Software Development Processes

Prof. George Candea
School of Computer & Communication Sciences

This is a story about 4 people named ANYBODY, EVERYBODY, SOMEBODY, and NOBODY. There was an important job to be done, and EVERYBODY was sure that SOMEBODY would do it. ANYBODY could have done it, but NOBODY did it. SOMEBODY got angry about that, because it was EVERYBODY's job. EVERYBODY thought ANYBODY could do it, but NOBODY realized that EVERYBODY wouldn't do it. It ended up that EVERYBODY blamed SOMEBODY when NOBODY did what ANYBODY could have done.

Software product lifecycle

Software product lifecycle

1. Requirements gathering & research
2. Specification, planning & design
3. Implementation & testing
4. Deployment
5. Support & maintenance

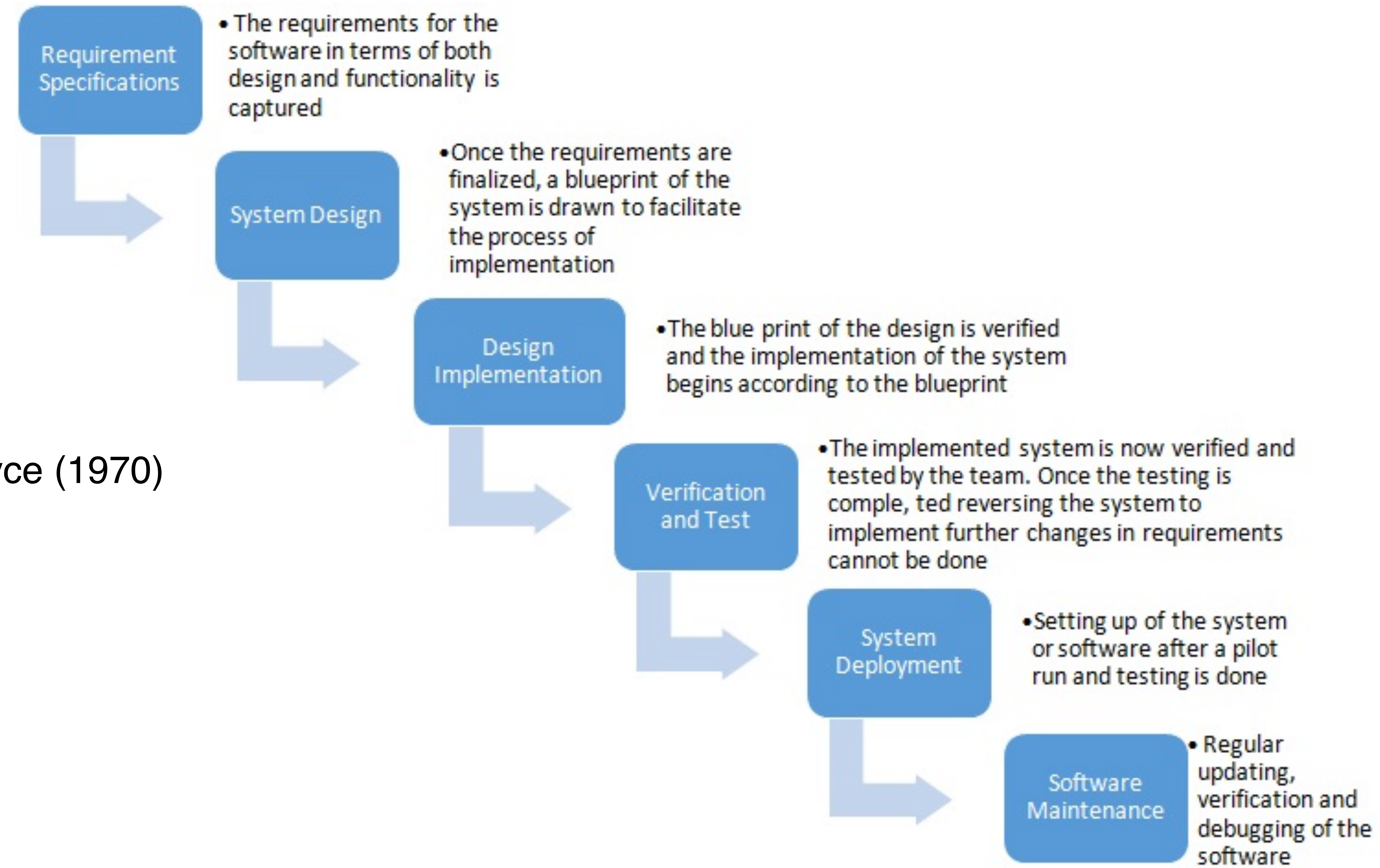
If I'd asked my customers what they wanted, they would have said a faster horse.

(Henry Ford)

Sw Dev Processes: The Waterfall Model

Prof. George Candea
School of Computer & Communication Sciences

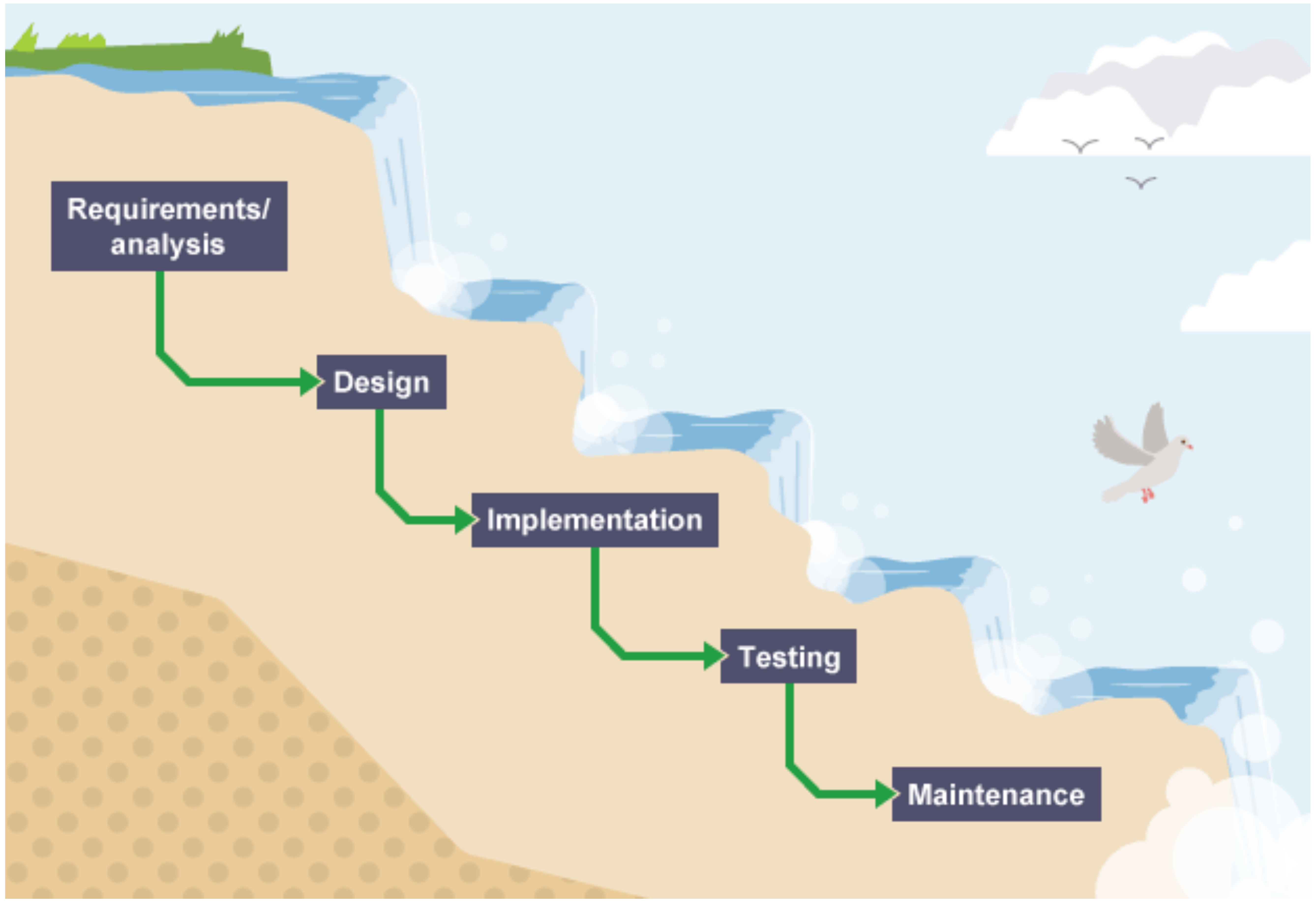
The Waterfall Model



Requirement Specifications
• The requirements for the software in terms of both design and functionality is captured
System Design
•Once the requirements are finalized, a blueprint of the system is drawn to facilitate the process of implementation
Design Implementation
•The blue print of the design is verified and the implementation of the system begins according to the blueprint
Verification and Test
•The implemented system is now verified and tested by the team. Once the testing is comple, ted reversing the system to implement further changes in requirements cannot be done
System Deployment
•Setting up of the system or software after a pilot run and testing is done
Software Maintenance
• Regular updating, verification and debugging of the software

- Winston W. Royce (1970)

Characteristics



- Key feature: linear, sequential
  - *each stage completes before the next starts*
  - *documentation and review at each phase transition*
  - *specifications serve as “contracts”*
  - *“freeze dates”*

Waterfall Strengths

- Early validation (can save 50x - 200x in cost)
  - *enforces stability of requirements*
- Structure + discipline
  - *strong control over process*
  - *good for inexperienced or new staff*
  - *mitigates risk of departing staff*
  - *clear progress metrics, good resource usage*

Waterfall Weaknesses

- Requirements must be known upfront
  - *perfecting a phase before moving on is unrealistic*
  - *many problems can only be discovered by doing*
- Inflexible → slow, costly, cumbersome
  - *high cost/benefit ratio (e.g., lots of documentation)*
- Customer does not get to preview the product
  - *<u>product</u> validation is delayed for a long time*
  - *promotes gap between users and developers*

When to Use ?

- Objectives + solution are clear
    - *product definition can be stabilized*
    - *mature technology, no risk of surprises*
    - *done before*
- Inexperienced project manager or team
- Large, complex projects (enterprise)
    - *project anyway subject to formal approvals*

Incremental Variation

- Waterfall with a divide-and-conquer strategy
  - *break project down into smaller parts*
  - *combine linear model with iterative approach, to reduce project risk*
- Three approaches
  - *sequence of mini-waterfalls / each release adds more functionality to product*
  - *break down into mini-waterfalls to be pursued in parallel (must design interfaces carefully)*
  - *do a waterfall up to (and including) design, then do iterative prototyping*

Strengths of Incremental Variation

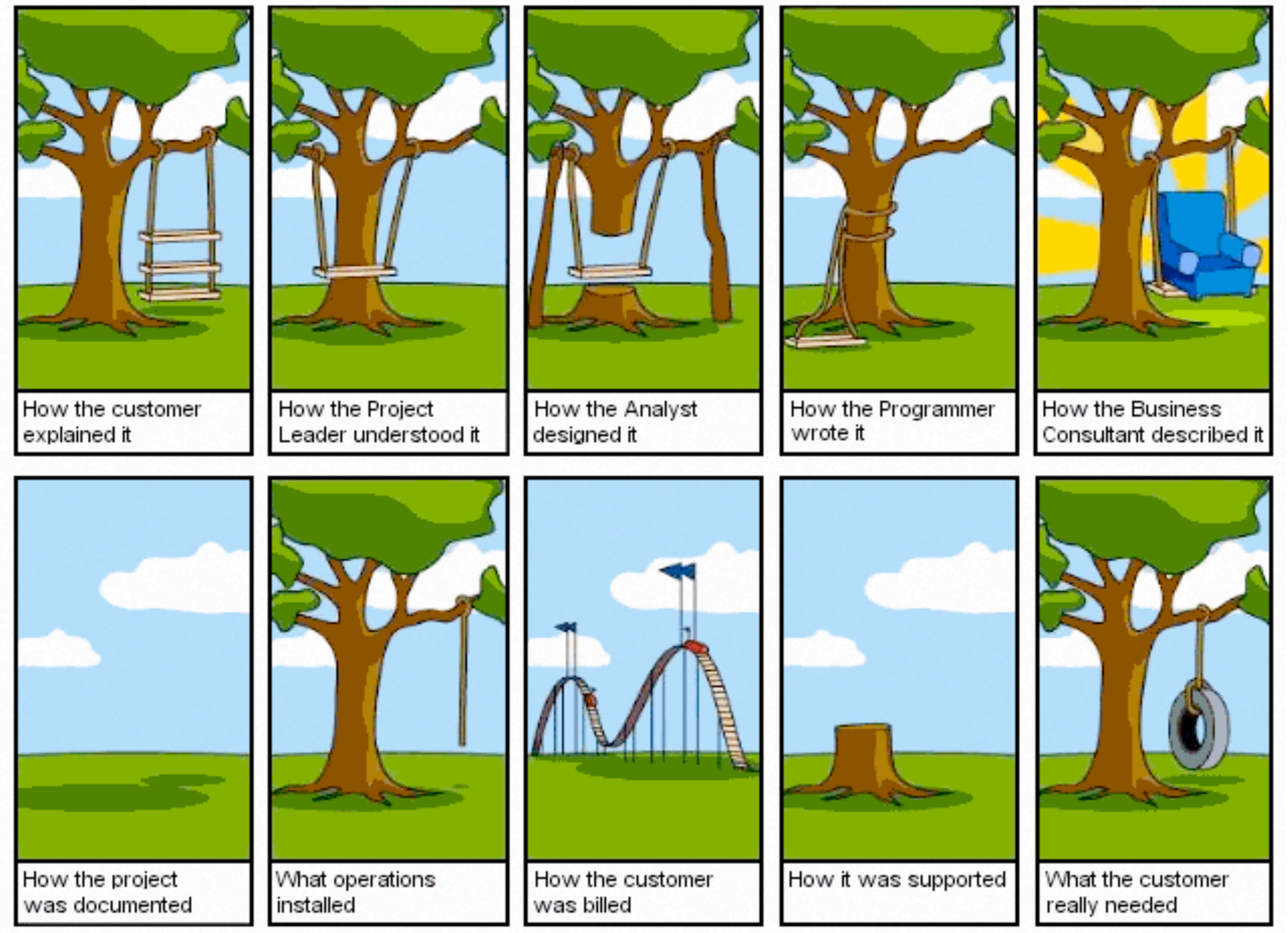
- Can exploit knowledge from prior increments
- Better control (documentation, review, etc.)
- Customer gets important functionality early
- Mitigates integration risks early (through increments)
- Can go into production sooner
- Can accommodate changing requirements

Weaknesses of Incremental Waterfall

- Mini-waterfalls do not encourage big-picture thinking
- Must define good interfaces, or else integration will not work
- Temptation to defer difficult functionality till later
- Not all requirements upfront — incompatibilities found late
    - *still requires good planning and design*

Sw Dev Processes: Agile Methods

Prof. George Candea
School of Computer & Communication Sciences


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Values of Agile Development

Prioritize

• Individuals and interactions • Working software • Customer collaboration • Responding to change	*over*	• Processes and tools • Comprehensive documentation • Contract negotiation • Following a plan

Principles of Agile Development

- Customer satisfaction by early and continuous delivery of valuable software
- Welcome changing requirements, even in late development
- Deliver working software frequently (weeks rather than months)
- Close, daily cooperation between business people and developers
- Projects are built around motivated individuals, who should be trusted
- Face-to-face conversation is the best form of communication (co-location)
- Working software is the primary measure of progress
- Sustainable development, able to maintain a constant pace
- Continuous attention to technical excellence and good design
- Simplicity—the art of maximizing the amount of work not done—is essential
- Best architectures, requirements, and designs emerge from self-organizing teams
- Regularly, the team reflects on how to become more effective, and adjusts accordingly

K. Beck et al., "Principles behind the Agile Manifesto", 2001

Agile Methods

- Emphasize iterative development
  - *a counter-reaction to "heavyweight" methods*
  - *leverage iterative dev (uncover problems early)*
- Adds a people-centric viewpoint
- Principle: “if something is good, do it a lot”
  - *frequent feedback (instead of planning)*
  - *communicate regarding impediments*
  - *use prototypes to learn more about requirements*

Agile Methods

- Scrum
- Kanban
- Adaptive Software Development
- Feature Driven Development
- Crystal Clear
- Extreme Programming
- Rapid Application Development
- Rational Unify Process

Sw Dev Processes: Scrum Framework I

Prof. George Candea

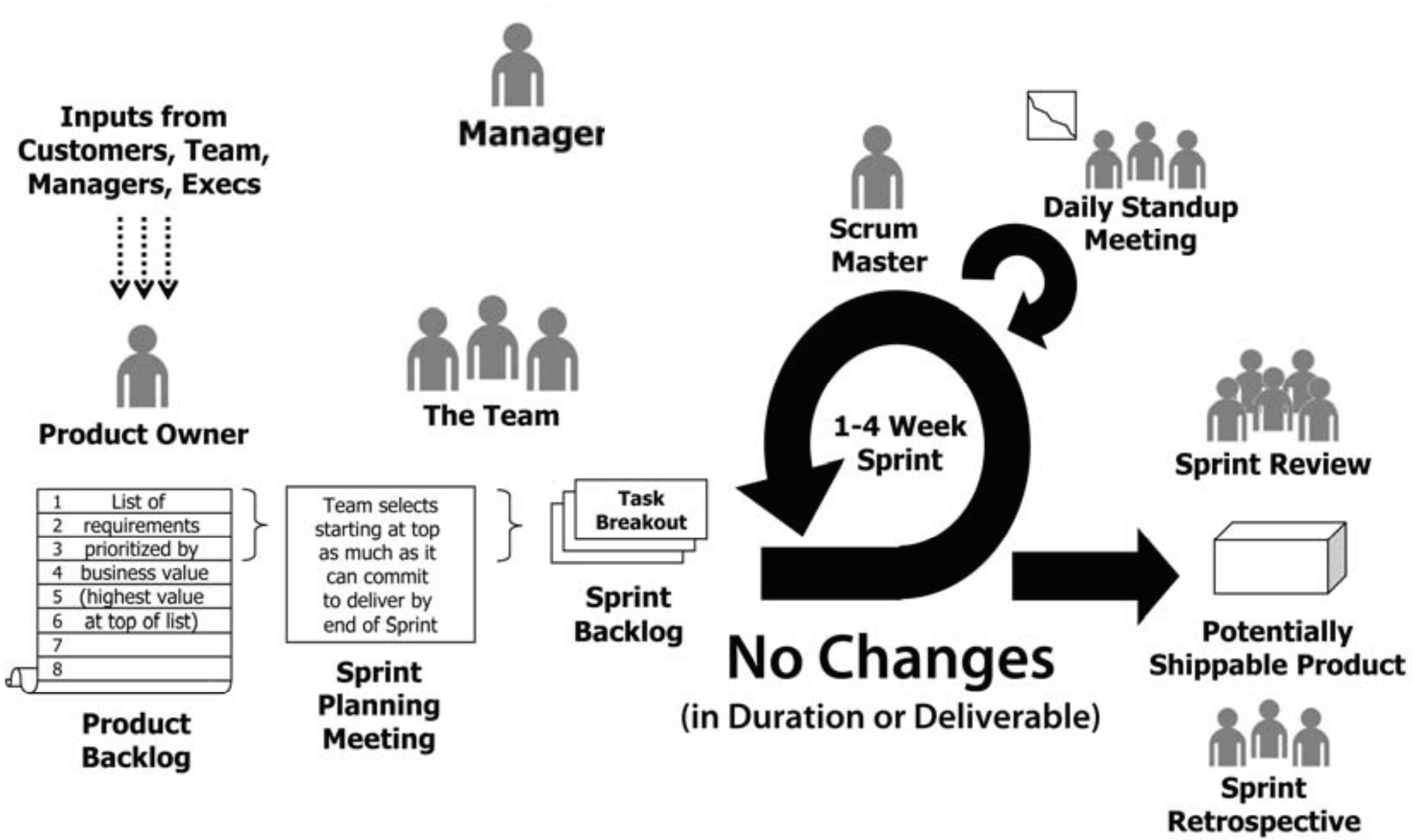
School of Computer & Communication Sciences

Scrum Method

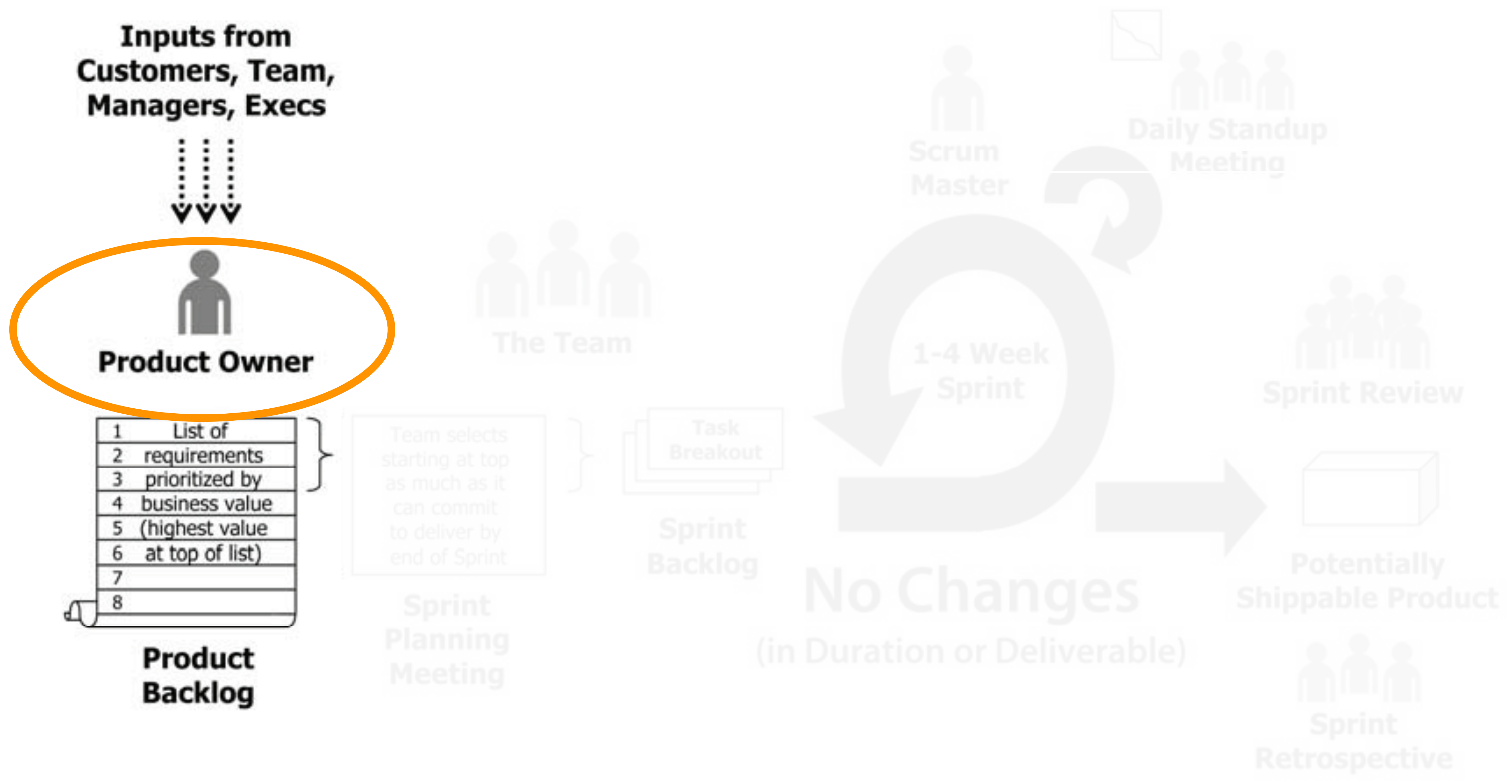
- Emerged in mid-80s, formalized in 1995
  - *OOPSLA paper by Ken Schwaber and Jeff Sutherland*
- Widely used
  - *Yahoo!, Microsoft, Google, Motorola, SAP, Cisco, General Electric, Lockheed Martin, ...*

Scrum Method

- Basic structure = ***Sprint***
  - *1-2 weeks (rigidly fixed length — sometimes longer, but never >1 month)*
  - *one after each other*
  - *<u>working product</u> at the end of <u>each</u> sprint*
- Cross-functional development teams of 3-9 people
- Meet daily

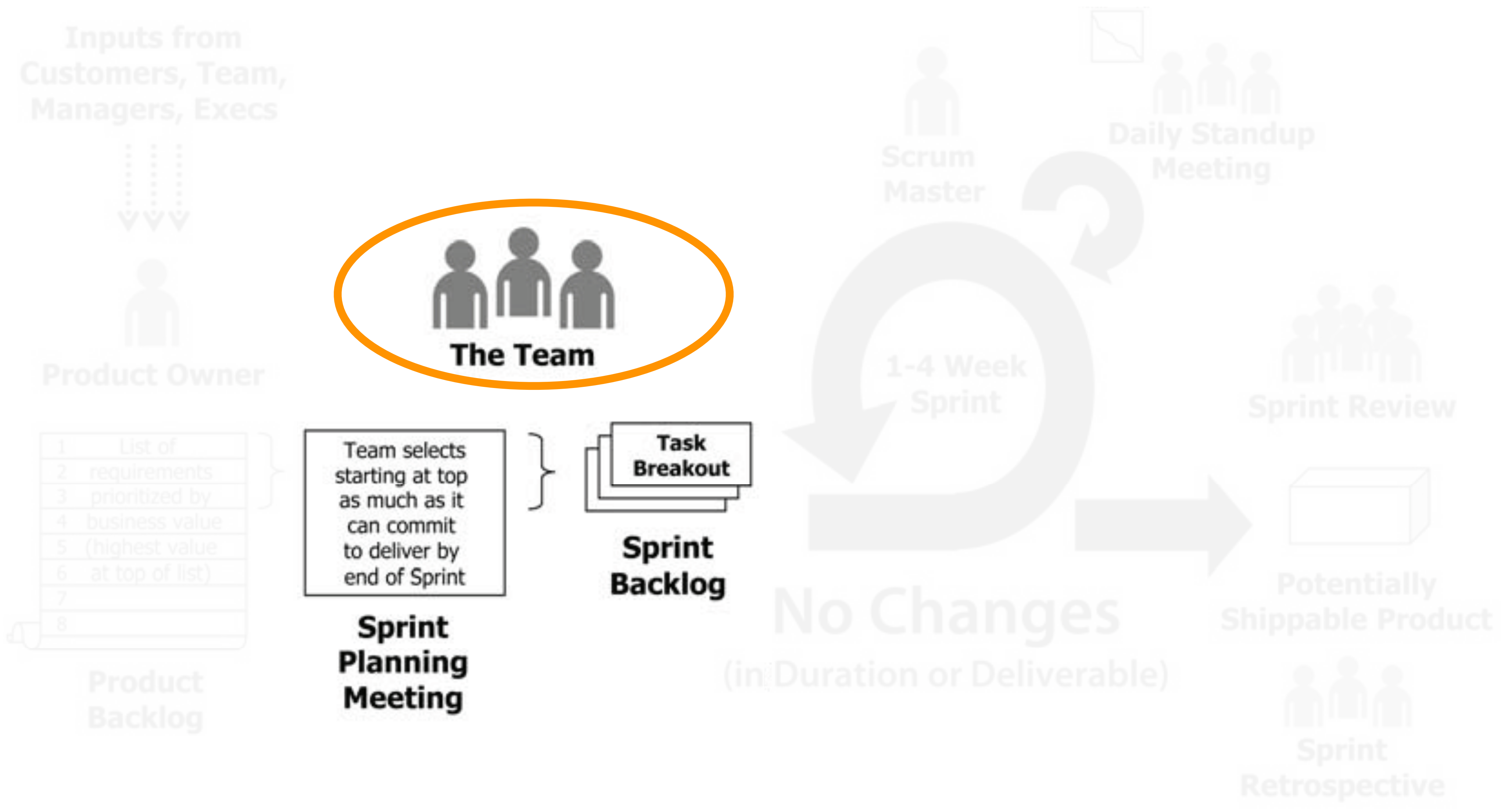

Inputs from Customers, Team, Managers, Execs
Manager
Product Owner
The Team
Scrum Master
Daily Standup Meeting
1-4 Week Sprint
Sprint Review
1 List of
2 requirements
3 prioritized by
4 business value
5 (highest value
6 at top of list)
7
8
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
No Changes
(in Duration or Deliverable)
Potentially Shippable Product
Sprint Retrospective

Scrum Method



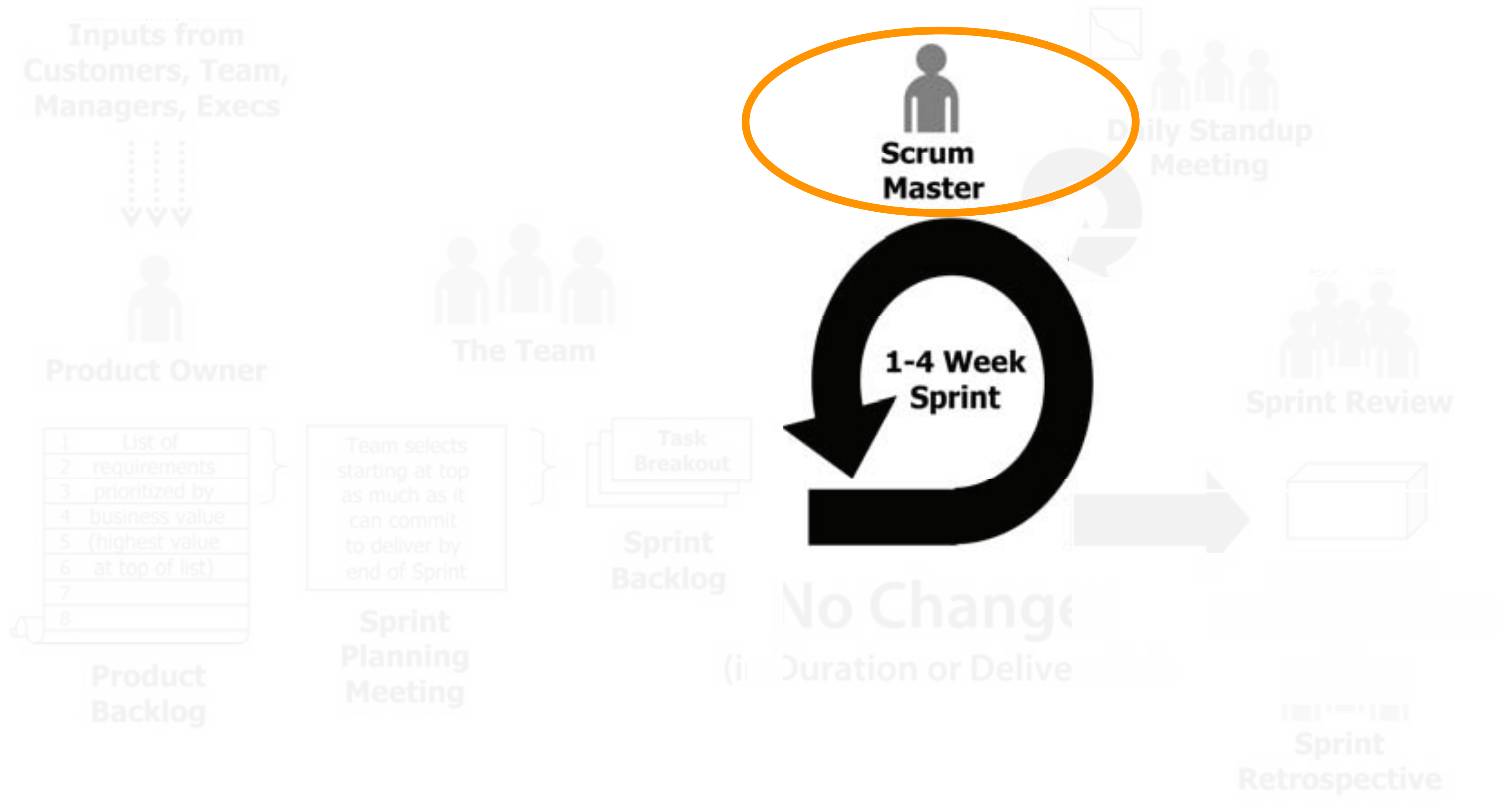
- Product owner
  - *represents customer's or end-user's interests*
  - *maximizes business value*
  - *translates needs into a priority list*
  - *equivalent to traditional product manager*

Scrum Method


Inputs from Customers, Team, Managers, Execs
Product Owner
The Team
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Product Backlog
Scrum Master
Daily Standup Meeting
1-4 Week Sprint
No Changes
(in Duration or Deliverable)
Sprint Review
Potentially Shippable Product
Sprint Retrospective

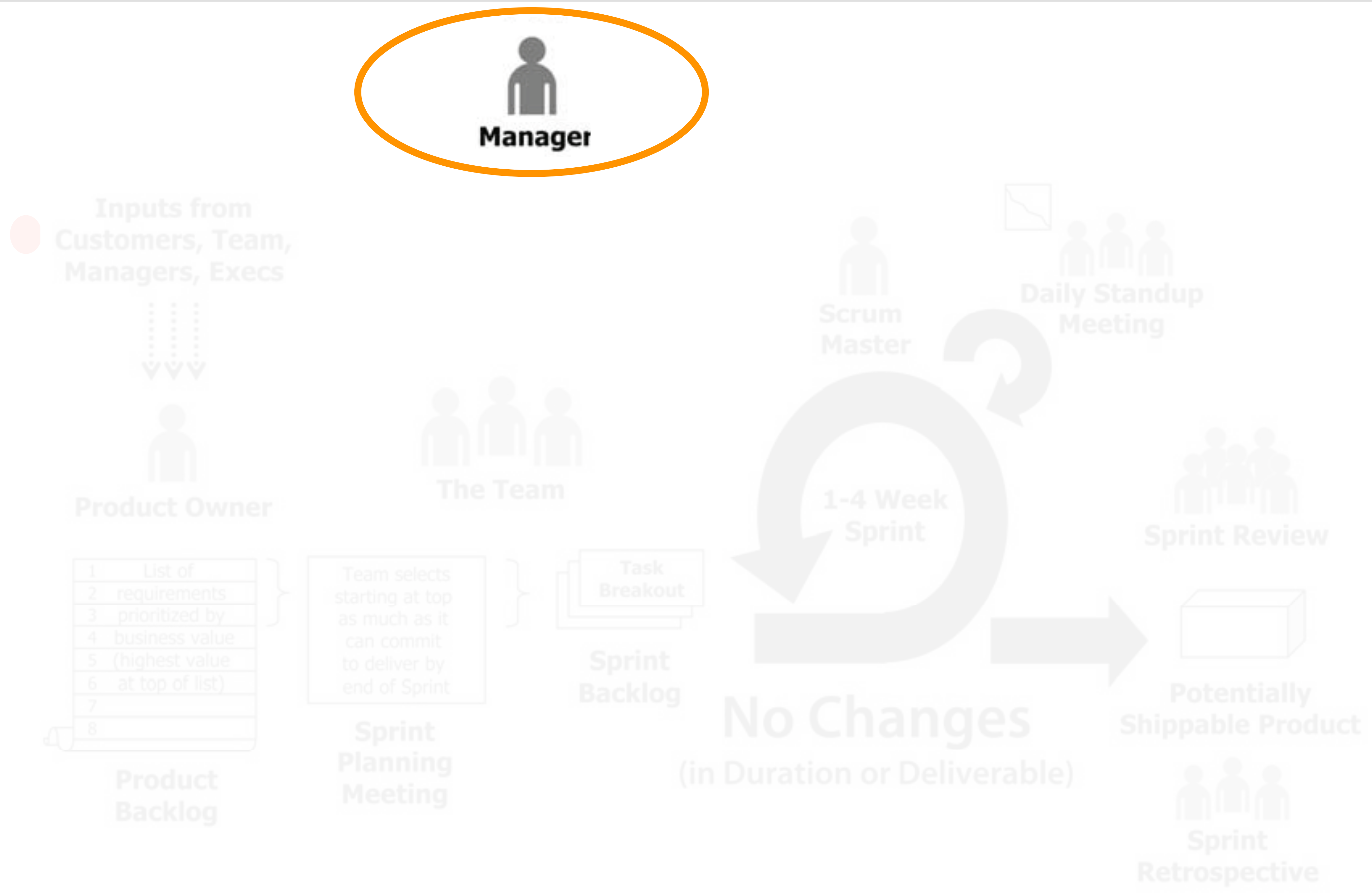
- The Team
  - *builds the product*
  - *cross-functional (developers, UI designers, testers, analysts, ...)*
  - *self-managing → autonomous + accountable*
  - *small (3 - 9 people)*
  - *for large projects, form several Scrum teams*

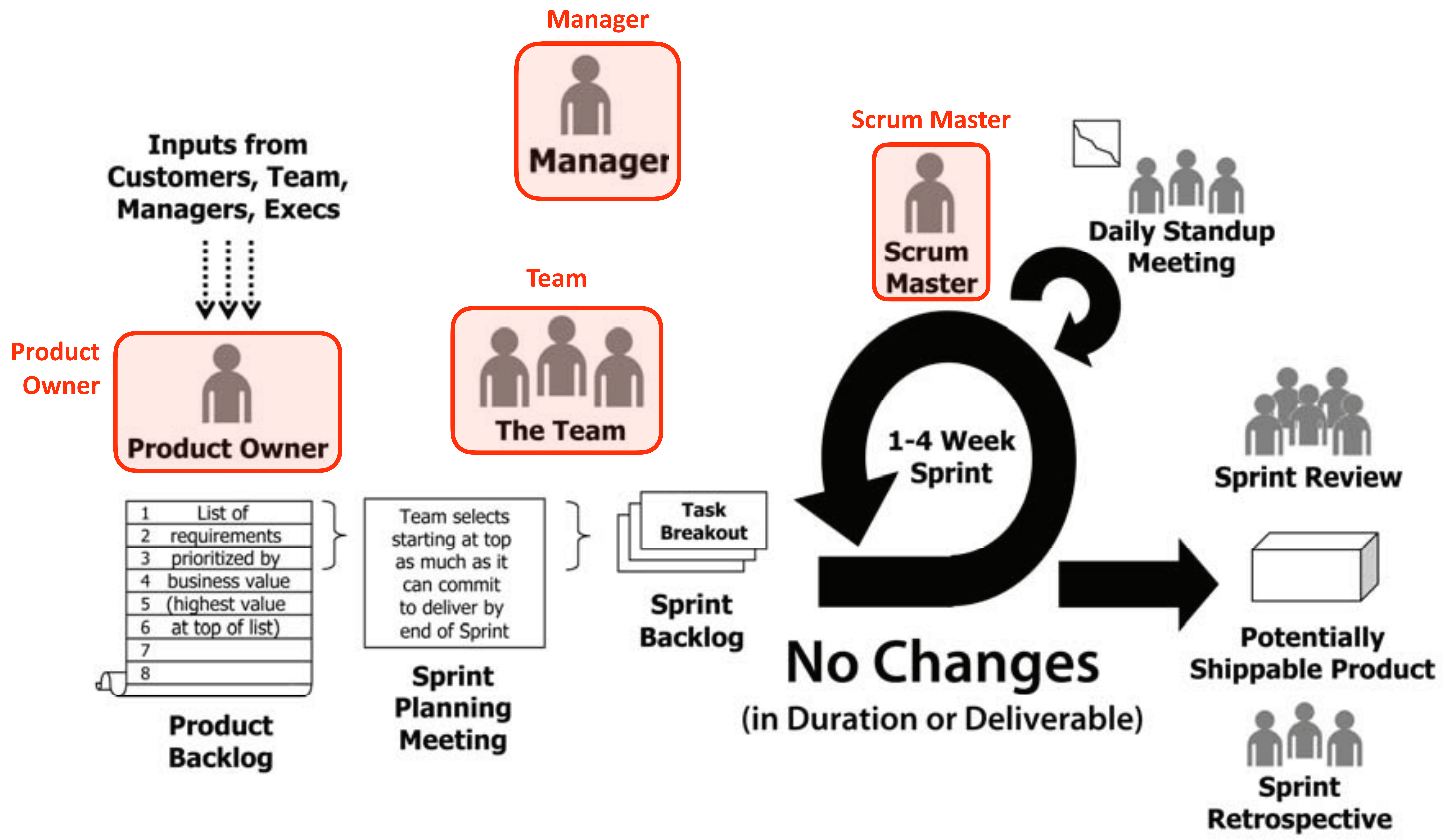
Scrum Method



- Scrum Master
  - *role = ensure team's success*
  - *NOT a manager of the team*
  - *protects team from outside interference (e.g., may push back on product owner)*
  - *helps resolve impediments*
  - *background can be varied: management, engineering, design, testing, etc.*
  - *Scrum Master could be a member of the team*

Scrum Method


Manager
Inputs from Customers, Team, Managers, Execs
Product Owner
The Team
Scrum Master
Daily Standup Meeting
1-4 Week Sprint
Sprint Review
1 List of 2 requirements 3 prioritized by 4 business value 5 (highest value 6 at top of list) 7 8
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
No Changes
(in Duration or Deliverable)
Potentially Shippable Product
Sprint Retrospective

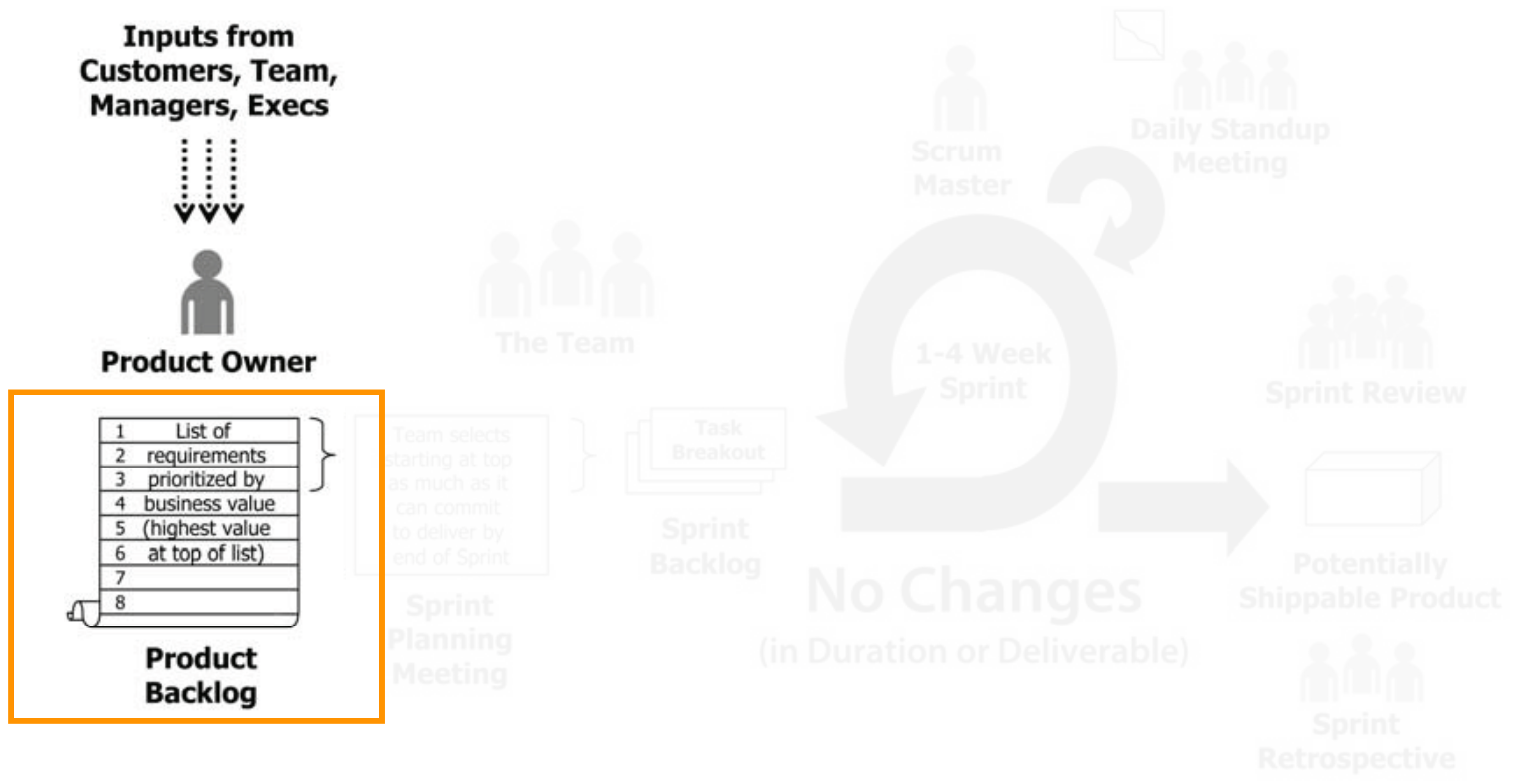

Manager
Manager
Inputs from Customers, Team, Managers, Execs
Scrum Master
Scrum Master
Daily Standup Meeting
Product Owner
Product Owner
Team
The Team
1-4 Week Sprint
Sprint Review
1 List of
2 requirements
3 prioritized by
4 business value
5 (highest value
6 at top of list)
7
8
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
No Changes
(in Duration or Deliverable)
Potentially Shippable Product
Sprint Retrospective

Sw Dev Processes: Scrum Framework II

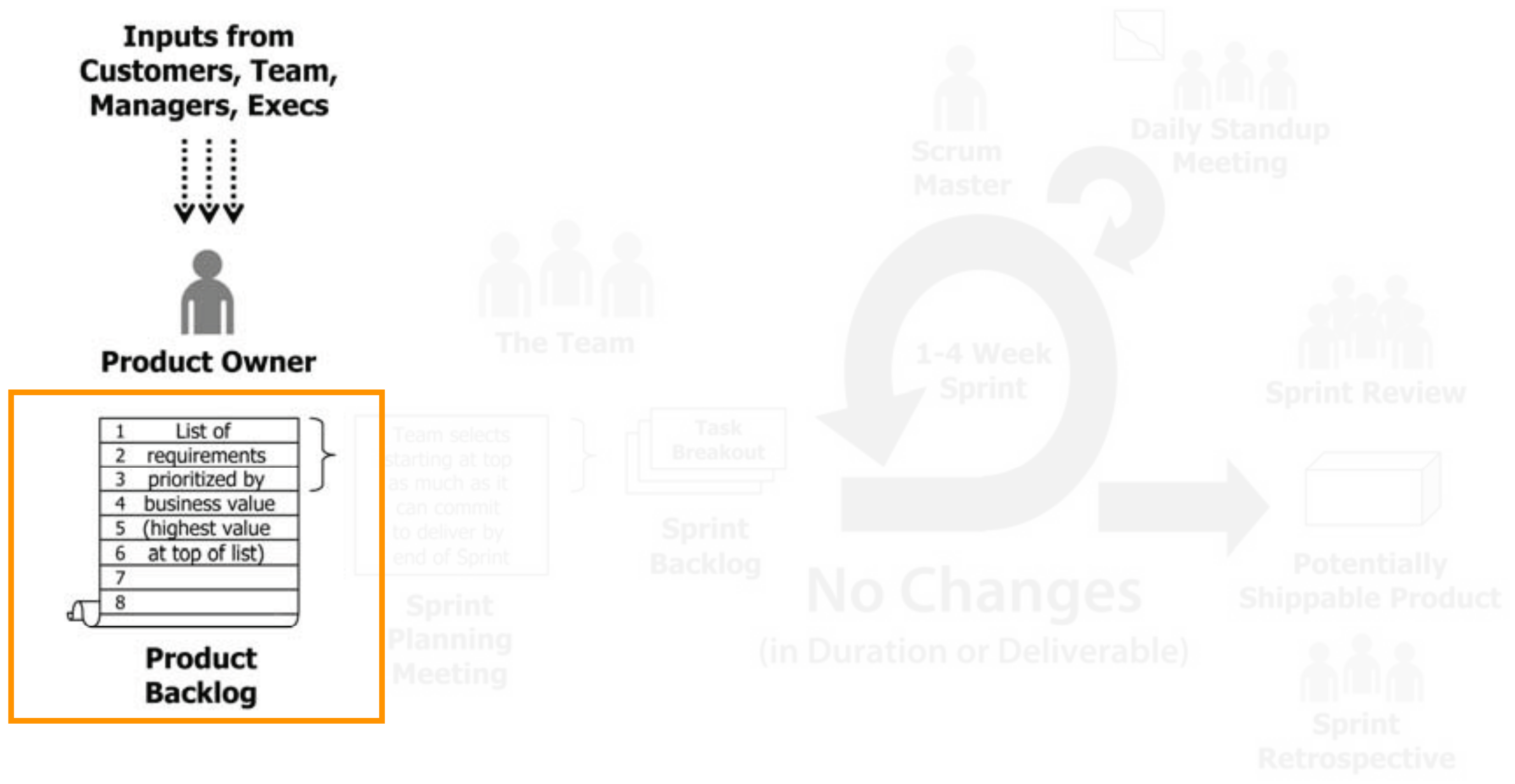
Prof. George Candea

School of Computer & Communication Sciences

Scrum Method



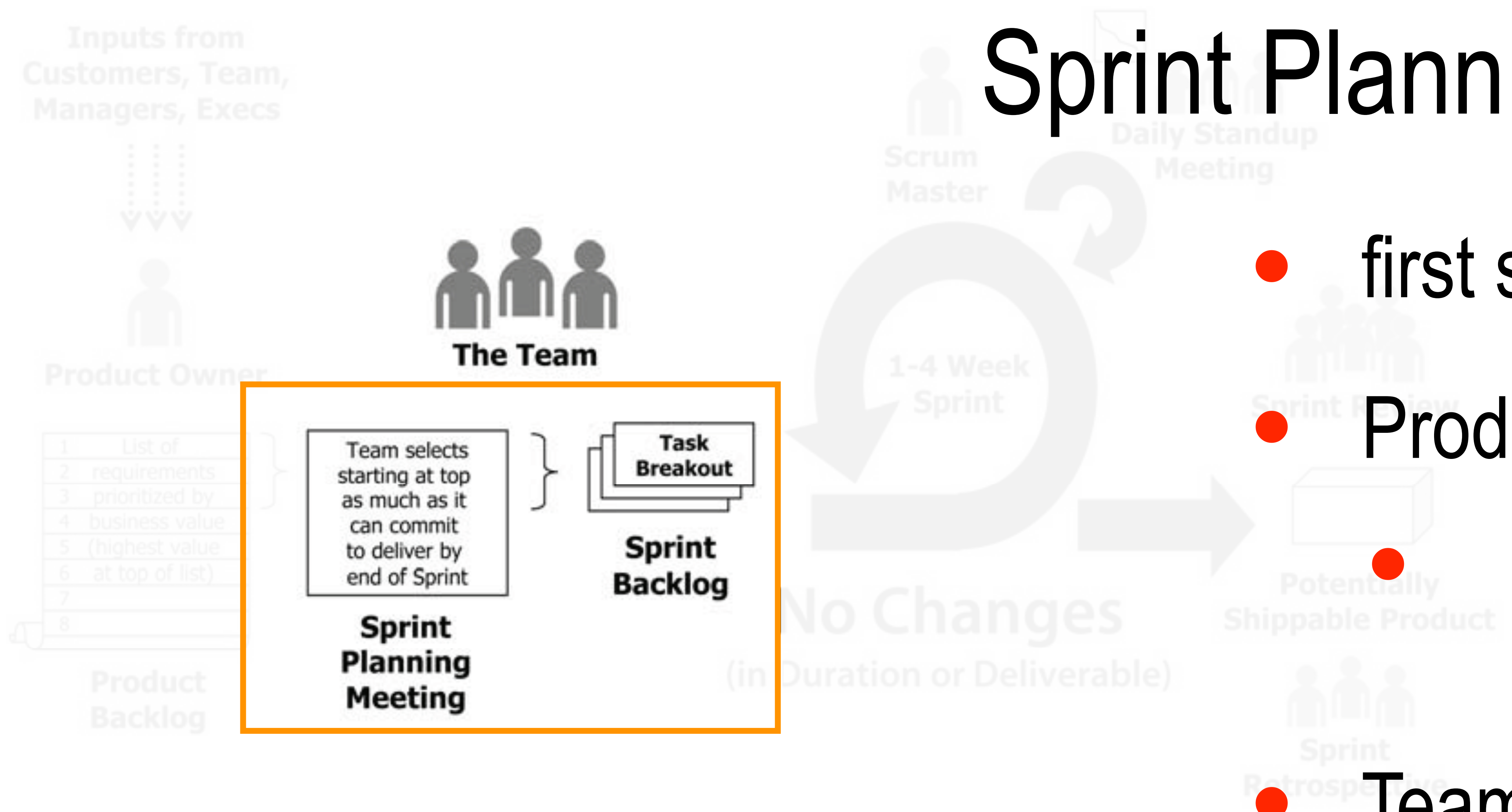
Scrum Method



Product Backlog

- Product Owner articulates product vision
- To-do list prioritized by value to customer
  - *contains features, development requirements, research / investigative tasks, bugs*
  - *articulated in terms of "user stories"*
- evolves over time (change is a given!)
  - *is the definitive view of "everything that could be done by the team ever, in order of priority"*
- Team provides time estimates
  - *product Owner uses them to prioritize Backlog*

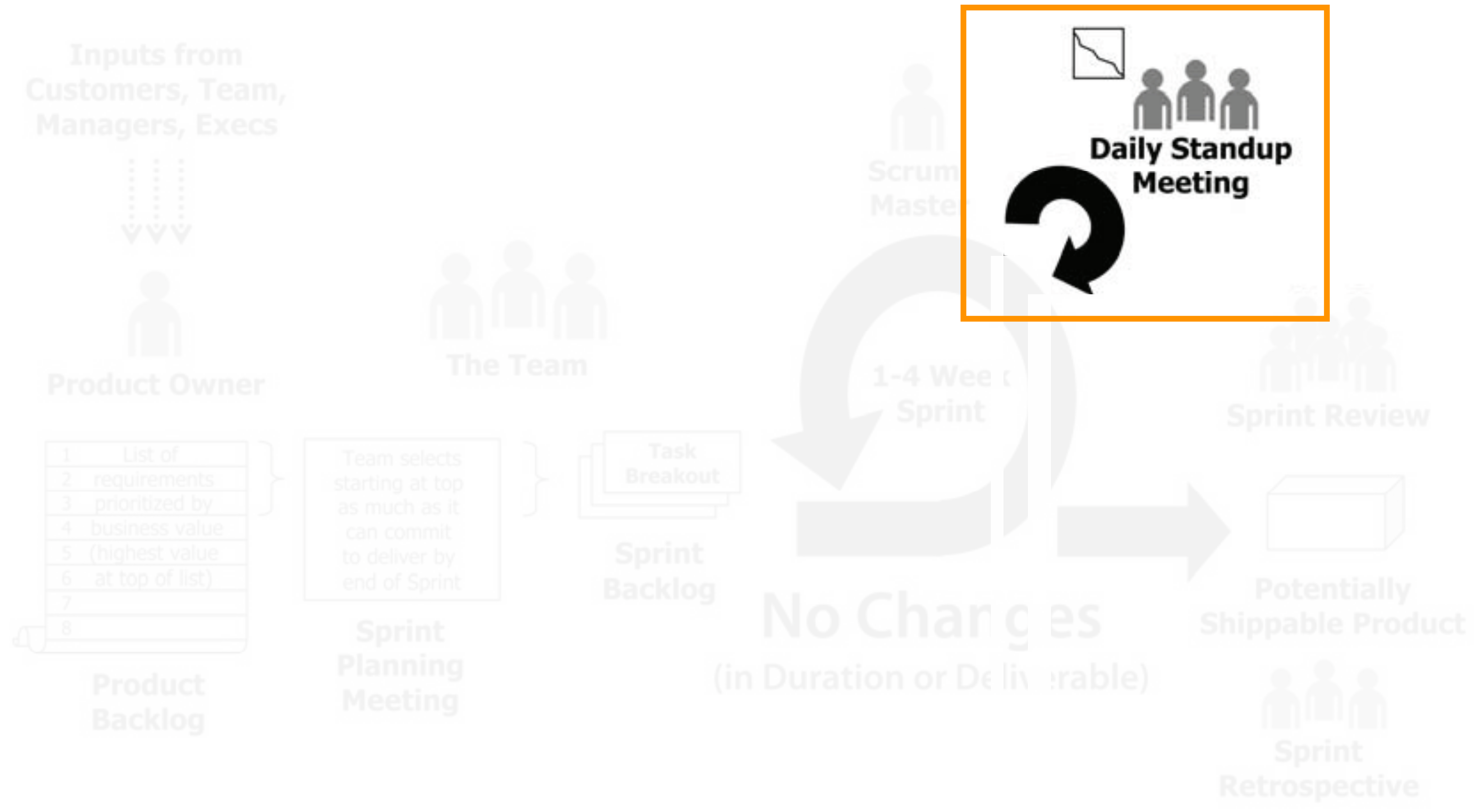
Scrum Method


Inputs from Customers, Team, Managers, Execs
Product Owner
Product Backlog
The Team
Team selects starting at top as much as it can commit to deliver by end of Sprint
Task Breakout
Sprint Backlog
Sprint Planning Meeting
Scrum Master
Daily Standup Meeting
1-4 Week Sprint
No Changes
(in Duration or Deliverable)
Sprint Review
Potentially Shippable Product
Sprint Retrospective

Sprint Planning Meeting

- first step of every Sprint
- Product Owner and Team review Product Backlog
  - *Gives team insight into the thinking of the customer*
- Team selects items they can complete in this Sprint (starting from the top)
- Team breaks each item down into tasks, thus producing the Sprint Backlog
- once Team has committed, no changes to Product Backlog

Scrum Method

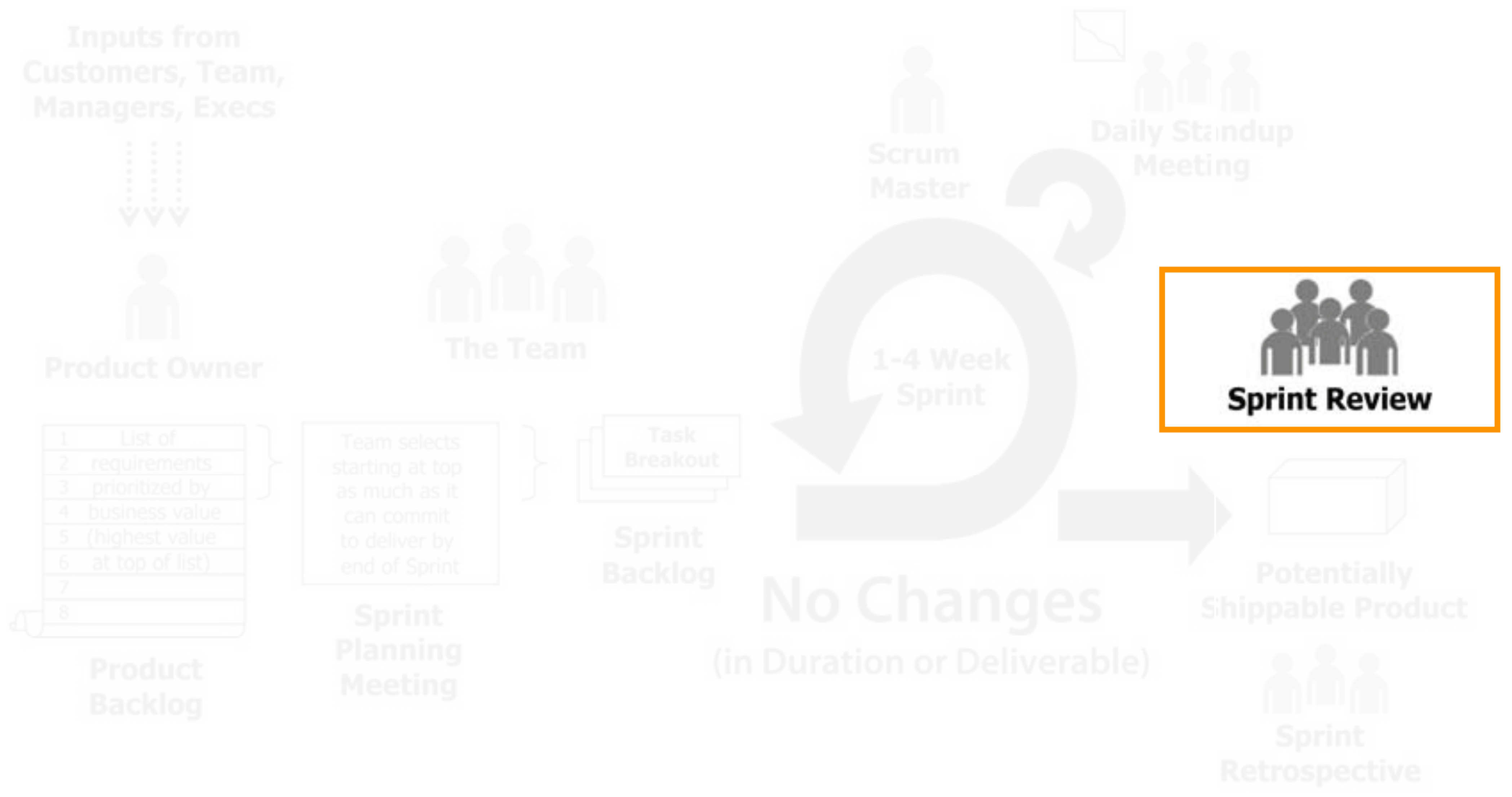


- Daily Stand-Up
  - *stand-up meeting at fixed time every day*
  - *meeting lasts <= 15 minutes*
  - *each team member reports 3 items:*
    - What (s)he has done since last meeting
    - What (s)he will do until next meeting
    - Any blocks or impediments
  - *no discussion, just reporting*
- After meeting
  - *Scrum Master resolves impediments and updates progress metrics*

Sprint Duration

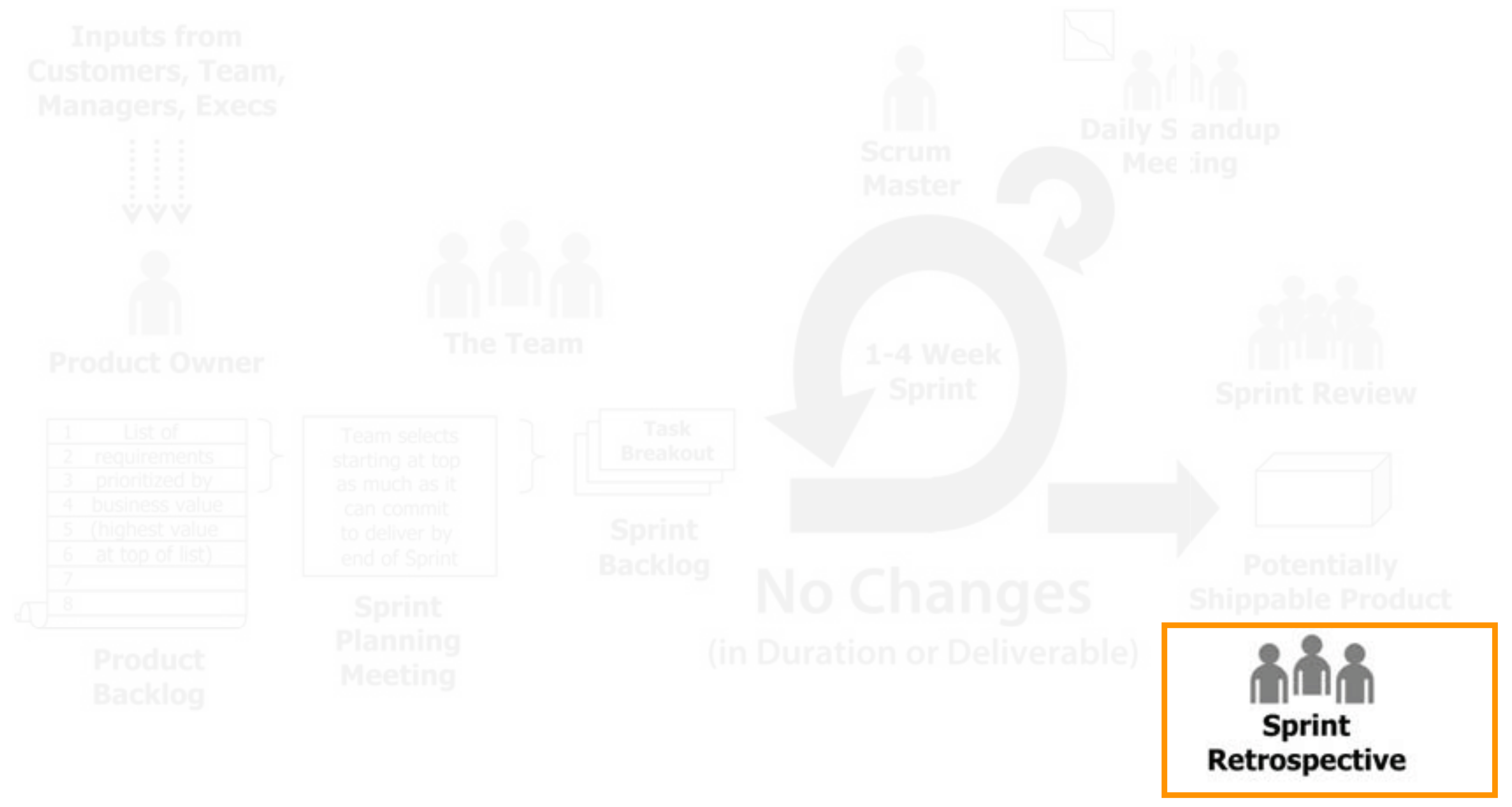
- A Sprint is never extended
- If goals not meet, team must own up to it
  - *takes some experience to determine how long tasks will take → over time, team gets better at it*
- Pick one Sprint duration and stick to it
  - *helps team improve their estimates*
  - *for SDP project: 1 week (sometimes 2 weeks)*

Scrum Method



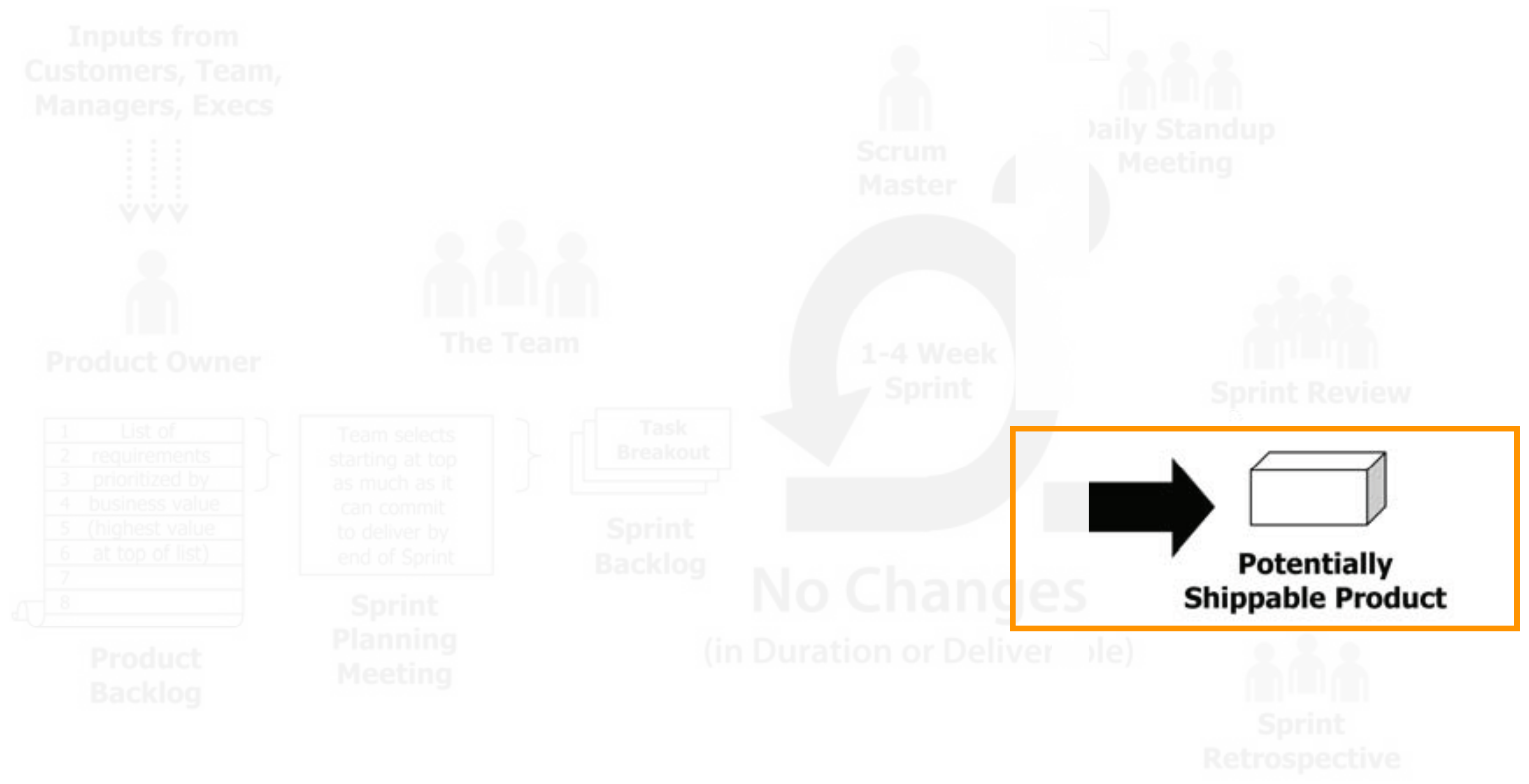
- Sprint Review
  - *demo the product (<30 minutes prep)*
  - *participants: Team + ScrumMaster + Product Owner + customers + stakeholders + experts ...*
  - *anyone can ask questions*
  - *meeting lasts as long as necessary*

Scrum Method

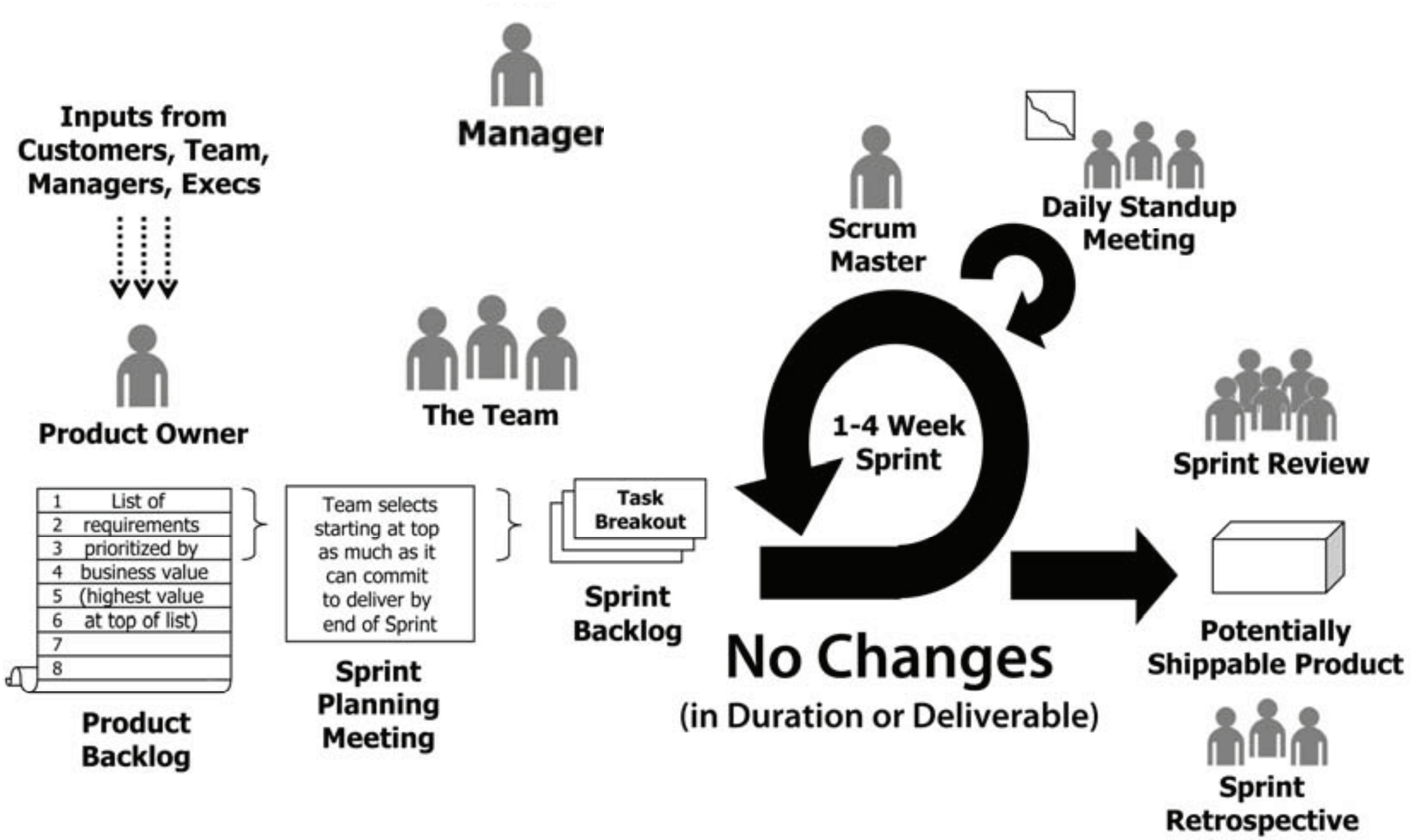


- Sprint Retrospective
  - *participants: Team + ScrumMaster + Product Owner*
  - *facilitated by neutral outsider (e.g., other ScrumMaster → good for cross-polination)*
  - *"What went wrong?" … "How can we improve?" …*
  - *Product Owner updates the Product Backlog (feeds into next Sprint Planning Meeting)*
  - *no downtime between Sprints, maintain the pace*

The End ?


Inputs from Customers, Team, Managers, Execs
Product Owner
The Team
Scrum Master
Daily Standup Meeting
1-4 Week Sprint
Sprint Review
List of requirements prioritized by business value (highest value at top of list)
Team selects starting at top as much as it can commit to deliver by end of Sprint
Task Breakout
Sprint Backlog
Potentially Shippable Product
No Changes
(in Duration or Deliverable)
Product Backlog
Sprint Planning Meeting
Sprint Retrospective

- Once Product Owner decides product is ready, do a final Release Sprint https://scrumguides.org/scrum-guide.html


Inputs from Customers, Team, Managers, Execs
Manager
Product Owner
The Team
Scrum Master
Daily Standup Meeting
1-4 Week Sprint
Sprint Review
1 List of
2 requirements
3 prioritized by
4 business value
5 (highest value
6 at top of list)
7
8
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
No Changes
(in Duration or Deliverable)
Potentially Shippable Product
Sprint Retrospective