steps
  1. Define goals
  2. Analyze risks
  3. Evaluate
  4. Plan next iteration



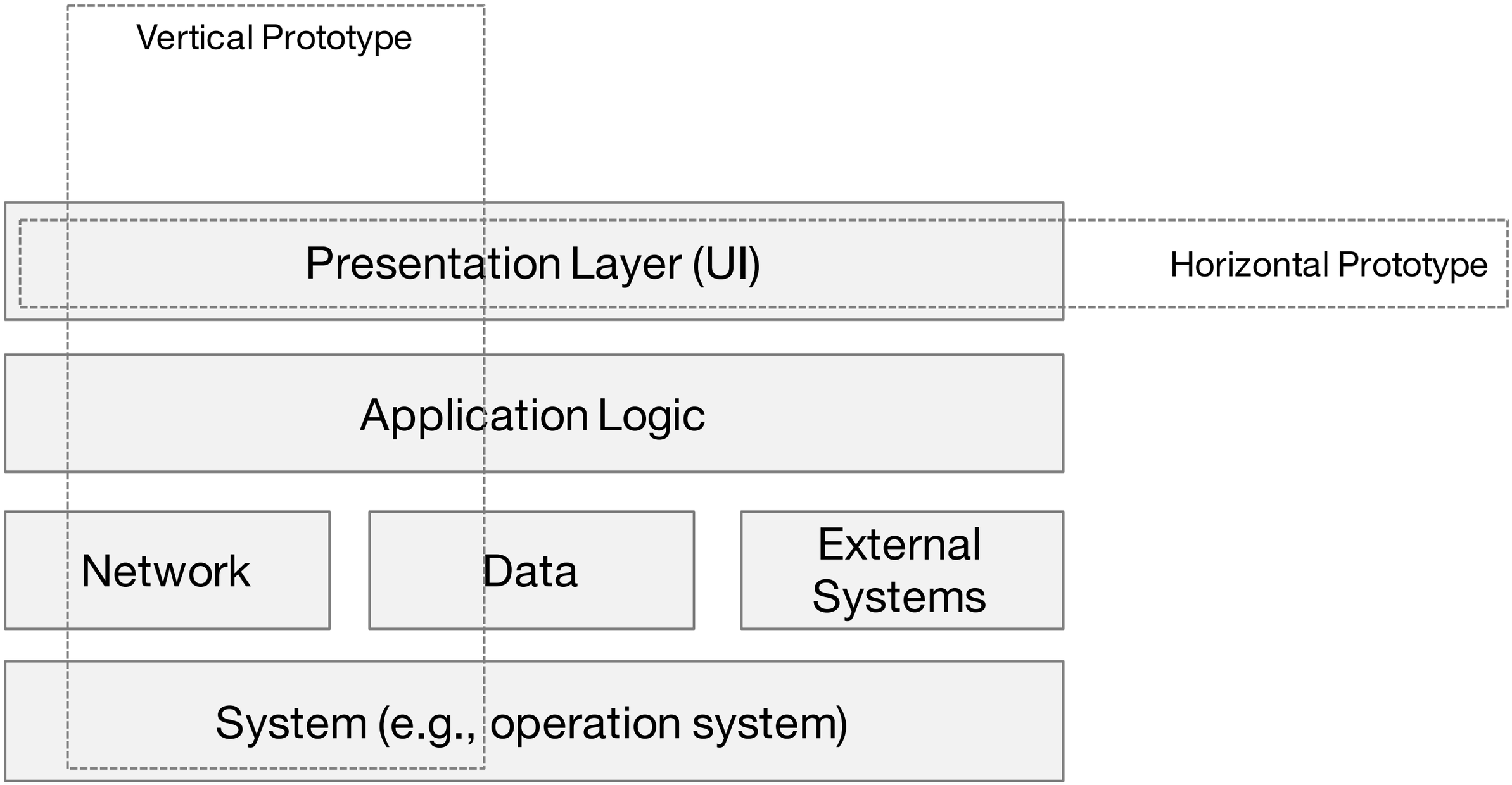
- **The reference of iterative/incremental development**
  - Concept: iterative approach
  - Goal: minimize risk
  - Key: create prototypes for continuous test/evaluation

Incremental/iterative approach

- Based on the spiral model
  - With each iteration, the system scope is extended
  - The system "grows" step by step
- Basis for almost all agile methods



Prototyping



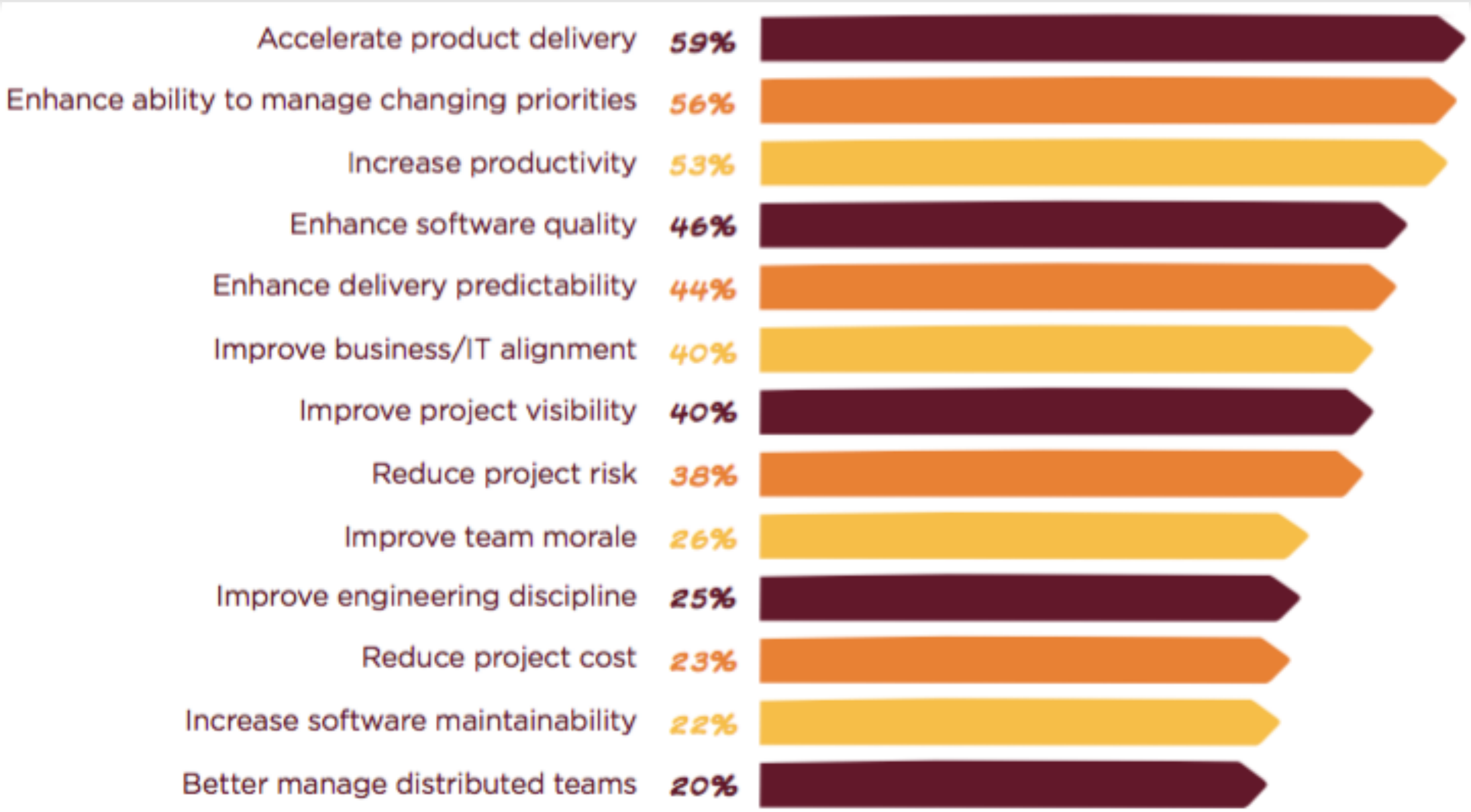
Agile Methods

Agile methodology

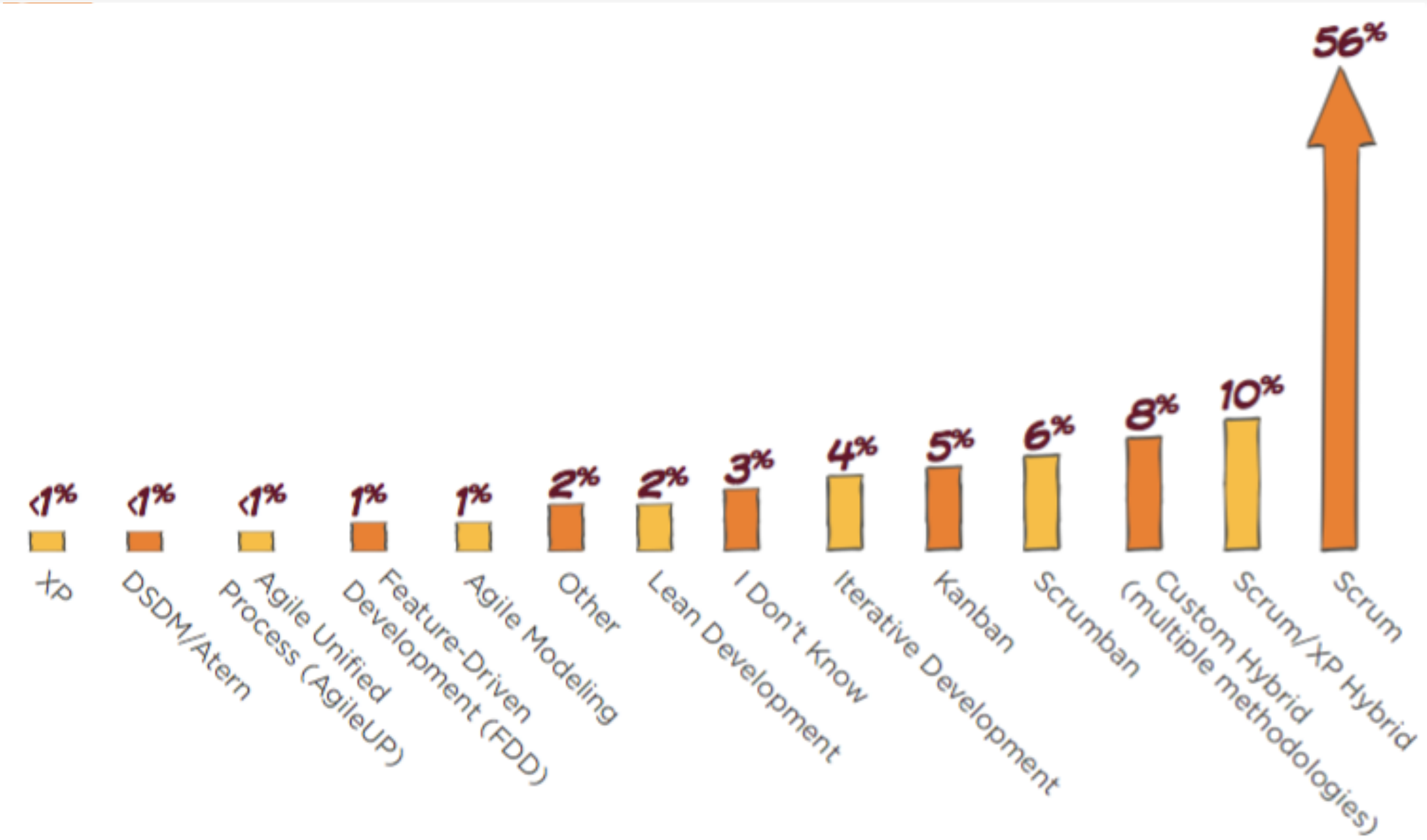
- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These methods:
  - focus on the code rather than the design
  - are based on an iterative approach to software development
  - are intended to deliver working software quickly and evolve this quickly to meet changing requirements.

- The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation, by fostering communication) and to be able to respond quickly to changing requirements without excessive rework.

Reasons for adopting agile [versionone survey 2015]



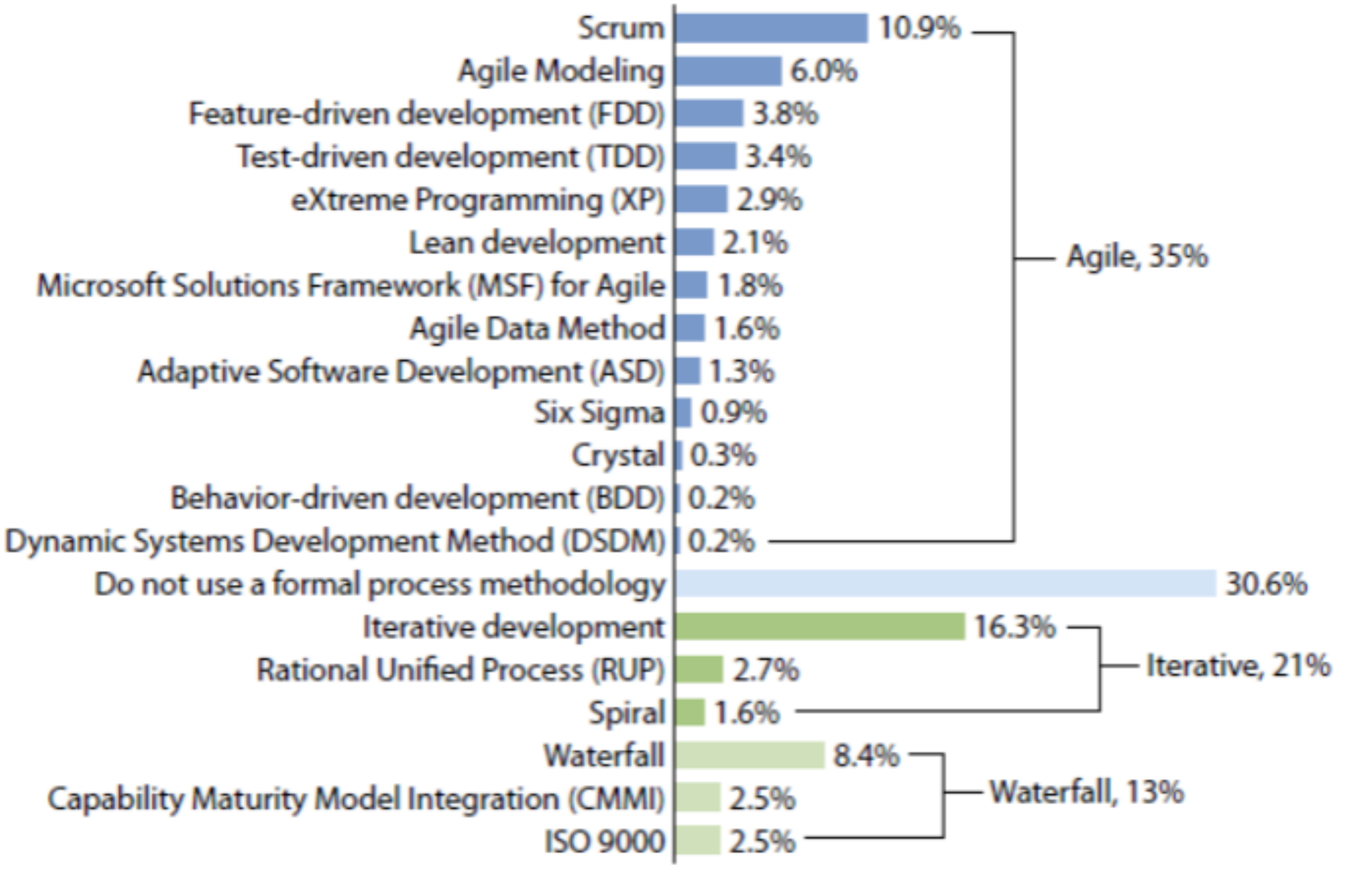
Agile method used [versionone survey 2015]



Agile techniques [versionone survey 2015]

- 80% Daily standup
- 79% Short iterations
- 79% Prioritized backlogs
- 71% Iteration planning
- 69% Retrospectives
- 65% Release planning
- 65% Unit testing
- 56% Team-based estimation
- 53% Iteration reviews
- 53% Taskboard
- 50% Continuous integration
- 48% Dedicated product owner
- 46% Single team (integrated dev & testing)
- 43% Coding standards
- 38% Open work area
- 36% Refactoring
- 34% Test-Driven Development (TDD)
- 31% Kanban
- 29% Story mapping
- 27% Collective code ownership
- 24% Automated acceptance testing
- 24% Continuous deployment
- 21% Pair programming
- 13% Agile games
- 9% Behavior-Driven Development (BDD)

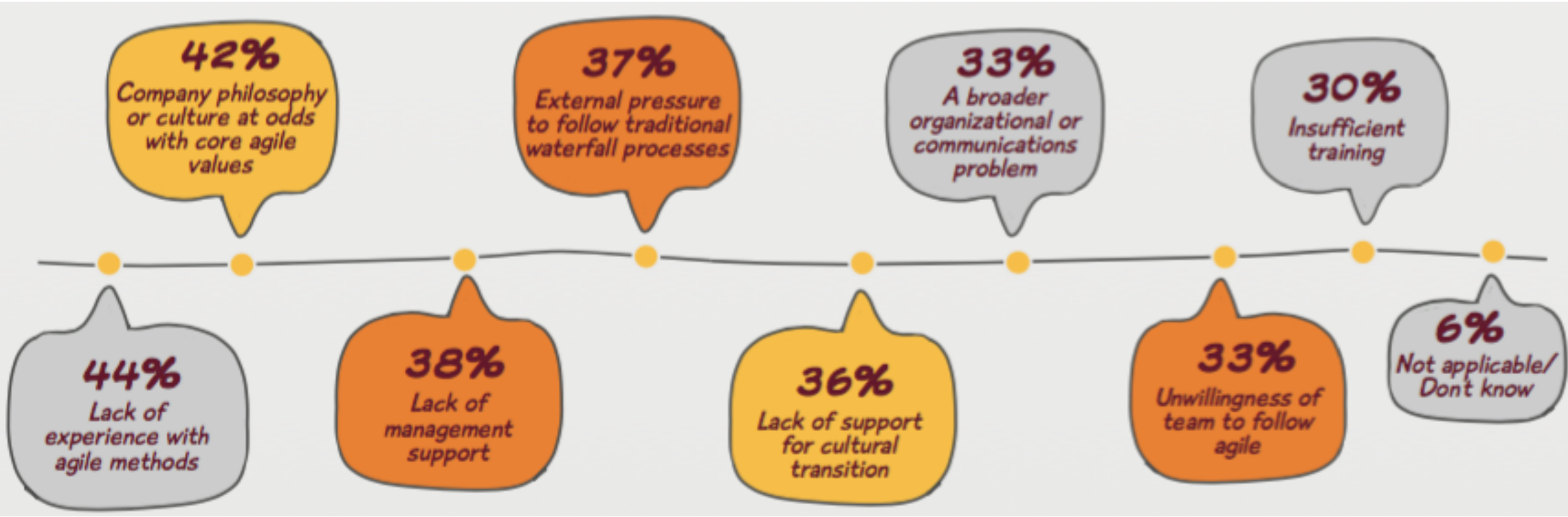
"Please select the methodology that most closely reflects the development process you are currently using."
(select only one)



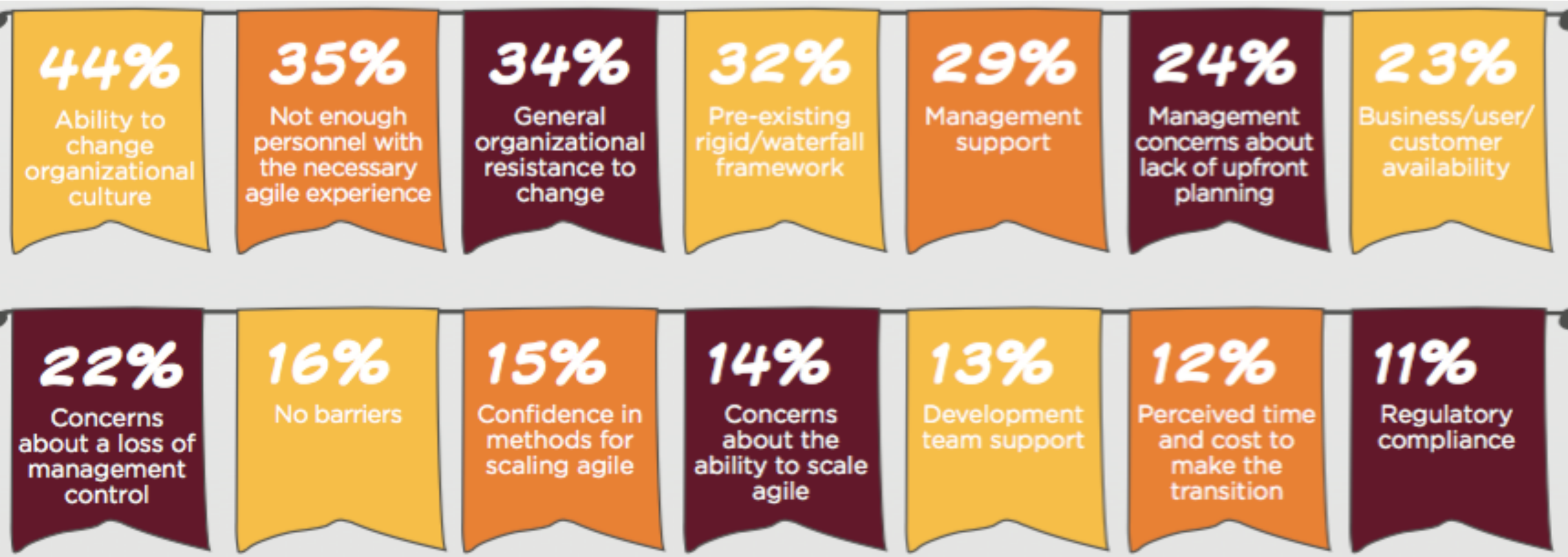
Base: 1,298 IT professionals

Source: Forrester/Dr. Dobb's Global Developer Technographics® Survey, Q3 2009

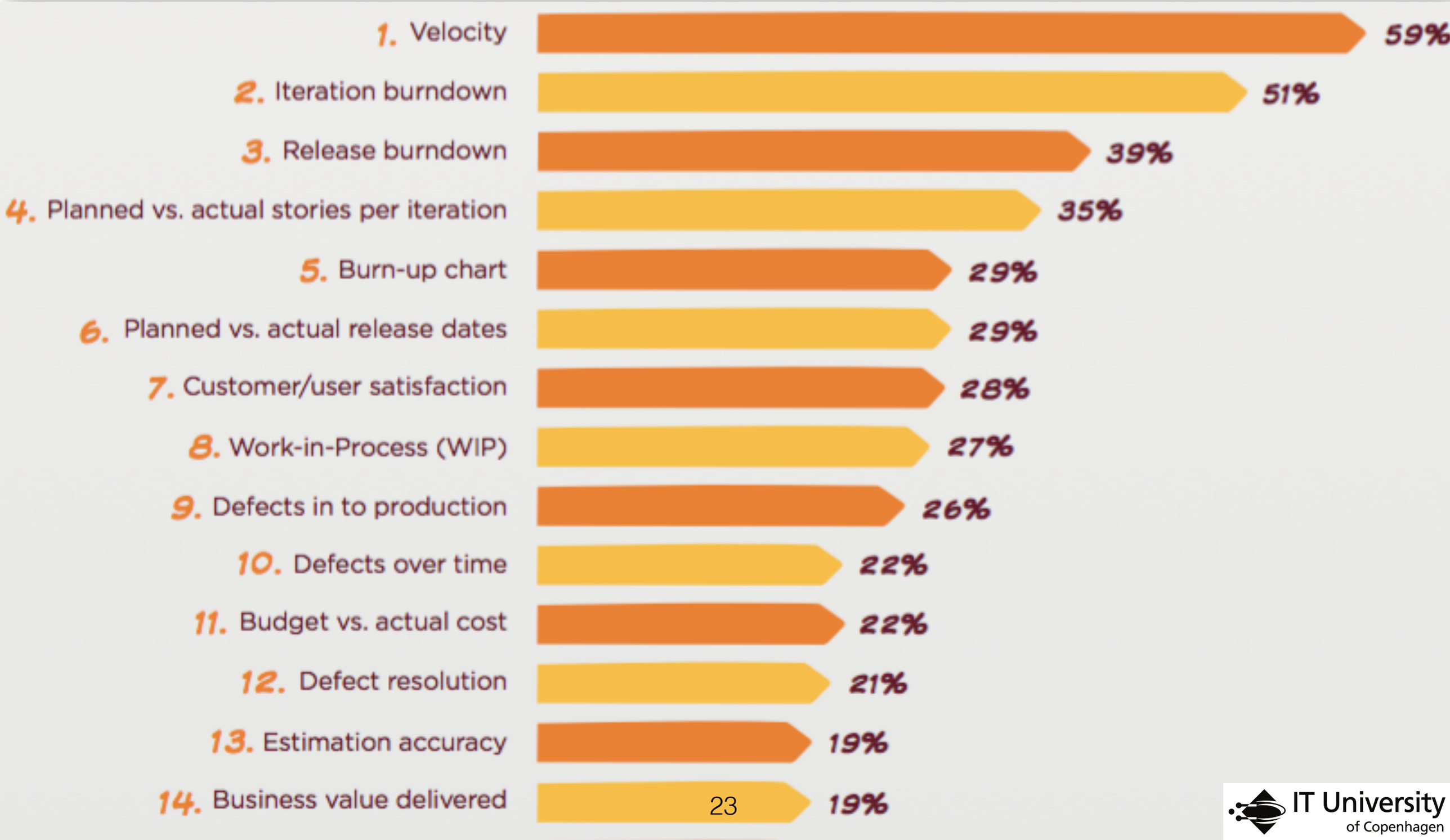
What impedes agile adoption? [versionone survey 2015]

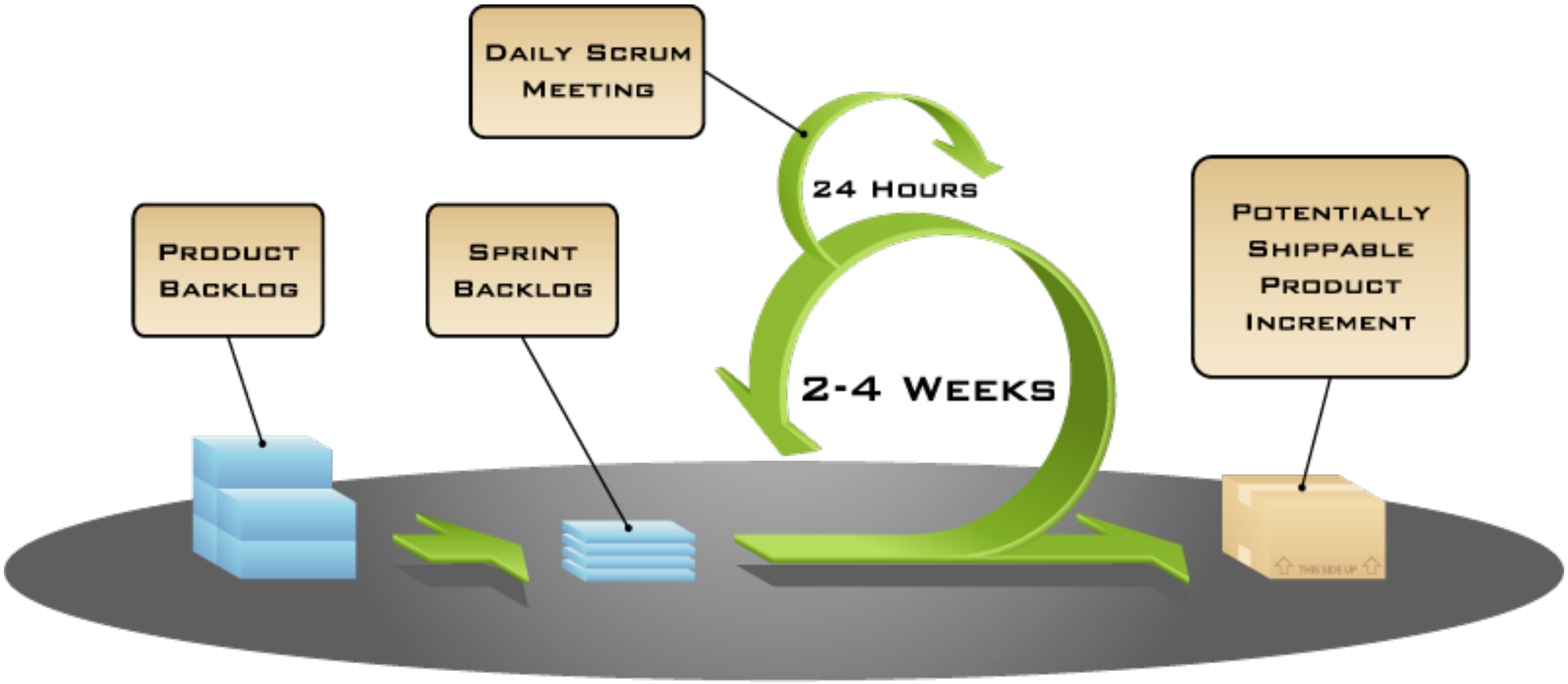


Barriers to adoption [versionone survey 2015]



How is success measured? [versionone survey 2015]

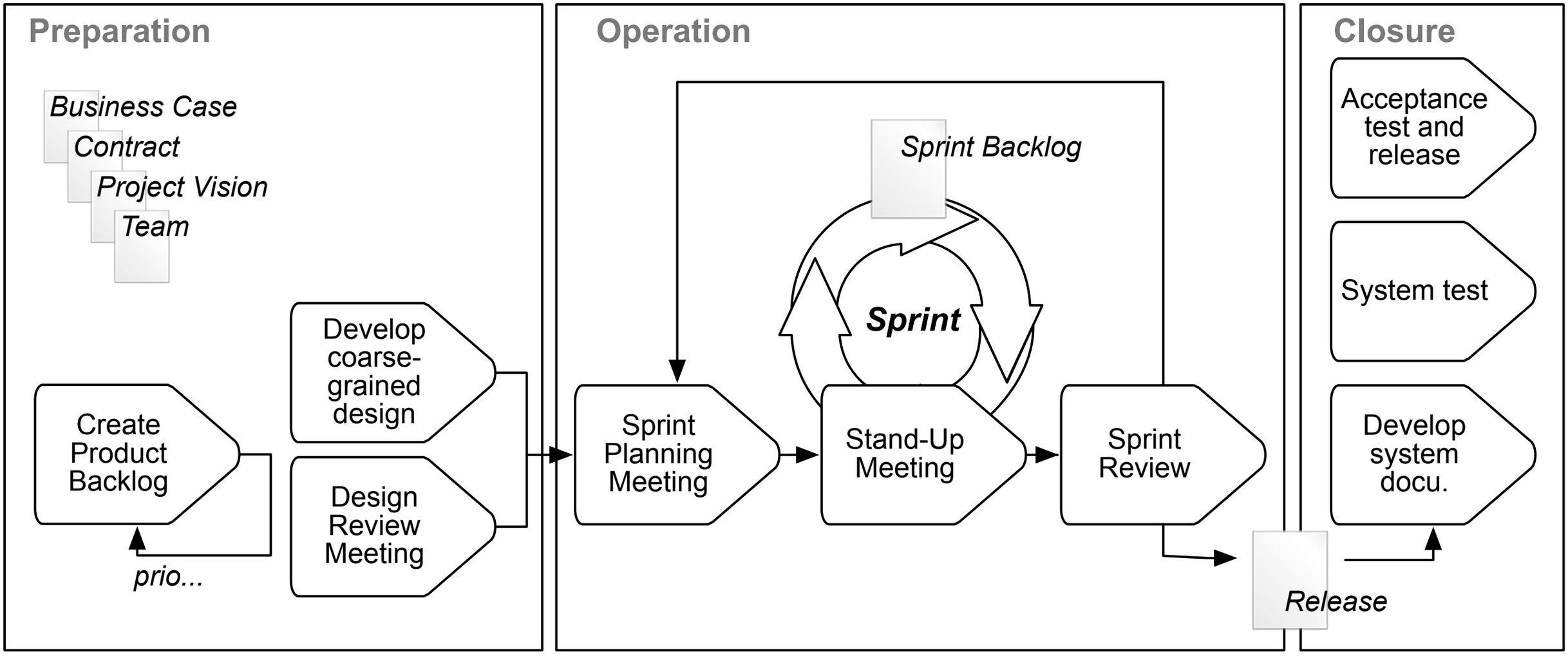




Scrum

Scrum: A more formal perspective

- Scrum is an agile method, which is focused on project management
  - Only few rules and artifacts
  - Focus is a self-organizing, cross-functional team



Agile Methods: Evaluation

- Pro:
  - Change- and feedback- affine
  - Client and team are in the spotlight
  - Approach is pragmatic and goal-oriented

- Con:
  - Requires harmony (“feeling blue” syndrome)
  - Requires highly skilled clients and team members
  - Availability of the client
  - Difficult contracting

Rich Processes

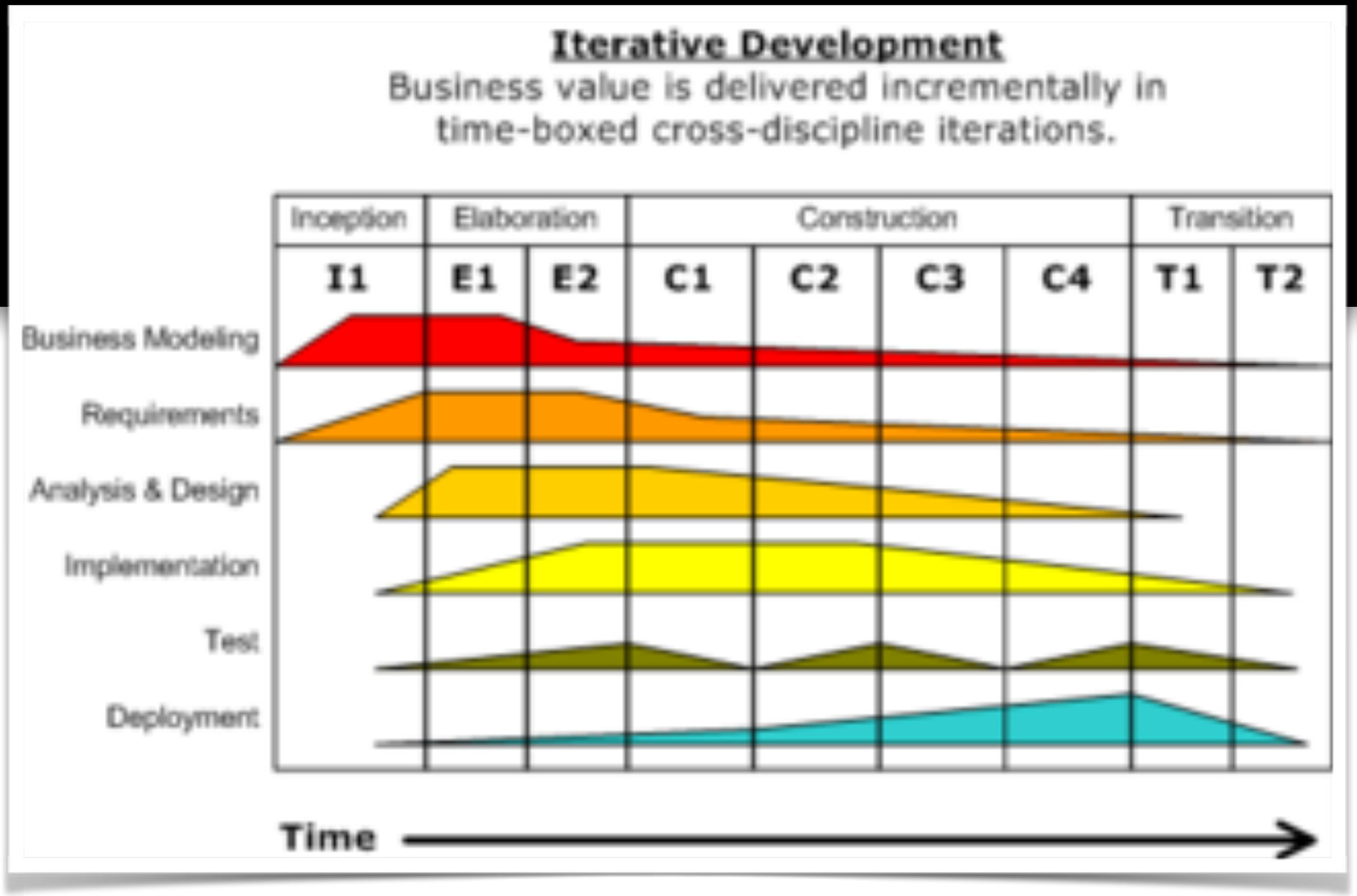
Rich Processes

- **Some Terminology…**
  - Disciplined process
  - Directed process
  - Structured process
  - Heavy-weight process

- Regardless of the terminology:
  A rich process is a means of the constructive management. A rich process describes the requirements regarding:
  - Software projects
  - Expected outcomes
  - Recommended approach
  - Responsibilities
  - …

- **Basic idea:**
  - Define "global" rules and a seamless description
  - Describe all (relevant/possible) aspects for software projects, e.g.:
    - Artifacts, activities, tasks, roles, …
    - Dependencies
    - Valid and consistent configurations
    - …
- Goal: make "gossip" explicit – avoid implicit knowledge (cf. Truck Factor)
- Goal: define reproducible and testable process descriptions (cf. certification)

Example: The RUP

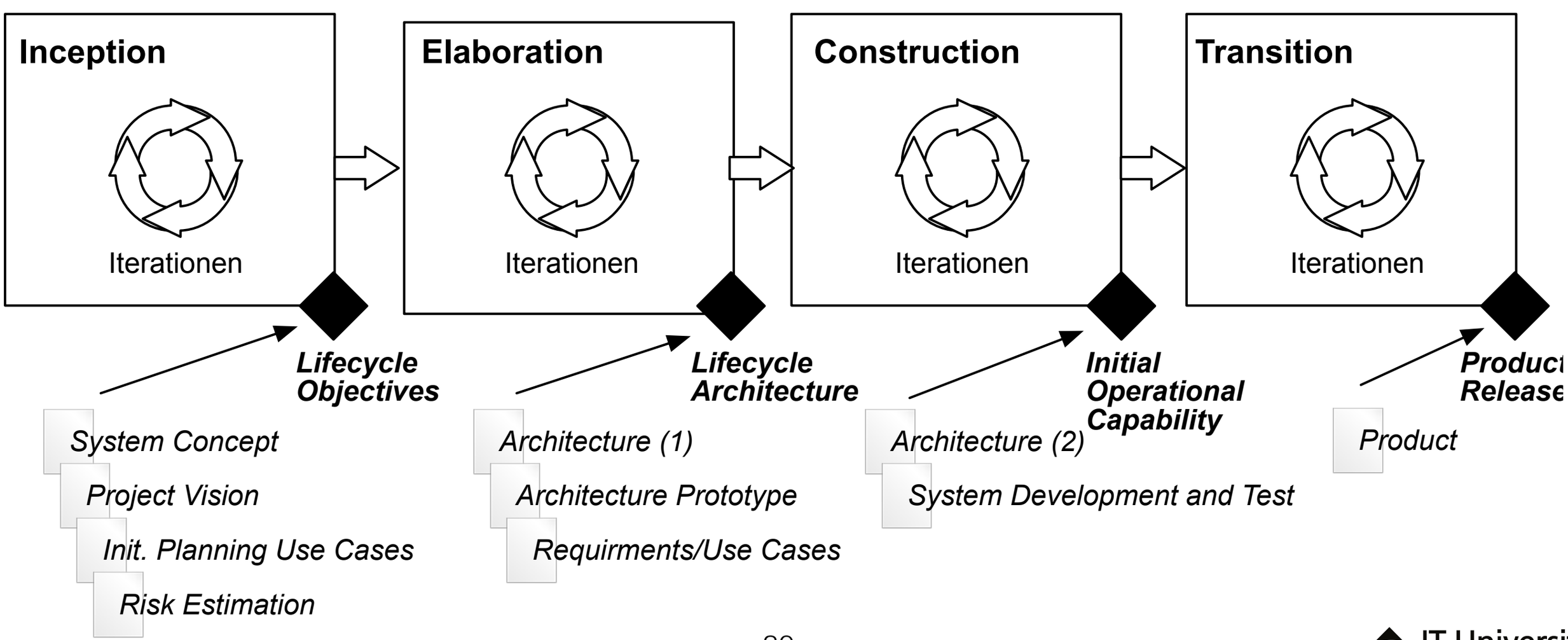


- Rational Unified Process [obsolete]
  - Object- and activity- oriented
  - Developed ~1999 (Rational, later IBM)

- Visualized as Matrix to show focus points
- Visual gives the "RUP Mountains"

Example: The RUP

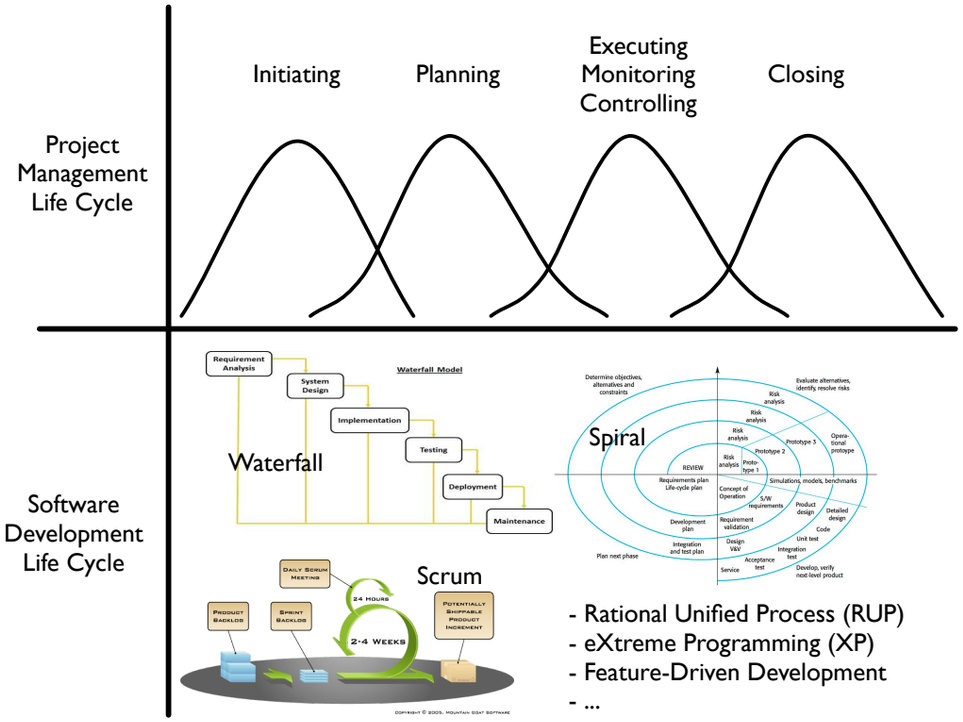
- RUP is a fully-fledged software development model
  - Focused on UML-based software development
  - RUP is an architecture-centric software process



Summary

Software processes are an important tool for the project organization and management

- Define vital project elements
  - Structural- and process- organization
  - Result structure
- Relate management and software technology
- Build the basis for
  - Project planning
  - Contracting
  - Controlling
  - …



- Rational Unified Process (RUP)
- eXtreme Programming (XP)
- Feature-Driven Development
- ...

The Business Model Canvas

Designed for:	Designed by:	On: dd/mm/yyyy
		Iteration #

Problem	Solution	Unique value proposition	Unfair advantage	Customer Segments
top 3 problems	top 3 features	single, clean, compelling message that states why you are different and worth buying	can't be easily copied or bought	target customers
	Key metrics key activities you measure		**Channels** path to customers	

Cost Structure	Revenue Streams
What are the most important costs inherent in our business model? Which Key Resources are most expensive? Which Key Activities are most expensive?	For what value are our customers really willing to pay? For what do they currently pay? How are they currently paying? How would they prefer to pay? How much does each Revenue Stream contribute to overall revenues?

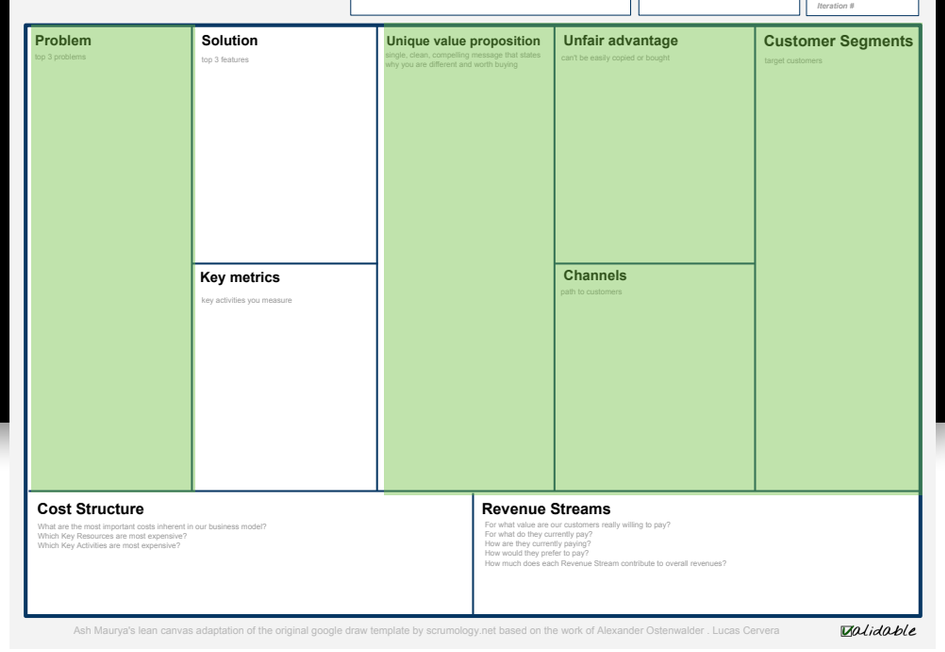
Ash Maurya's lean canvas adaptation of the original google draw template by scrumology.net based on the work of Alexander Ostenwalder . Lucas Cervera

validable

Exercise: Lean business model canvas

IT University of Copenhagen

A guideline



1. **List top 3 problems.** For the customer segment you are working with, describe the top 1-3 problems they need solved.
2. **List existing alternatives.** Then document how you think your early adopters address these problems today. Unless you are solving a brand new problem (unlikely), most problems have existing solutions. Many times these may not be a readily obvious competitor.
3. **Identify other user roles.** Identify any other user roles that will interact with this customer.
4. **Hone in on possible early adopters.** With these problems in mind, get more specific on the customer segment. Narrow down the distinguishing characteristics of your prototypical customer.
5. **Unique value proposition.** State why you are different (in a way that matters). Target early adopters. Focus on the benefits to the customers (not the features). Answer: What, Who, and Why. Create a high-concept pitch.
6. **Unfair advantage.** "A real unfair advantage is something that cannot be easily copied or bought." — Jason Cohen
7. **Channels.** Path to customers and especially your early adopters.