LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

# Agile Software Development with Scrum

## An Iterative, Empirical and Incremental Framework for Completing Complex Projects

(Slides by Prof. Dr. Matthias Hölzl, based on material from Dr. Philip Mayer with input from Dr. Andreas Schroeder and Dr. Annabelle Klarl)

## CHAOS Report 2009

Completion of projects:

- 32% success
- 44% challenged
- 24% impaired

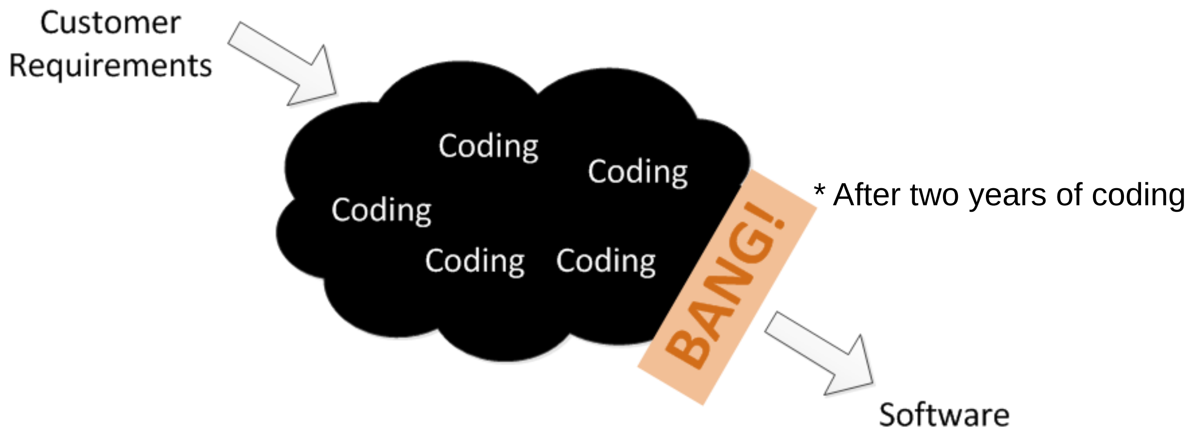
2/3 of all projects fail partially

Some of the reasons for failure:

- Incomplete requirements
- Changing requirements
- Little involvement of the customer
- Low support by the management

## Sequential paradigm

- The **Big Bang** approach to software does **NOT** work:

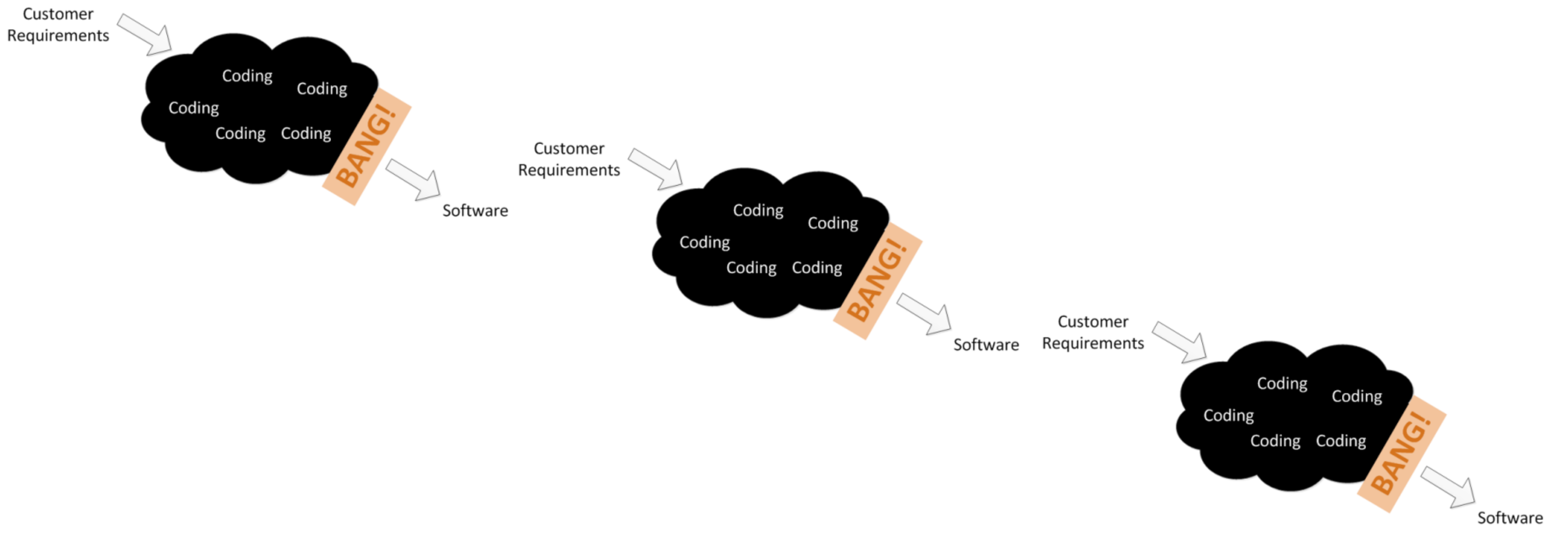


- No interaction with the customer in the black cloud!
- **The problem**:
  - Requirements might have been misunderstood or changed.
  - The resulting system is not what the customer wanted.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

## Iterative paradigm

- The **iterative** approach to software does **not** work **either**:



- Requirements are captured while product is unknown.
- Requirements Phase is drawn out until no time for implementation is left.

## Agile paradigm

**Change is the only constant in SW development**

- "Expect the unexpected!"
  Agile methods build on the ability to react to change.
- "Get it working!"
  Agile methods deliver working software frequently.
- "Please the customer!"
  Agile methods build on openness and communication.

## The Agile Manifesto

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.

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

## Why agile software development?

**Software development is like new product development, not like manufacturing**

- Manufacturing: building the same model again and again
- Software development: creating something new

We need:

- Research and Learning
- Creativity
- Subtle Control and Self-organization

## Examples for agile methods

A multitude of agile processes has been introduced

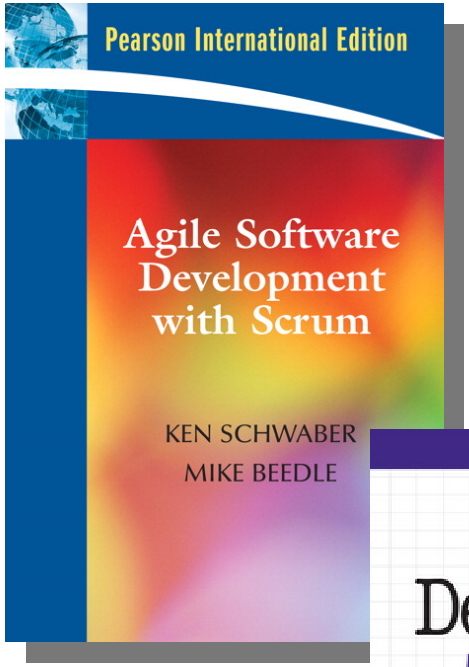
- Kanban www.kanbanblog.com/explained
- XP (eXtreme programming) www.extremeprogramming.org
- Scrum www.scrum.org
- …

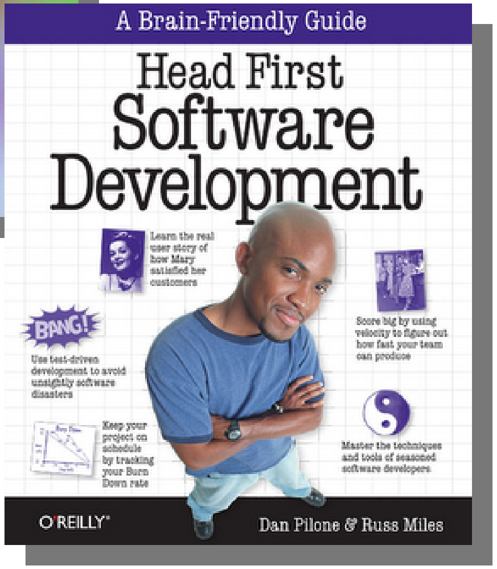
Here >>

# Agile methods in SWEP

- Scrum (mainly)
- XP
- Head First Software Development

- The Scrum process
  - follows the agile manifesto
  - is intended for groups of up to seven people
  - consists of simple rules and is thus easy to learn


Pearson International Edition
Agile Software Development with Scrum
KEN SCHWABER
MIKE BEEDLE



## Agenda

LMU

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

# Agenda

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

## The source of Scrum

**Scrum in rugby**

Strategy for getting the ball back into play

www.andrewgoss.net/sport.html

**Scrum as an agile method**

„a holistic or „rugby“ approach – where a team tries to go the distance as a unit, passing the ball back and forth“

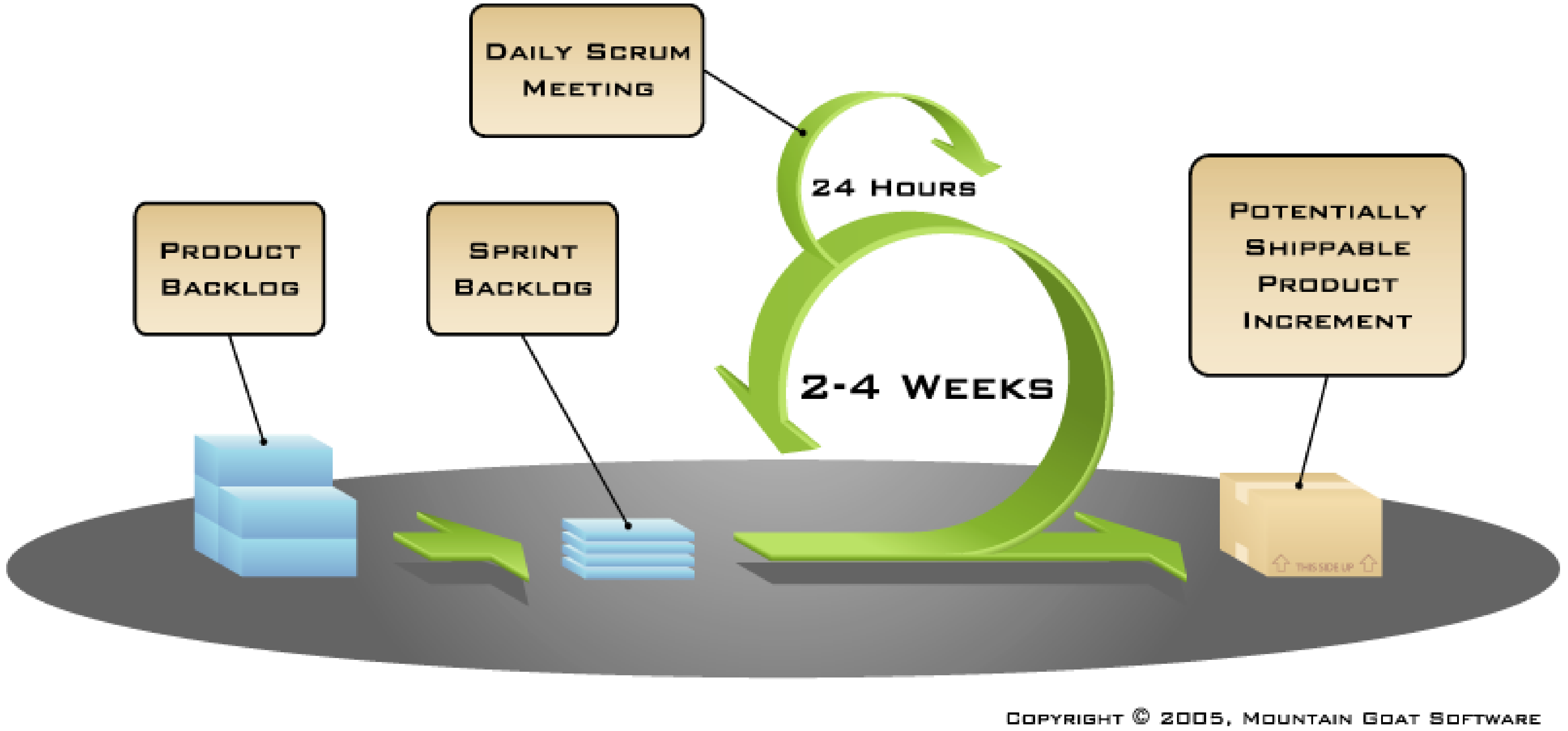
Takeuchi, H. & Nonaka, I. The new new product development game. *Harvard Business Review* **64**, 137-146 (1986).

## The three legs of Scrum

Scrum is grounded in **empirical process control theory** and is therefore not guided by a fixed project plan, but by

- **Transparency**
  „Everything can be seen by everybody.“
- **Inspection**
  „The process is continuously monitored.“
- **Adaption**
  „Feedback mechanisms are the heart of Scrum.“

## The big picture


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
This side up
Copyright © 2005, Mountain Goat Software

An iterative, empirical and incremental framework

## Product Backlog and Sprints

- **Product Backlog**: **Everything** with respect to the product or the process, that **anyone** is interested in, is represented in the Product Backlog.
- **Sprints**: A Sprint is a short timeframe of about four weeks for working on the Sprint Backlog - a **fixed** subset of the Product Backlog, producing a **working piece of software**.
- **Daily Scrum Meetings**: The current **progress** of work and any **impediments** are revealed in this daily time-boxed meetings.

## Summary

**Software development is about <u>shipping</u> software that brings the <u>customer's ideas</u> to life.**

Which means:

- **Shipping software**: The software must be completed, executable and delivered – **on time** and **on budget**.
- **Customer's Ideas**: The customer has a vision of his product. The developer must be flexible enough to extract that image, **implement it** and nevertheless **react to changes**.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

## Agenda

## About pigs and chickens

A chicken and a pig are together when the chicken says,
**„Let‘s start a restaurant!“**

The pig thinks it over and says,
**„What would we call this restaurant?“**

The chicken says, **„Ham n‘ Eggs!“**

The pig says,

**„No, thanks. I‘d be committed, but you‘d only be involved!“**

- Pigs: Everyone with total commitment to the project
- Chickens: Everyone else who is interested in the project

## Scrum Master

The Scrum Master is **responsible for the success** of Scrum.

- Responsibilities:
  - Introduces/Briefs the Product Owner
  - Forms a Scrum Team
  - Assists all Planning Meetings
  - Ensures that **Scrum values, practices and rules are enforced**
  - **Removes any impediments**
- Although being a management role, the Scrum Master should be **your friend in need**.
  - In SWEP, the Scrum master is one of the team members.

## Product Owner

The Product Owner is officially **responsible for the product**.

- Responsibilities:
  - Represents the customer
  - Maintains the **Product Backlog**
    - Registers new Items
    - Prioritizes Items
    - Get the estimates for Items
  - Makes the Product Backlog visible to everyone
- **Developers only listen to the Product Owner**
  - (regarding priorities and user stories)
- In the SWEP, the PO is a tutor.

Best if it is a person employed by the customer BUT WHO IS ALSO knowledgeable in being a "SCRUM product owner"

## Scrum Team (I)

The Scrum Team **commits to achieving a Sprint Goal**.

A common overarching "theme" of the sprint, according to which user stories are selected

- 7 (+-2) developers
  - < 5 developers impose skill constraints
  - > 9 developers induce complex coordination
- Responsibilities:
  - Decides on a Sprint Goal in compliance with the Scrum Master and the Product Owner
  - Commits to turn the selected set of the Product Backlog into a working product during a Sprint
  - Has **full authority how** to achieve the Sprint Goal

## Scrum Team (II)

"We swim and sink together"

- The whole team is responsible for the whole product
- Normally full time members
- No titles
- All-round developers
  – or at least willing to assist each other

Best not to have specialists because they may leave, get sick, must work on another thing, …

Team composition may only change at the **end** of a Sprint, but experts can be invited to assist the development!

## Chickens

- Everyone else who is interested in the project
  - **Customer** (except PO)
  - Management
  - Other Scrum Teams (working on the same project, depending projects or totally different projects)
  - Observers

- Chickens are **not allowed to influence** the work of the Scrum Team **during a Sprint**!

## Summary

- Pigs are **committed** to the project.
  - The Scrum Master enforces the Scrum rules.
  - The Product Owner manages the Product Backlog.
  - The Scrum Team is committed to the Sprint Goal.

- Chickens are only **involved** into the project.
  - Scrum Teams must not listen to chickens.
  - Chickens are only allowed to consult.

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

## Agenda

# Release Planning Meeting

- The **Product Vision** is the customer’s mental image about his software.
- The Release Planning Meeting plans the various releases over the course of the next year(s)
- The goal of the Release Planning Meeting is „How can we turn this vision into a winning product?“
  - Overall features and functionalities
  - Major risks
  - Probable delivery date and cost
- But how do we extract the correct requirements from the Product Vision?

## Product Backlog Items

- In Scrum, requirements are captured in the form of **Product Backlog Items** (PBI).
- For the SWEP, we use **User Stories** and **Issues** as PBIs.
  - A User Story captures **one thing** (and one thing only) that the software needs to do for the customer.
    - A User Story has a **title** and a **short description**
    - The description should fit on a DIN A6 index card (if it is too long, it needs to be split in two)
  - An Issue captures **one thing** that is hard to mold into a User Story e.g. software quality issues like bugs and safety, security or performance as well as documentation matters.

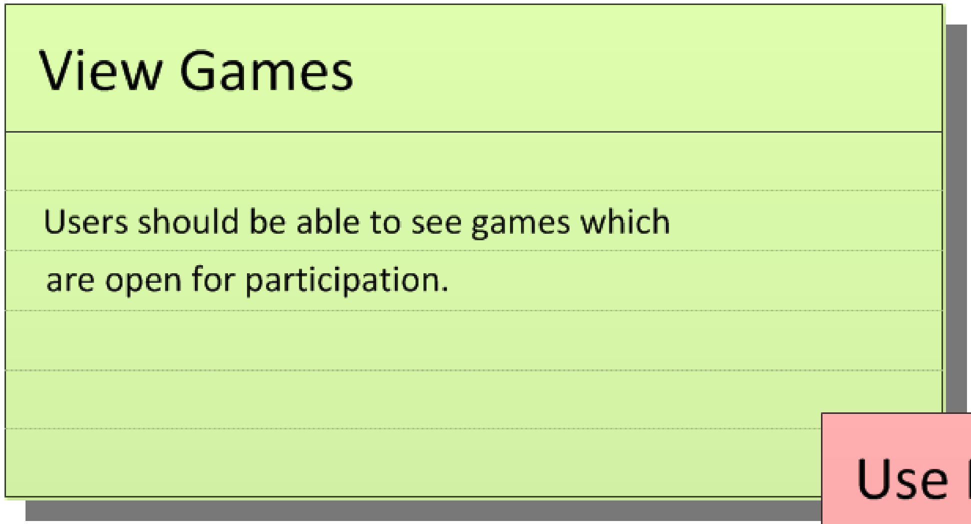
## Capturing User Stories

- User Stories are **customer-oriented**.
  - User Stories are written with and for the customer
  - They must be written in a language the customer can understand

- Techniques for capturing requirements
  - Blueskying: brainstorming with the customer
  - Role playing: developer acts as the new software
  - Observation: developer watches the customer do the tasks to be supported by the new software

## Examples

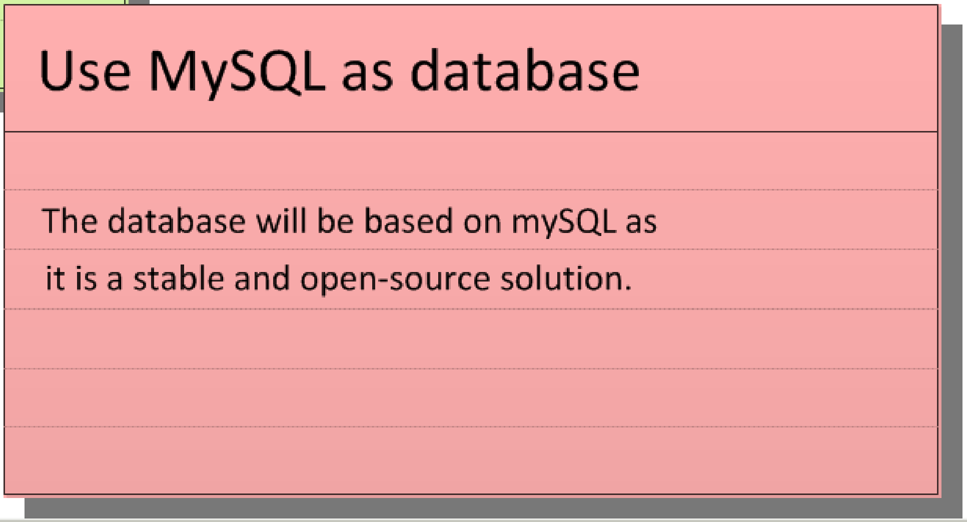
- Good Story (customer-level):



View Games

Users should be able to see games which are open for participation.

- Bad Story (too technical):



Use MySQL as database

The database will be based on mySQL as it is a stable and open-source solution.

## Product Backlog

- All User Stories and Issues make up the Product Backlog.
- The Product Backlog is **never** complete!
  - **Everyone** (pigs and chickens) may add items.
  - The Product Backlog evolves during the project by adding or changing requirements.
- User Stories and Issues get
  - …a priority (by the PO)
  - …a time estimate (by the Scrum team)

## Prioritizing

The Product owner **prioritizes** the Product Backlog Items in compliance **with the customer**.

- Important ones get a higher priority and must be implemented first.
  - Priorities should be taken out of the set of **{10,20,30,40,50}** with 10 being most important .
  - Priorities are added to the Product Backlog Items.

- Priorities for Product Backlog Items **might change** depending on estimates or changing requirements.

## Estimating

- Time required for implementing User Stories or Issues are **estimated** by the **Scrum team**
  - ideally in separate sessions before release or sprint planning
  - without PO (bias); except for clarifications.
- Estimation means guessing the number of hours for constructing each PBI.
  - **How long** will it take it get it done?
- User Stories and Issues may be split into tasks for estimation (but we don’t do this in the SWEP).

# Estimating (II)

- The whole Scrum Team is responsible for the project.
- Everybody should, in principle, be able to implement each functionality. Thus, estimation takes **everybody** into account!
- Each estimate should include time for
  - Design
  - Code and Document
  - Test and Review
  - Integration and Delivery
- To arrive at a number **everybody is comfortable with** we use Planning Poker.

## Planning Poker (I)

- Planning Poker
  - A certain PBI is presented.
  - Every developer thinks about the PBI and how long it will take **himself** to implement it, all things considered.
  - Every developer **privately** chooses a card from the deck with cards for 0, ½, 1, 2, 3, 5, 8, 13, 20, 40 and 100 hours.
  - All cards are **simultaneously** uncovered.
  - High and low estimates are discussed.
  - The estimation process is repeated until convergence.

## Planning Poker (II)

Everybody must be able to implement the functionality with this estimate!

- The goal is convergence.
  - The team **must come up with a single estimate.**
  - If the estimates differ a lot, this indicates (probably) hidden assumptions and less confidence.
- Thus, a second goal is to uncover **assumptions**
  - ...about what is part of a story and what is not
  - ...about the skills required or the need to acquire them first
  - ...about the complexity of the task
- This might require asking the customer for clarification.
- And, a third goal is to **transfer knowledge.**

## Estimating (III)

Meaningful estimation requires **knowledge** about

- … the existing codebase
- … the effort involved in using the libraries and technologies

- It is borderline impossible to come up with meaningful estimates if these factors are completely unknown.
- Therefore, get familiar with the technologies **before the first sprint start (!)**

## Summary

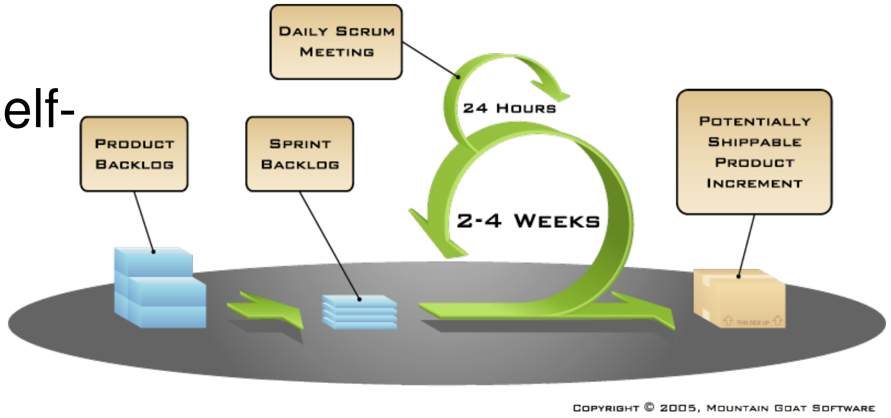
- Requirements are captured as Product Backlog Items.
  - **User Stories** are customer-oriented.
  - **Issues** capture more technical things.
  - The Product Backlog is never complete.
- The Product Owner assigns **priorities** to the Items, indicating which functionality should be implemented first.
- Product Backlog Items are **estimated.**
  - The aim is confidence by all developers
  - ...and getting rid of assumptions.

## Agenda

## Agile paradigm

**Change is the only constant in SW development**

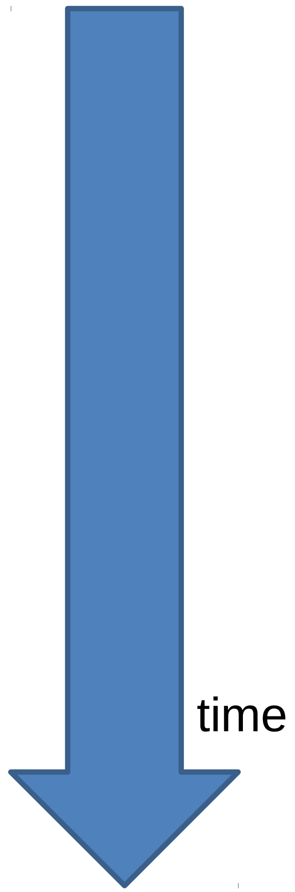
- Requirements, Estimates, and Priorities might change – but this is considered in the process and dealt with **in a controlled way**.
- **Releases**. Our process is based on releases which take about three months.
  - A release of the software is a self-contained set of functions.
- **Sprints**. Each release is split into Sprints which take about four weeks.


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software

## Sprint

- **Fixed** period of time, e.g. „four weeks“
- **Fixed** set of functionality to accomplish
  - Sprint Goal
  - Sprint Backlog
- During Sprint
  - **No interferences** with the development work
  - No additional functionality
  - No new technologies
  - **Free timing** for the Scrum Team

## Activities during a sprint



1. Sprint Planning Meeting
   - Assigning Product Backlog Items
   - Determining Velocity
2. Development work
   - Holding Daily Scrum Meetings
   - Updating Whiteboard and Burn-Down-Chart
3. Sprint Review
   - Demoing the piece of running software
4. Sprint Retrospective
   - Learning from the past

## Sprint Planning Meeting

- Fixed timeframe (to prevent dawdling)
- Participants: PO, Scrum Master, Scrum Team
- The goal is to decide
  - … **what** will be done
  - … **how** it will be done

- That basically means assigning Product Backlog Items to the Sprint as Sprint Backlog Items and planning how to realize them.
- PBIs should ideally have been prioritized and estimated before the meeting.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

## Planning User Stories

**The Sprint Planning Meeting is the main meeting for planning the Sprint!**

- Which means...
  - … pick User Stories (PBIs in general),
  - … discuss realization approach (with UML sketches),
  - … assign User Stories (PBIs in general) to team members.
- But: **How many Product Backlog Items fit into a Sprint?**

## Assigning Items: Reality

- In principle, the available days are four weeks i.e. 20 working days multiplied by the number of developers (e.g. 3):

  3 x 20 = 60 days

- **However**: Estimates are based on **ideal** days or hours. Unfortunately, the real world keeps intruding with
  - Installing Software
  - Team Communication
  - Paperwork
  - Hardware breakdowns
  - Sickness and Holidays

# Velocity (I)

- **Solution**: The amount of available days is reduced by a factor, the team velocity:

$$3 \times 20 \times 0.7 = 42 \text{ days}$$

- That means, we can select Product Backlog Items with a total estimate of 42 days for the Sprint – and not more!

- As an initial factor, a value of 0.7 is assumed.
- But the velocity is unique for each team and must therefore be monitored and changed over time…

## Velocity (II)

- How we will determine velocity
  - Track all time (incl. overhead time)
  - Compute velocity based on available data

$$V = \text{Worked} / (\text{Worked} + \text{Overhead})$$

- Reasons
  - Lab is no full-time job
  - Flexible time management
  - Empirical approach to velocity computation in our context

## Controlling the Process

- During a Sprint, the Scrum Team works on Sprint Backlog Items until they are „done“.
  - Each team has its own **Definition of Done (DoD)**.
  - Ours implies full functionality, no known errors/bugs, clean code, integration, tests, documentation.

- It is important to stay on track: If a User Story or Issue takes longer or shorter than expected, or if additional problems come up, the team must **know** about this.
- This information is gathered in Daily Scrum Meetings.

## Daily Scrum Meetings

- **Daily** at a fixed time and place: **15 minutes**
- Everybody may attend, but **only the pigs** (Scrum Team, Scrum Master and Product Owner) are allowed to speak.
- The goal is to see
  - … what was done since the last meeting
  - … what will be done before the next meeting
  - … what obstacles are in the way

- That basically means that **every** team member has to **briefly** report on these three questions.

## No Overhead!

- Some principles ensure that these meetings are productive and informative for everybody.
  - Start **sharply** at the designated time. (regardless of who is present)
  - Report **briefly** only relevant things.
  - Report on the “**what**”, not on the “how”.
  - Detailed discussion may continue **afterwards**.
  - **Stand up** during the meeting.
- The intention is to keep the finger on the pulse of the project.

## Social responsibility

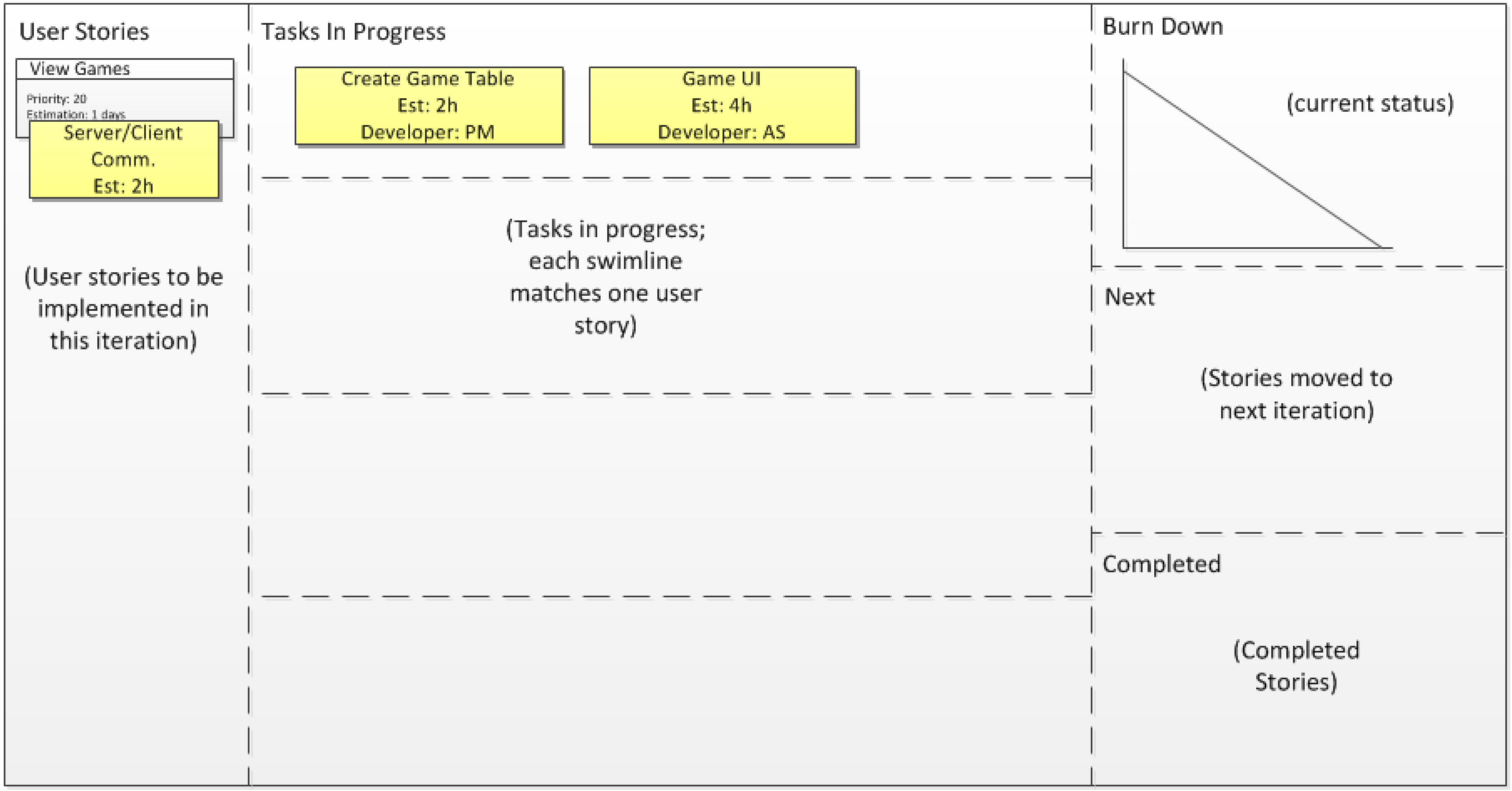
- All team members must **actively** attend.
- This enforces the **social responsibility** for everybody:
  - Honest report what has been done
  - Face-to-face promise what is done next
  - Pressure on the management to solve problems

- Scrum builds on openness and honesty!
  - **Full transparency** of all fails and delays, but also of any progress and completion
  - Only way for the Scrum Team to react to changes.

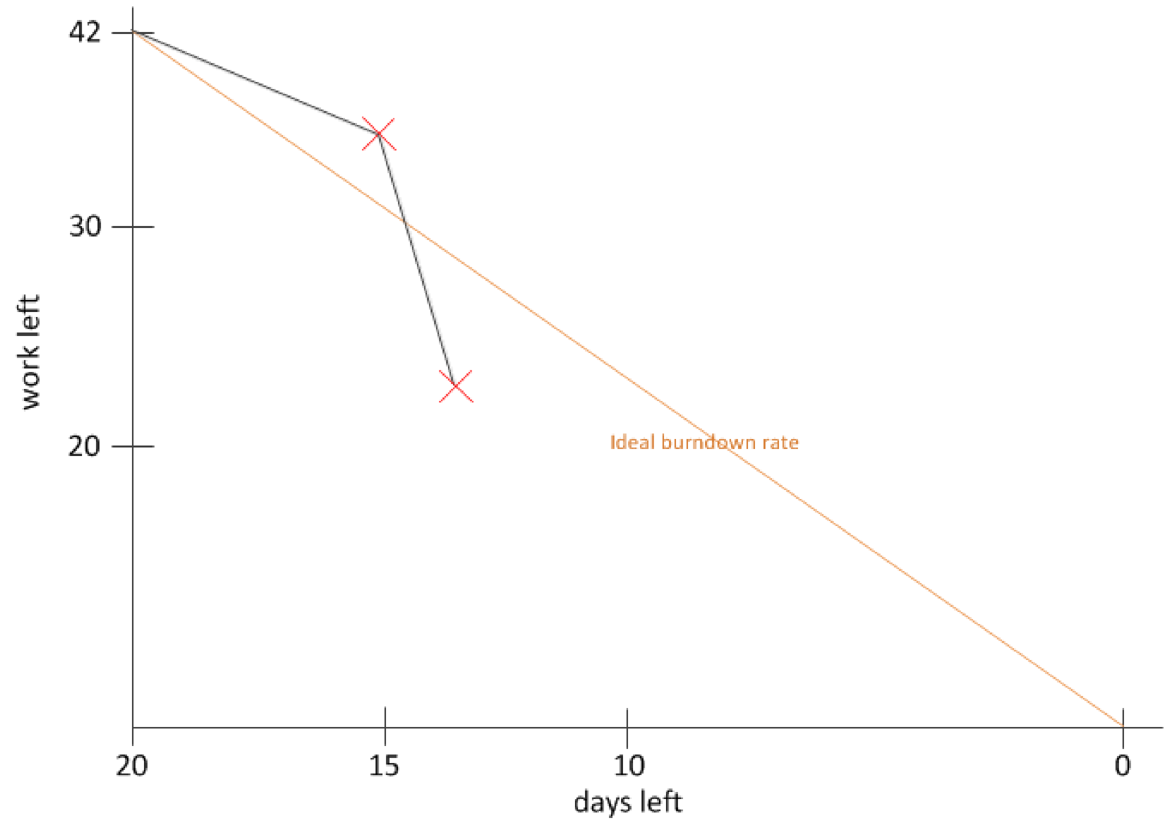
## Visualizing the progress

- The Whiteboard keeps track of the **current progress**.
- It shows
  - … which Sprint Backlog Items must be implemented during the Sprint
  - … which tasks are in progress
  - … which tasks have been completed during the Sprint
  - … how fast the development progress is compared to the plans for this Sprint
- The whiteboard should ideally only be updated during the Daily Scrum Meetings
- …but for SWEP, we update online before the Scrum meeting

# The Whiteboard


User Stories
View Games
Priority: 20
Estimation: 1 days
Server/Client Comm. Est: 2h
(User stories to be implemented in this iteration)
Tasks In Progress
Create Game Table Est: 2h Developer: PM
Game UI Est: 4h Developer: AS
(Tasks in progress; each swimline matches one user story)
Burn Down
(current status)
Next
(Stories moved to next iteration)
Completed
(Completed Stories)

## The Burn-Down Chart (I)



The Burn-Down-Chart shows the remaining work, NOT the actual required working time

## The Burn-Down-Chart (II)

- The chart shows
  - X-Axis: working days left until the end of the iteration
  - Y-Axis: sum of task estimates yet to be done
  - The straight line is the ideal burn-down rate:
    This is how the tasks are planned against the time available.
- During Daily Scrum Meetings, the current status is added:
  - The sum of the remaining task estimates are plotted on the intersection with remaining days.
  - If the point lies above the ideal burn down rate, the team is behind schedule. Else, it is ahead of schedule.
- The chart needs to be updated when tasks change their status, are added or removed or **when estimates change**.

## Problems

- The chart shows **whether the remaining workload matches the remaining time**
  - Too slow: The functionality must be reduced or some Sprint Backlog Items have to be scheduled for the next Sprint; PO must be notified
  - Too fast: May add new items from product backlog, or bugs or maintenance items; PO must be notified
  - The reason is logged for the next estimation (or velocity).
- **Unplanned tasks** may still occur (like bugs/maintenance)
  - Those can be added as new items with an estimate and a priority and are split into manageable tasks.
  - The PO does NOT have access to the sprint backlog and may not add or change tasks during the sprint; new items must be added to the product backlog and may be considered for the next sprints.

## Sprint Review

- A Sprint comes to an end when time runs out.
- At this point, a **running version of the software must be available** – if not all tasks/items were handled, these have been pushed back before.

- In the Sprint Review of about four hours,
  - … a demo is given to the customer.
  - … is summarized what went wrong and what right
  - … is discussed what was achieved
- **Note**: The Sprint Review should not be extensively prepared, often PowerPoint slides are forbidden.

## Sprint Retrospective

- In the Sprint Retrospective, we want to **learn from the past:**
  - Revisit estimates – why did they differ from the actual time? What can be done better next time?
  - Calculate the new velocity, but keep in mind that the team velocity should only account for overhead, not as a buffer for wrong estimates

    velocity = estimated days / required days
  - Revisit team composition, tools, methods of communication…
- **The next iteration** begins just like the last.

## Abnormal Sprint termination

- Very rarely, a Sprint must be cancelled earlier if
  - … a Sprint Goal becomes obsolete
    (e.g. the customer's priorities change heavily)
  - … the Sprint Goal is not achievable
    (e.g. the Scrum Team cannot manage the selected Backlog)
  - … too many impediments occur
    (e.g. the Scrum Master and the management fail in removing impediments)
- Note: Abnormal Sprint termination **consumes resources** for re-grouping and re-planning.

## Summary

- Scrum is a **Controlled Process** to stay on top of the current progress, problems and changes.
- During a Sprint,
  - … the set of functionalities to implement does not change.
  - … the  current progress is always transparent.
  - … impediments are immediately dealt with.
- Before the next Sprint,
  - … the last Sprint is reviewed to enhance productivity.
  - … priorities and estimates are re-adjusted.
  - … new ideas and functionalities are taken into consideration.

# Agenda

## More than 9 Developers

- Sometimes, we need more than 9 developers:
  - Time constraints
  - Scope of project

- HOWEVER:
  > 9 developers in one Scrum Team induce complex coordination

**How do we scale Scrum to larger projects?**

## Several Scrum Teams

To scale Scrum, we implement several Scrum Teams:

- Each Scrum Team is an own unit which means…
  - … it has its own Goals, Sprint Backlogs, Meetings
  - … it has its own team dynamics.

- The Scrum Teams collaborate in the same project by…
  - … working on the same Product Backlog.
  - … sharing their progress in „Scrum of Scrums“ Meetings.

## How to extend Scrum

- Most things are done independently:
  - …its **own Sprints (Goal and Backlog**)
  - …its own **Daily Scrum Meetings** (where the other Scrum team members may be present as chickens)
  - …its **own Sprint review** and **retrospective**
- However:
  - …results are integrated into a common product
  - …each Scrum team designates 1-2 representatives to meet in a “Scrum of Scrums” meeting to discuss dependencies
  - …these people should be those best suited to handle inter-team dependency issues

## Summary

To scale Scrum, we implement several Scrum Teams where

- … each Scrum Team pursues its own Sprint Goal,
- … but all Scrum Teams collaborate in completing the same project vision

**Minimize interactions and dependencies between teams!**

**Maximize cohesion within each team!**

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

# Agenda

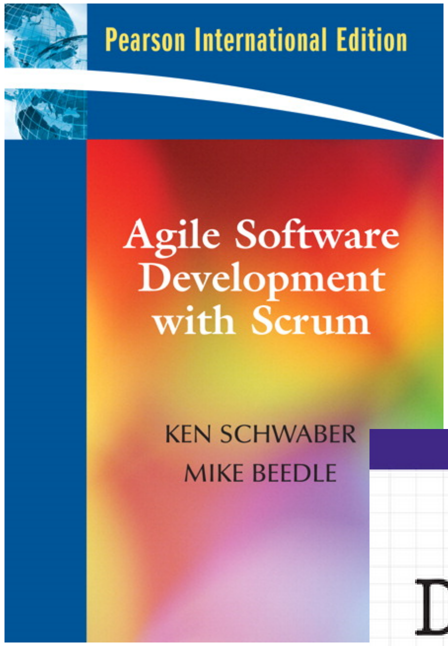
# Values in Scrum

- Commitment
  - The Scrum Team has full authority how to do the work.
  - The whole Scrum Team is responsible for the whole product.
- Openness
  - Everything is visible to everyone.
- Courage
  - Do your best, don't give up!
- Respect
  - Respect everyone's strengths and weaknesses!
  - Provide help and do your best!

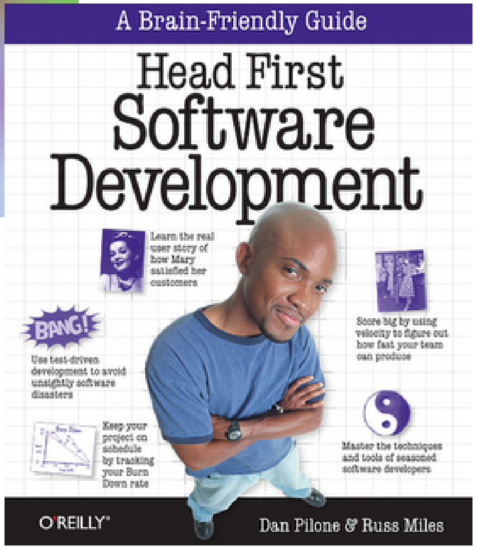
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

# The Scrum process for SWEP

- This talk has presented an agile method based on Scrum, XP and the HFSD process.
- Please make yourself familiar with the process in the within the next week.


Pearson International Edition
Agile Software Development with Scrum
KEN SCHWABER
MIKE BEEDLE

+ scrum.org



# The Scrum process for SWEP

- Here:
  - Tutor: Product Owner
  - Students:
    - Scrum Master
    - Development
    - Sprint Planning, Sprint Review/Retrospective
    - Release Presentation, including Metadata Analysis:
      - Burdown Chart
      - Velocity
      - What went well
      - What went wrong
      - ...