Tim Pengajar IF2250

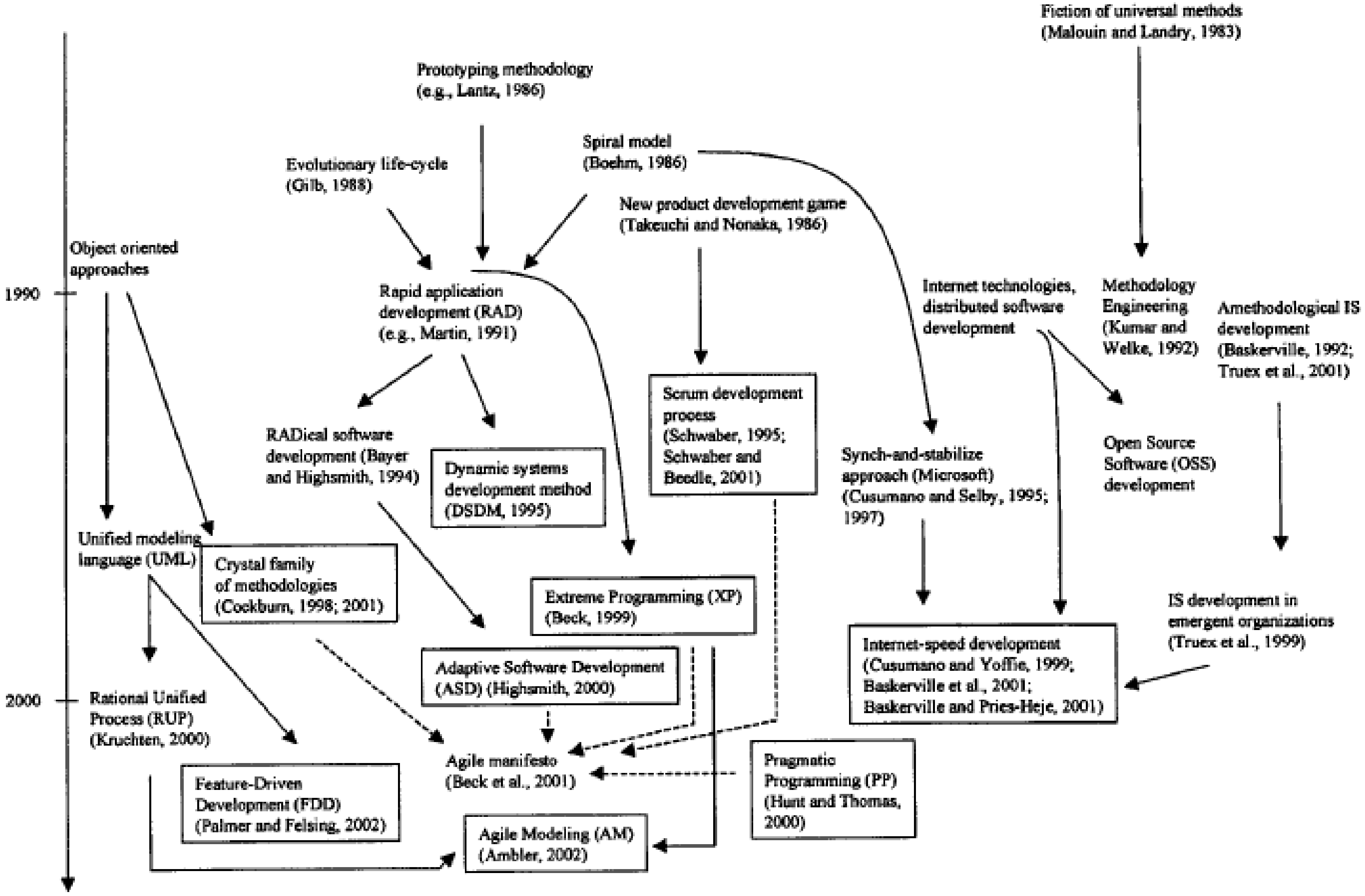
IF2250 – Rekayasa Perangkat Lunak

Agile Process dan Scrum

SEMESTER II TAHUN AJARAN 2022/2023

INSTITUT TEKNOLOGI BANDUNG 1920

KNOWLEDGE & SOFTWARE ENGINEERING



1990

2000

Object oriented approaches

Prototyping methodology (e.g., Lantz, 1986)

Evolutionary life-cycle (Gilb, 1988)

Spiral model (Boehm, 1986)

Fiction of universal methods (Malouin and Landry, 1983)

New product development game (Takeuchi and Nonaka, 1986)

Rapid application development (RAD) (e.g., Martin, 1991)

Internet technologies, distributed software development

Methodology Engineering (Kumar and Welke, 1992)

Amethodological IS development (Baskerville, 1992; Truex et al., 2001)

RADical software development (Bayer and Highsmith, 1994)

Dynamic systems development method (DSDM, 1995)

Scrum development process (Schwaber, 1995; Schwaber and Beedle, 2001)

Synch-and-stabilize approach (Microsoft) (Cusumano and Selby, 1995; 1997)

Open Source Software (OSS) development

Unified modeling language (UML)

Crystal family of methodologies (Cockburn, 1998; 2001)

Extreme Programming (XP) (Beck, 1999)

IS development in emergent organizations (Truex et al., 1999)

Internet-speed development (Cusumano and Yoffie, 1999; Baskerville et al., 2001; Baskerville and Pries-Heje, 2001)

Adaptive Software Development (ASD) (Highsmith, 2000)

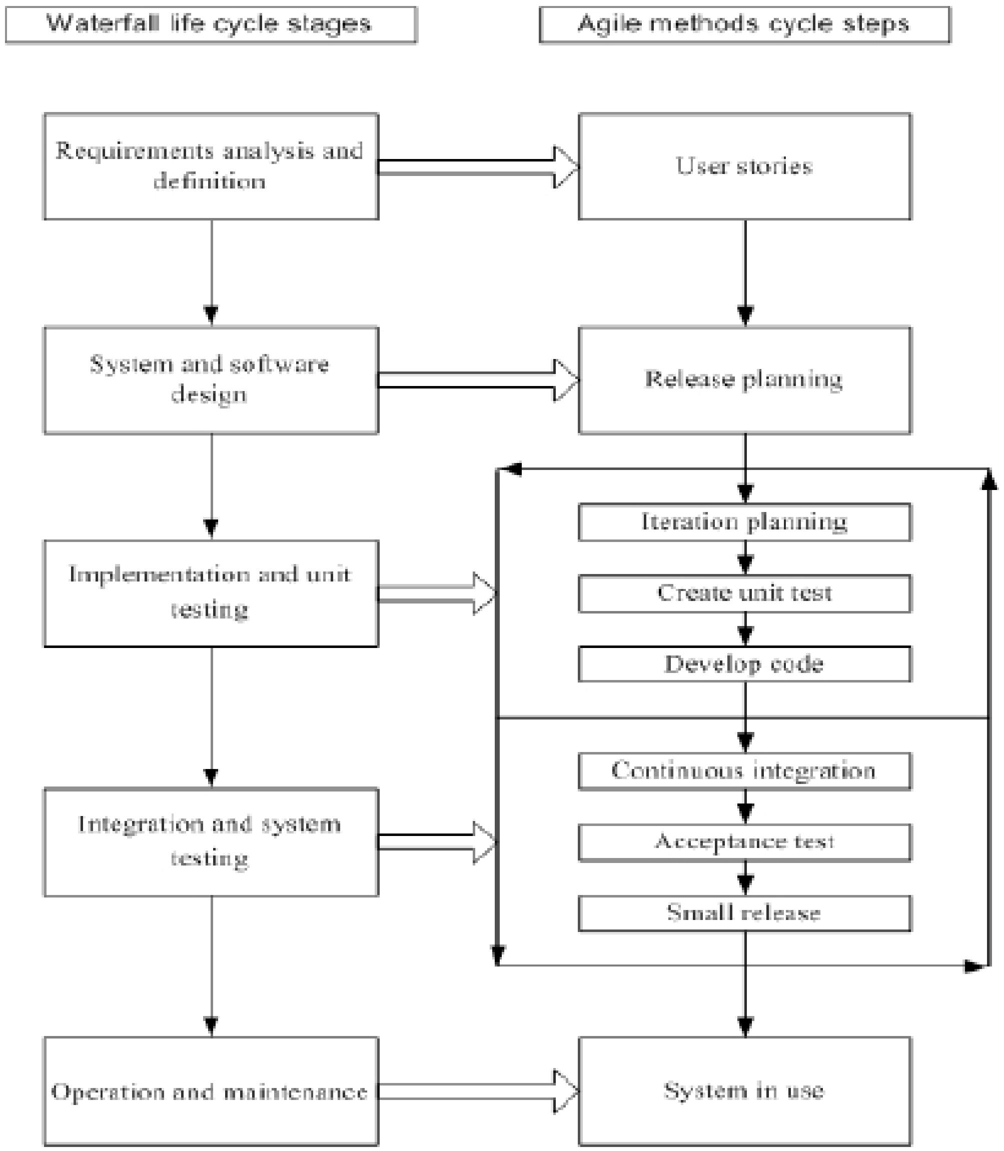
Rational Unified Process (RUP) (Kruchten, 2000)

Agile manifesto (Beck et al., 2001)

Pragmatic Programming (PP) (Hunt and Thomas, 2000)

Feature-Driven Development (FDD) (Palmer and Felsing, 2002)

Agile Modeling (AM) (Ambler, 2002)


Waterfall life cycle stages
Agile methods cycle steps
Requirements analysis and definition
User stories
System and software design
Release planning
Implementation and unit testing
Iteration planning
Create unit test
Develop code
Integration and system testing
Continuous integration
Acceptance test
Small release
Operation and maintenance
System in use

Agile Definition

- Agile *[Merriam-Webster]*
  - "1: marked by ready ability to move with quick easy grace;"
  - "2: having a quick resourceful and adaptable character."
- In agile software development,
  - "the ability to respond to change."

Characteristics of Agility(I)

- Empowered, self-organizing teams
- Multi-discipline, cross-functional teams (whole team culture)
- Project- and product-centric focus, minimal organizational focus
- Shared responsibility, role-based accountability
- Shared vision of standards of excellence
- Close, continuous collaboration, direct communication

Characteristics of Agility(2)

- **Early, frequent, and continuous demonstration** of progress through concrete deliverables
- **Rapid feedback, reflection, learning, adjustment**
- **Small work batch sizes**, minimal specialization, reduced queuing delays
- **Just in time production**, minimize production of artifacts not immediately (or ever) consumed
- **Low friction** – simplicity, minimalism, pragmatism
- **Avoidance of "debt**," focus on forward movement
- **Parallelism** and **opportunistic** control
- **Sustainable**, **constant**, **predictable pace**

Goals & Potential Benefits of Agility

- Delivering the most value to the business, efficient use of resources, maximize ROI and time-to-ROI
- Faster development, higher productivity
- Flexibility to respond to change and leverage learning
- Better quality solutions, more enduring systems
- More fulfilling development culture

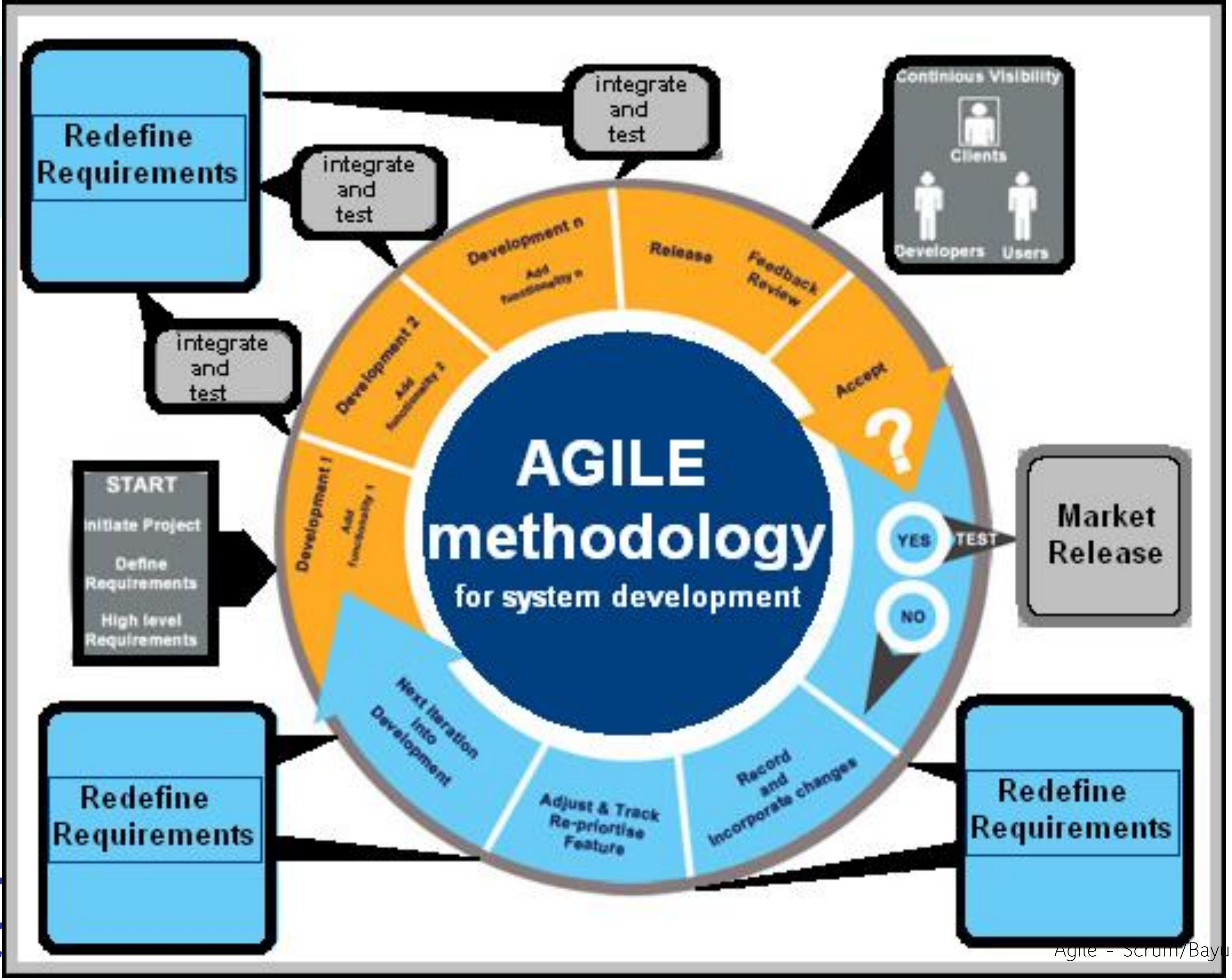
Agile Software Development Methodology

- Agile Methodology is develop based on iterative and incremental development
- It is based on feedback from the clients.
- In this methodology the requirements and solutions evolve through collaboration between self-organizing, cross-functional teams who work in close liaison with the clients.
- It promotes evolutionary development, adaptive planning and encourages rapid and flexible response to change.
- It is suitable for
  - Product (and less for services)
  - Smaller to medium sized products
- Development teams for such products is ranging from 5 to 20-20 members
  - A team of 100 members was also noted to be a success

Agile Manifesto

- **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 value in the items on the right, we value the items on the left more."


Redefine Requirements
integrate and test
integrate and test
Continious Visibility
Clients
Developers
Users
integrate and test
START
initiate Project
Define Requirements
High level Requirements
Development 1
Add functionality 1
Development 2
Add functionality 2
Development n
Add functionality n
Release
Feedback Review
Accept
?
AGILE
methodology
for system development
YES
TEST
NO
Market Release
Next iteration into Development
Adjust & Track Re-prioritise Feature
Record and incorporate changes
Redefine Requirements
Redefine Requirements

Core Principles

- “Fits just right” process
- Continuous testing and validation
- Consistent team collaboration
- Rapid response to change
- Ongoing customer involvement
- Frequent delivery of working software

How Agile is Different

- **Focus on collaboration**:
  - Less paperwork and more conversation
  - Stakeholders actively involved
- **Focus on working software**:
  - Greater feedback makes agile projects easier to manage
  - Less documentation is required
  - Less bureaucracy
- **Agilists are generalizing specialists**:
  - Less hand offs between people
  - Less people required
  - Specialists find it difficult at first to fit into the team
- **Agile is based on practice, not theory**:
  - This is a significant change from traditional
  - You need to see how agile works in practice to truly understand it

Mythbusters

Myth

1. No Documentation
2. Undisciplined
3. No Planning
4. Not Predictable
5. Does Not Scale
6. Is a Fad
7. Silver Bullet
8. RUP isn’t agile
9. Not Fixed Price

Reality

1. Agile Documentation
2. Requires great discipline
3. Just-in-time (JIT) planning
4. Far more predictable
5. Eclipse is agile
6. It’s quickly becoming the norm
7. It requires skilled people
8. RUP is as agile as you make it
9. Agile provides stakeholders control over the budget, schedule, and scope

KNOWLEDGE & SOFTWARE ENGINEERING

Agile Methodologies

- Adaptive Software Development
- Agile Modeling
- Agile Unified Process (AUP)
- Crystal
- Dynamic Systems Development Method (DSDM)
- Essential Unified Process (EssUP)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Lean Development
- Open Unified Process (OpenUP)
- **Scrum**
- Velocity tracking

Scrum

Scrum

About Scrum

- Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time.
- It allows us to rapidly and repeatedly inspect actual working software (every two weeks to one month).
- The business sets the priorities. Our teams self-manage to determine the best way to deliver the highest priority features.
- Every two weeks to a month anyone can see real working software and decide to release it as is or continue to enhance for another iteration.

- Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time.
- It allows us to rapidly and repeatedly inspect actual working software (every two weeks to one month).
- The business sets the priorities. Our teams self-manage to determine the best way to deliver the highest priority features.
- Every two weeks to a month anyone can see real working software and decide to release it as is or continue to enhance for another iteration.

History of Scrum

- **1995:**
  - analysis of common software development processes → not suitable for empirical, unpredictable and non-repeatable processes
  - Design of a new method: Scrum by Jeff Sutherland & Ken Schwaber
  - Enhancement of Scrum by Mike Beedle & combination of Scrum with Extreme Programming

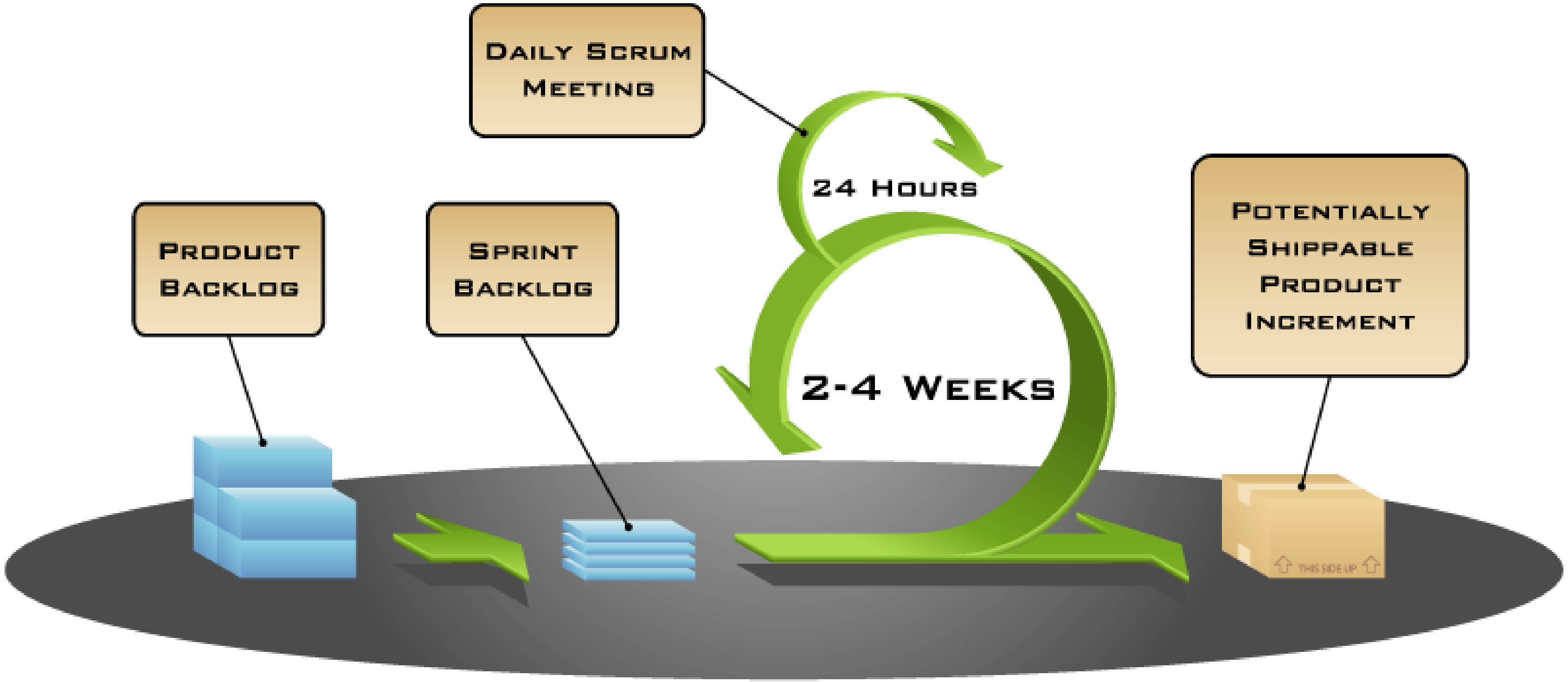
- **1996:**
  introduction of Scrum at OOPSLA conference

- **2001:**
  publication “Agile Software Development with Scrum” by Ken Schwaber & Mike Beedle

Characteristics

- Self-organizing teams
- Product progresses in a series of month-long "sprints"
- Requirements are captured as items in a list of "product backlog"
- No specific engineering practices prescribed
- Uses generative rules to create an agile environment for delivering projects
- One of the "agile processes"

How it works

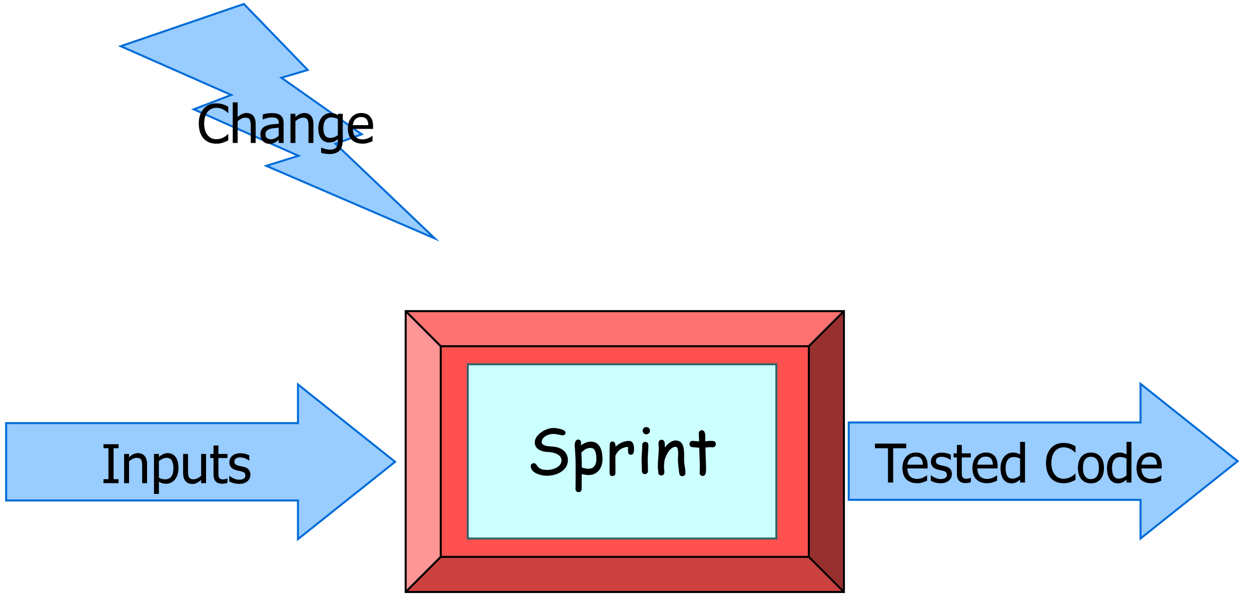


Copyright © 2005, Mountain Goat Software

Sprints

- Scrum projects make progress in a series of “sprints”
- Target duration is one month
    - +/- a week or two
        - But, a constant duration leads to a better rhythm
- Product is designed, coded, and tested during the sprint

No changes during the sprint



- Plan sprint durations around how long you can commit to keeping change out of the sprint

Scrum Framework

- **Roles** :
  - Product Owner,
  - ScrumMaster,
  - Team
- **Ceremonies** : Sprint Planning, Sprint Review, Sprint Retrospective, & Daily Scrum Meeting
- **Artifacts** : Product Backlog, Sprint Backlog, and Burndown Chart (+ Kanban Board)

Product Owner

- Define the features of the product
- Decide on release date and content
- Be responsible for the profitability of the product (ROI)
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results.

The Scrum Master

- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences

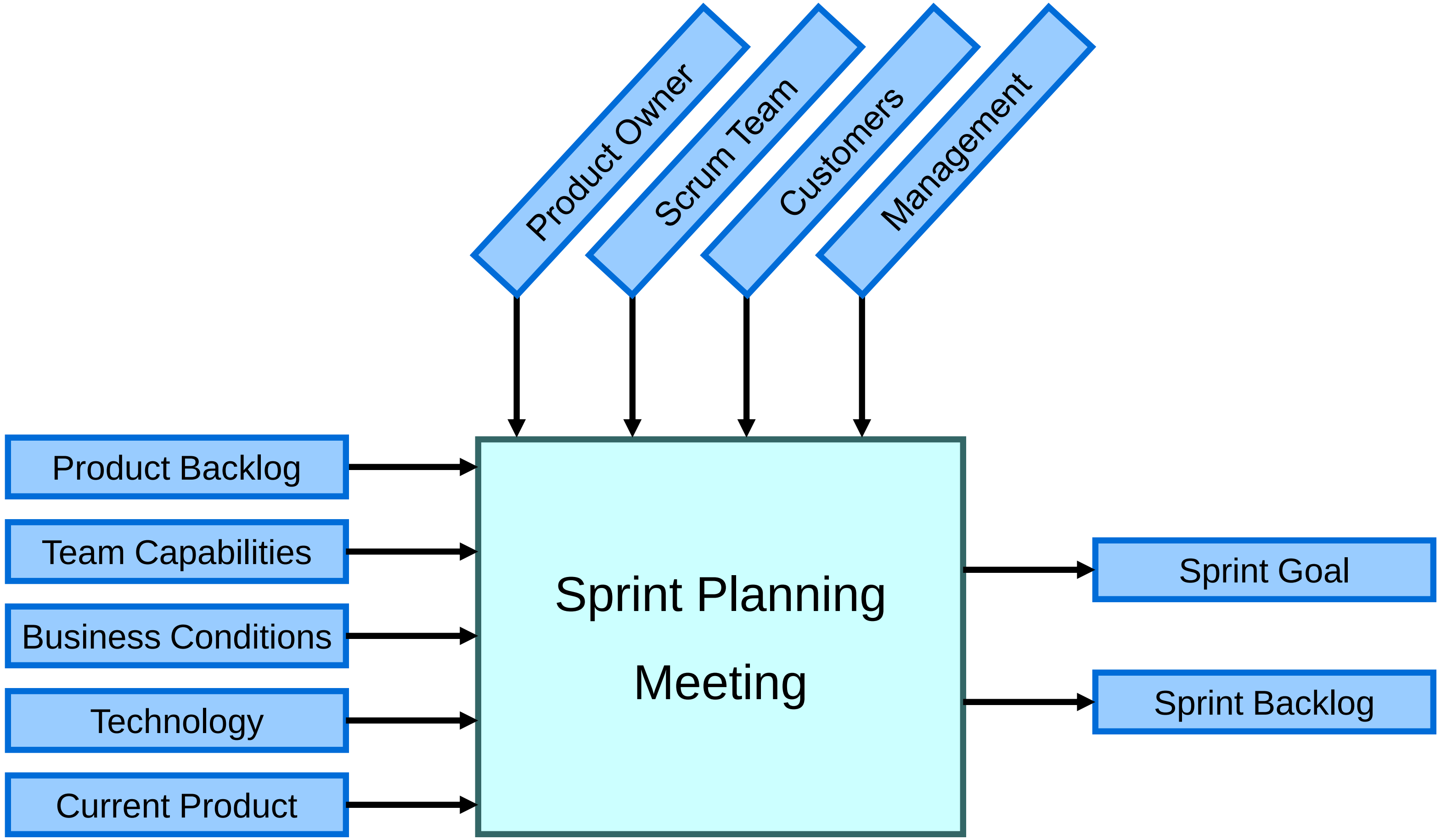
Scrum Team

- Typically 5-10 people
- Cross-functional
  - QA, Programmers, UI Designers, etc.
- Members should be full-time
  - May be exceptions (e.g., System Admin, etc.)
- Teams are self-organizing
  - What to do if a team self-organizes someone off the team??
  - Ideally, no titles but rarely a possibility
- Membership can change only between sprints

Ceremonies

- Sprint Planning Meeting
- Sprint
- Daily Scrum
- Sprint Review Meeting
- Sprint Retrospective

Spring Planning Meeting



Parts of Sprint Planning Meeting

- 1st Part:
  - Creating Product Backlog
  - Determining the Sprint Goal.
  - Participants: Product Owner, Scrum Master, Scrum Team
- 2nd Part:
  - Participants: Scrum Master, Scrum Team
  - Creating Sprint Backlog

Pre-Project/Kickoff Meeting

- A special form of Sprint Planning Meeting
- Meeting before the begin of the Project

Sprint

- A month-long iteration, during which is incremented a product functionality
- NO outside influence can interfere with the Scrum team during the Sprint
- Each Sprint begins with the Daily Scrum Meeting

Daily Scrum

- Parameters
  - Daily
  - 15-minutes
  - Stand-up
  - Not for problem solving
- Three questions:
  1. What did you do yesterday
  2. What will you do today?
  3. What obstacles are in your way?

Daily Scrum

- Is NOT a problem solving session
- Is NOT a way to collect information about WHO is behind the schedule
- Is a meeting in which team members make commitments to each other and to the Scrum Master
- Is a good way for a Scrum Master to track the progress of the Team

Scrum FAQs

- Why daily?
  - "How does a project get to be a year late?"
    - "One day at a time."
      - Fred Brooks, The Mythical Man-Month.
- Can Scrum meetings be replaced by emailed status reports?
  - No
    - Entire team sees the whole picture every day
    - Create peer pressure to do what you say you'll do

Sprint Review Meeting

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
  - 2-hour prep time rule
- Participants
  - Customers
  - Management
  - Product Owner
  - Other engineers

Sprint Retrospective Meeting

- Scrum Team only
- Feedback meeting
- Three questions
  - Start
  - Stop
  - Continue
- Don’t skip for the first 5-6 sprints!!!

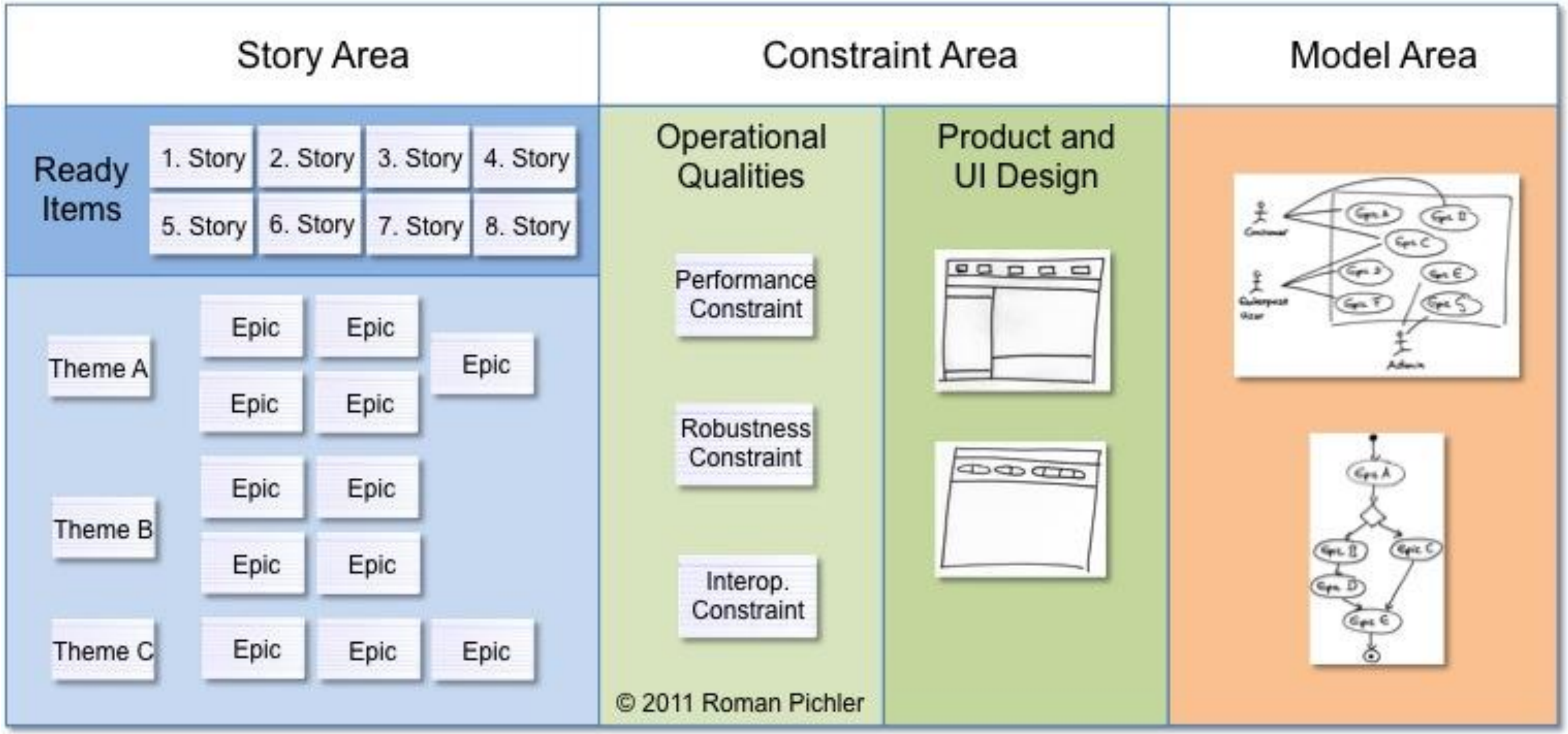
Product Backlog

- A list of all desired work on the project
  - Usually a combination of
    - story-based work (“let user search and replace”)
    - task-based work (“improve exception handling”)
- List is prioritized by the Product Owner
  - Typically a Product Manager, Marketing, Internal Customer, etc.

Product Backlog

- Requirements for a system, expressed as a prioritized list of Backlog Items
- Is managed and owned by a Product Owner
- Spreadsheet (typically)
- Usually is created during the Sprint Planning Meeting
- Can be changed and re-prioritized before each PM

An example of Product Backlog



Story Area

Ready Items

1. Story
2. Story
3. Story
4. Story
5. Story
6. Story
7. Story
8. Story

Theme A: Epic, Epic, Epic, Epic, Epic

Theme B: Epic, Epic, Epic, Epic

Theme C: Epic, Epic, Epic

Constraint Area

Operational Qualities

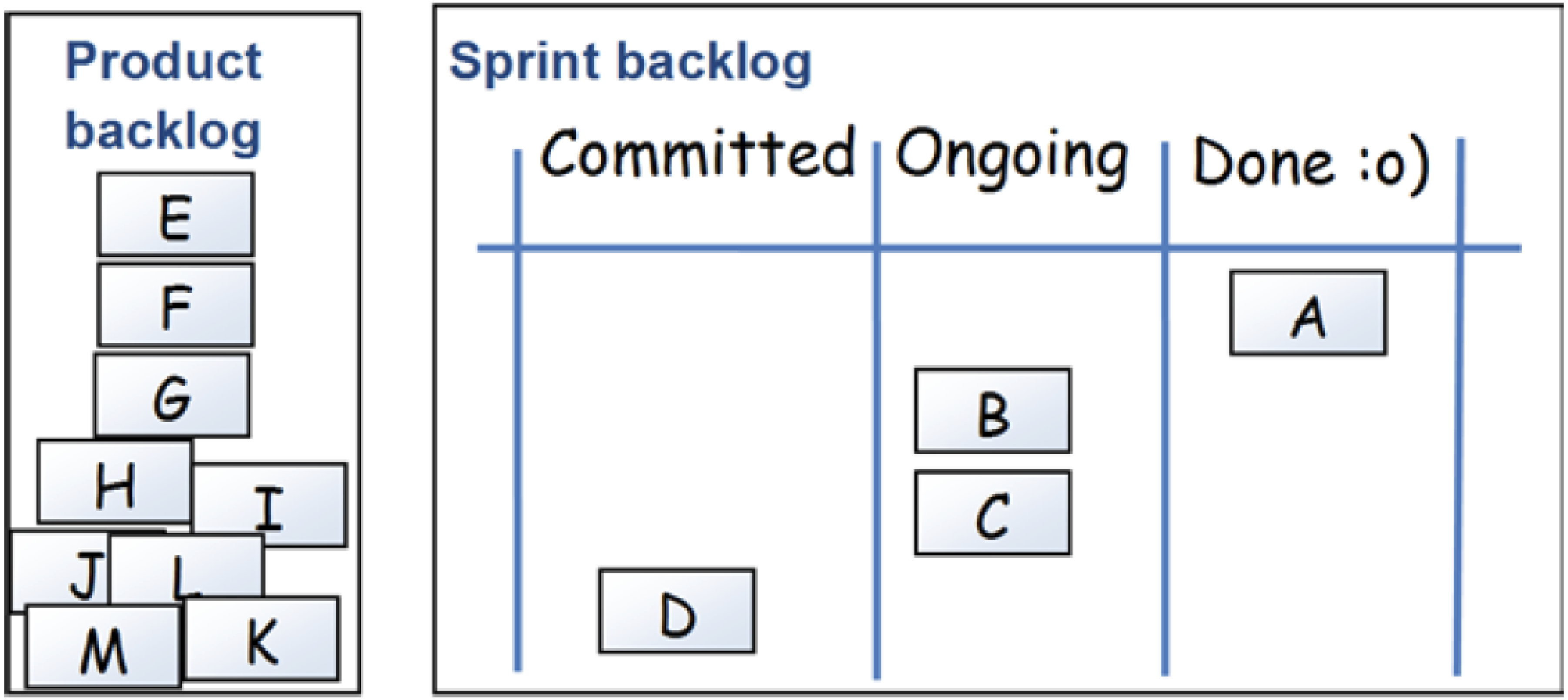
- Performance Constraint
- Robustness Constraint
- Interop. Constraint

Product and UI Design

Model Area

© 2011 Roman Pichler

Product Backlog to Sprint Backlog



Product backlog

E

F

G

H

I

J

L

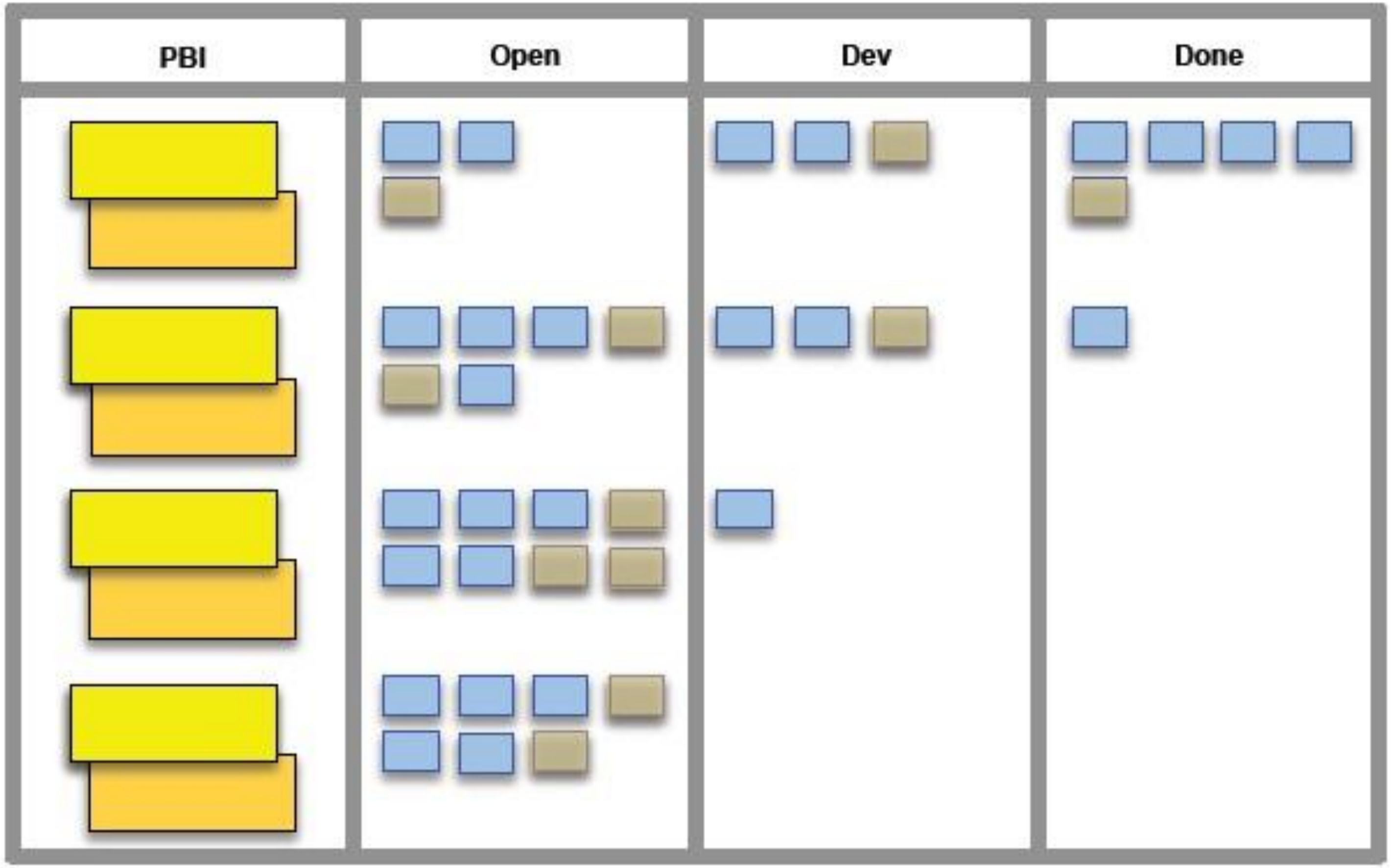
M

K

Sprint backlog

Committed	Ongoing	Done :o)
		A
	B	
	C	
D		

Kanban Board



From Sprint Goal to Sprint Backlog

- Scrum team takes the Sprint Goal and decides what tasks are necessary
- Team self-organizes around how they'll meet the Sprint Goal
  - Manager doesn't assign tasks to individuals
- Managers don't make decisions for the team
- Sprint Backlog is created

Sprint Backlog during the Sprint

- Changes
  - Team adds new tasks whenever they need to in order to meet the Sprint Goal
  - Team can remove unnecessary tasks
  - But: Sprint Backlog can only be updated by the team
- Estimates are updated whenever there’s new information

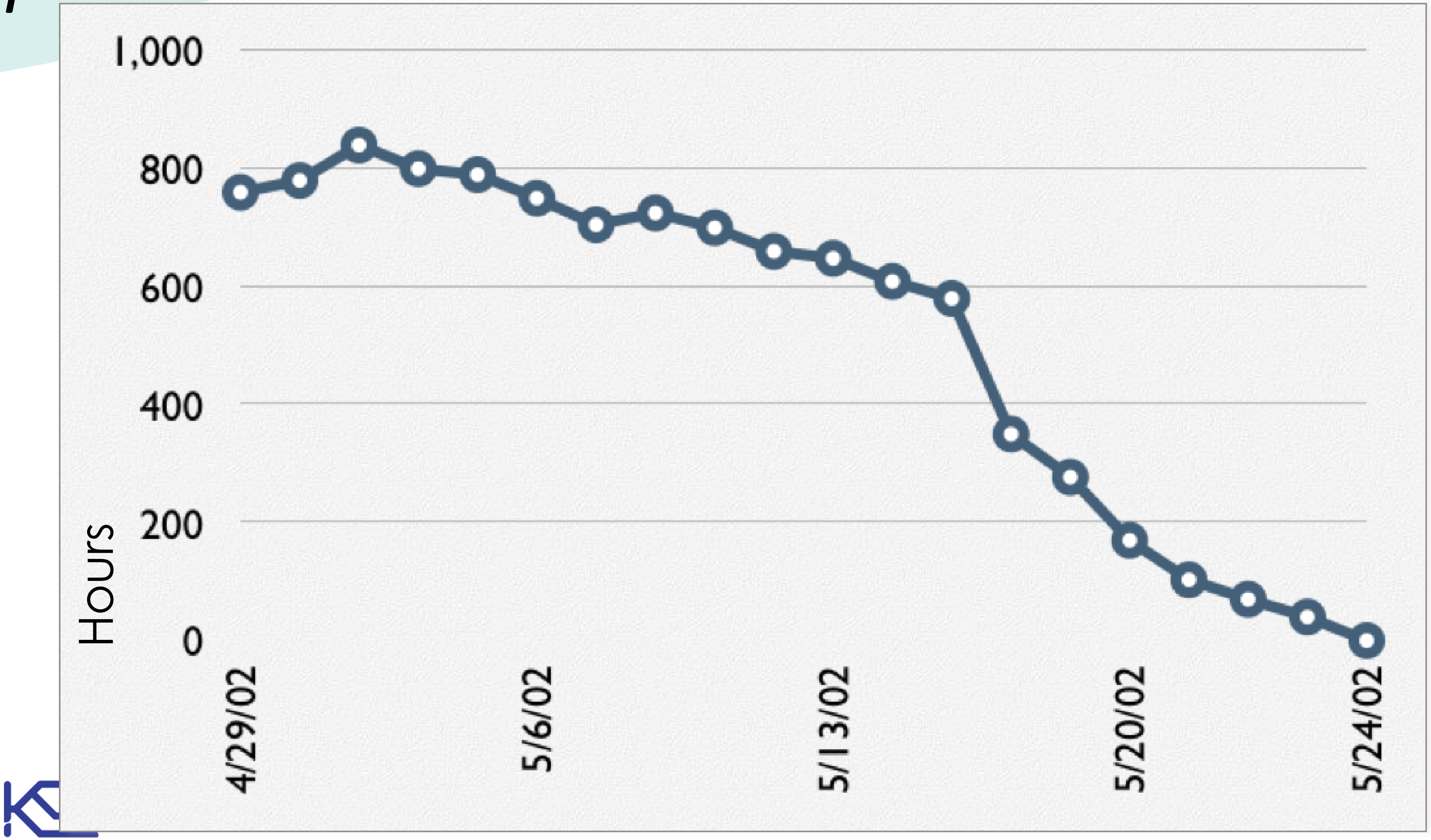
Sprint Backlog

- A subset of Product Backlog Items, which define the work for a Sprint
- Is created ONLY by Team members
- Each Item has it’s own status
- Should be updated every day
- No more than 300 tasks in the list
- If a task requires more than 16 hours, it should be broken down
- Team can add or subtract items from the list. Product Owner is not allowed to do it

A sprint backlog

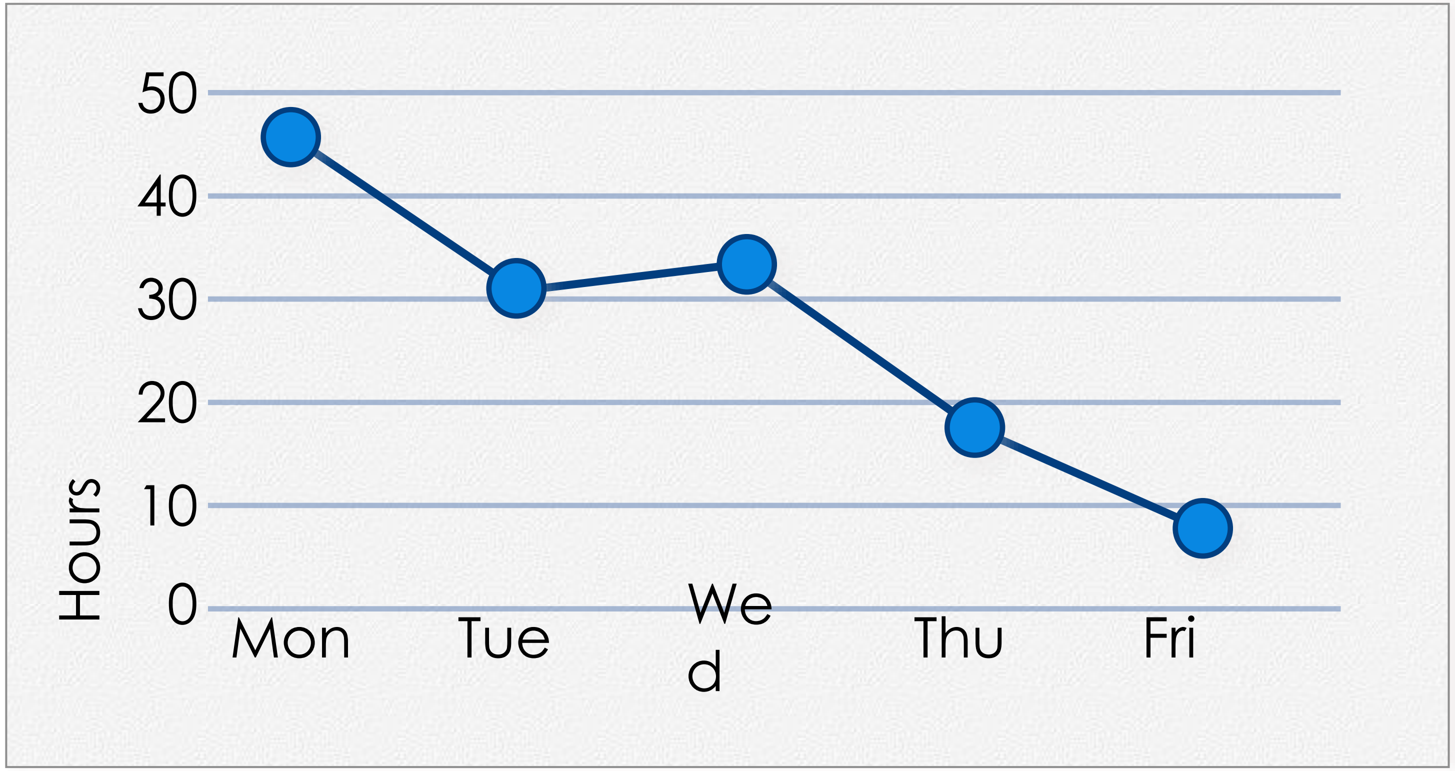
Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	4	
Test the middle tier	8	16	16	11	8
Write online help	12				
Write the foo class	8	8	8	8	8
Add error logging			8	4	

A sprint burndown chart

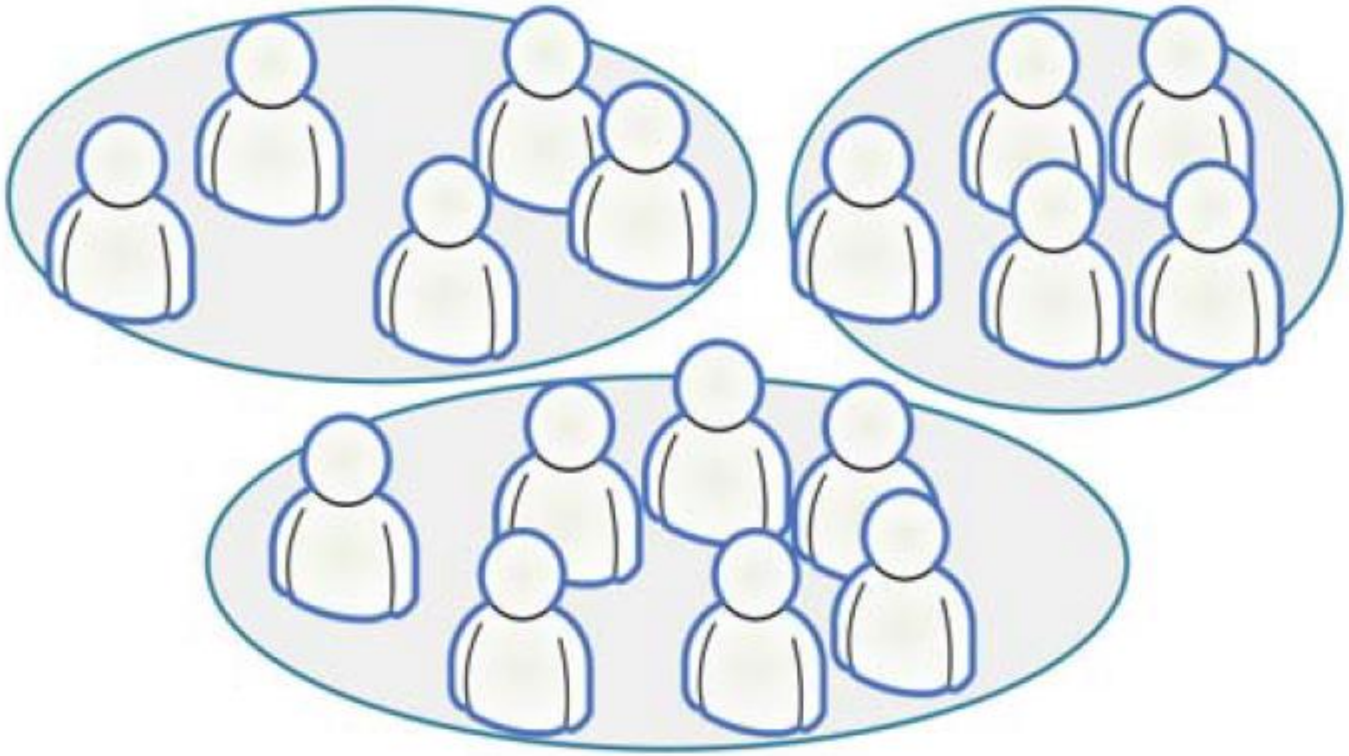


KNOWLEDGE & SOFTWARE ENGINEERING

Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	7	
Test the middle tier	8	16	16	11	8
Write online help	12				

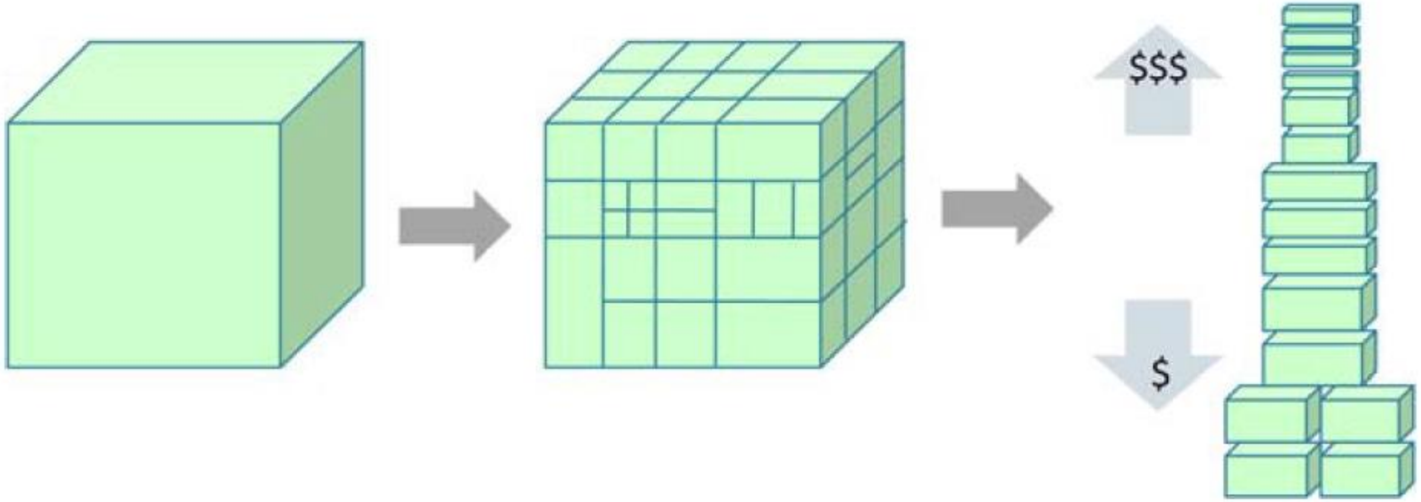


Split your organization to small-cross functional, self-organizing teams



Split your work into a list of small, concrete deliverables.

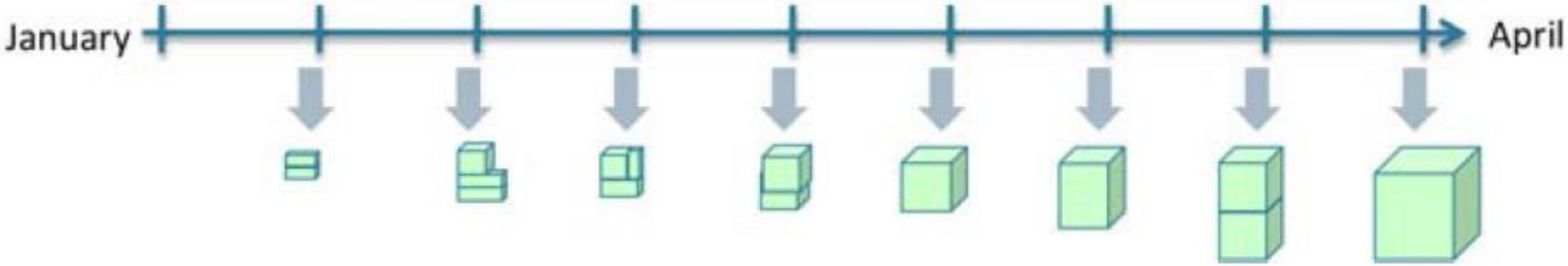
Sort the list by priority and estimate the relative effort of each ite



Split time

- Split time into short fixed-length iterations (usually 1 – 4 weeks), with potentially shippable code demonstrated after each iteration.

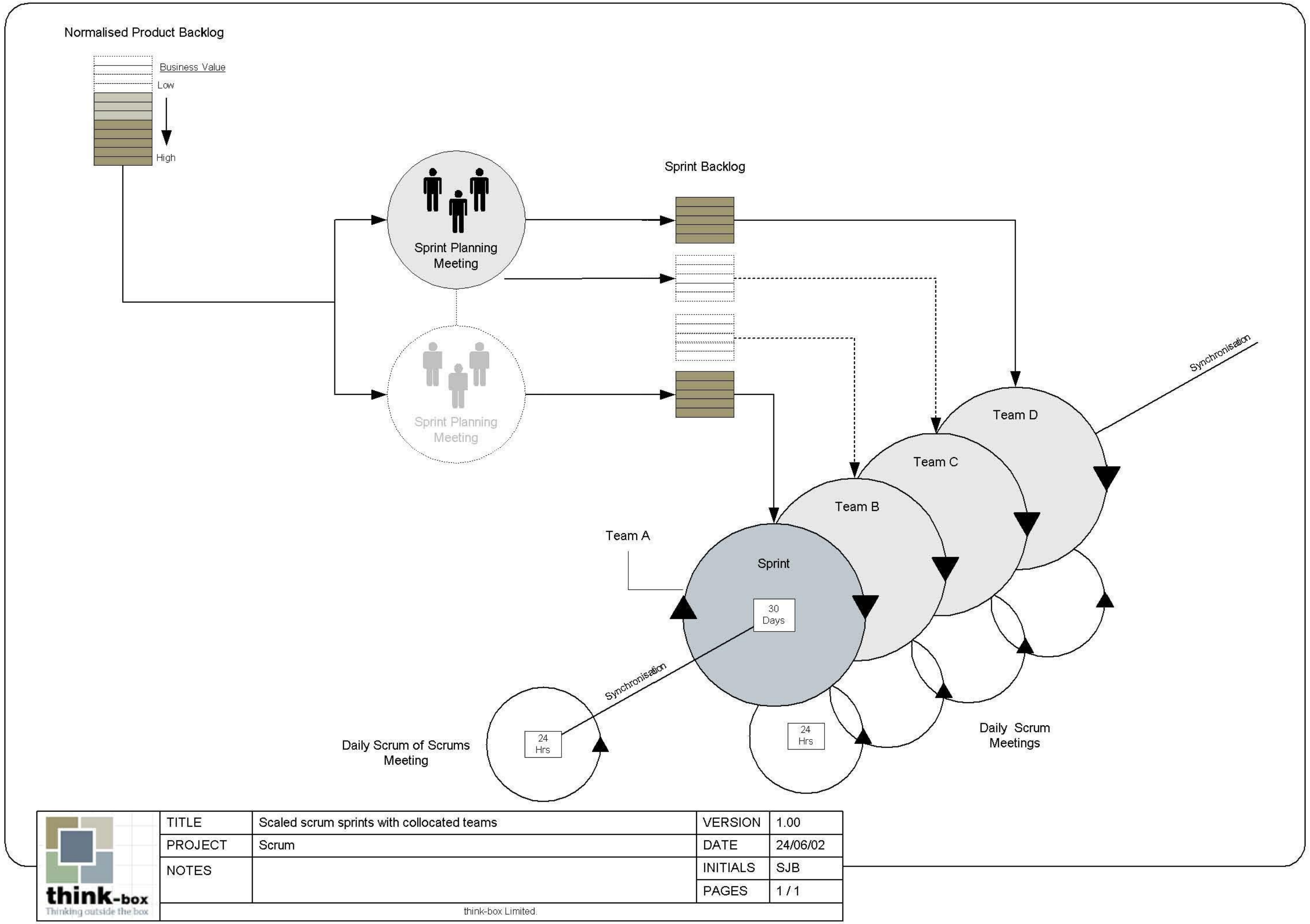
- Optimize the release plan and update priorities in collaboration with the customer, based on insights gained by inspecting the release after each iteration.
- Optimize the process by having a retrospective after each iteration.



Scalability of Scrum

- A typical Scrum team is 6-10 people
- Jeff Sutherland - up to over 800 people
- "Scrum of Scrums" or what called "Meta-Scrum“
- Frequency of meetings is based on the degree of coupling between packets

Exection of Scrum



TITLE	Scaled scrum sprints with collocated teams	VERSION	1.00
PROJECT	Scrum	DATE	24/06/02
NOTES		INITIALS	SJB
		PAGES	1 / 1

think-box Limited.

KNOWLEDGE & SOFTWARE

Advantages/Disadvantages

- Advantages
  - *Completely developed and tested features in short iterations*
  - *Simplicity of the process*
  - *Clearly defined rules*
  - *Increasing productivity*
  - *Self-organizing*
  - *each team member carries a lot of responsibility*
  - *Improved communication*
  - *Combination with Extreme Programming*

- Drawbacks
  - *“Undisciplined hacking” (no written documentation)*
  - *Violation of responsibility*
  - *Current mainly carried by the inventors*

Finally, …

Scrum is a software *management* process,
NOT a software engineering process.

Acknowledgement

- Presentation slide of Agile Logic, “Extreme Programming”, http://www.agilelogic.com/ resources.html
- Presentation Slides of Daniel Baranowski, “Extreme Programming”.
- Presentation Slides of Scott W. Ambler, “Agile Software Development: What’s Really Going On”, IBM Software group,
- Presentation slide of Yogi Bera, “Software Development Life Cycle (SDLC)”