



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Chapter 05

FROM V-MODEL TO SCRUM

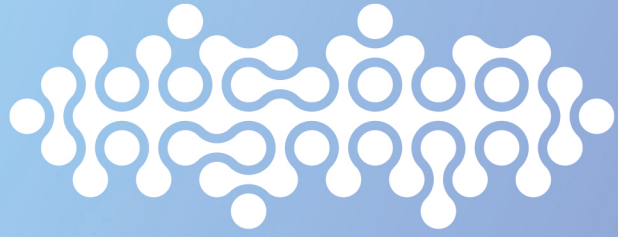
Today's Agenda

Part 1: What Traditional Methods Struggle With

- The V-Model explained
- Exercise: "The Telephone Game"
- The Planning Paradox

Part 2: Scrum in Action

- Scrum 3-5-3: Roles, Events, Artifacts
- Exercise: "The Sprint Game"
- The Power of the Board

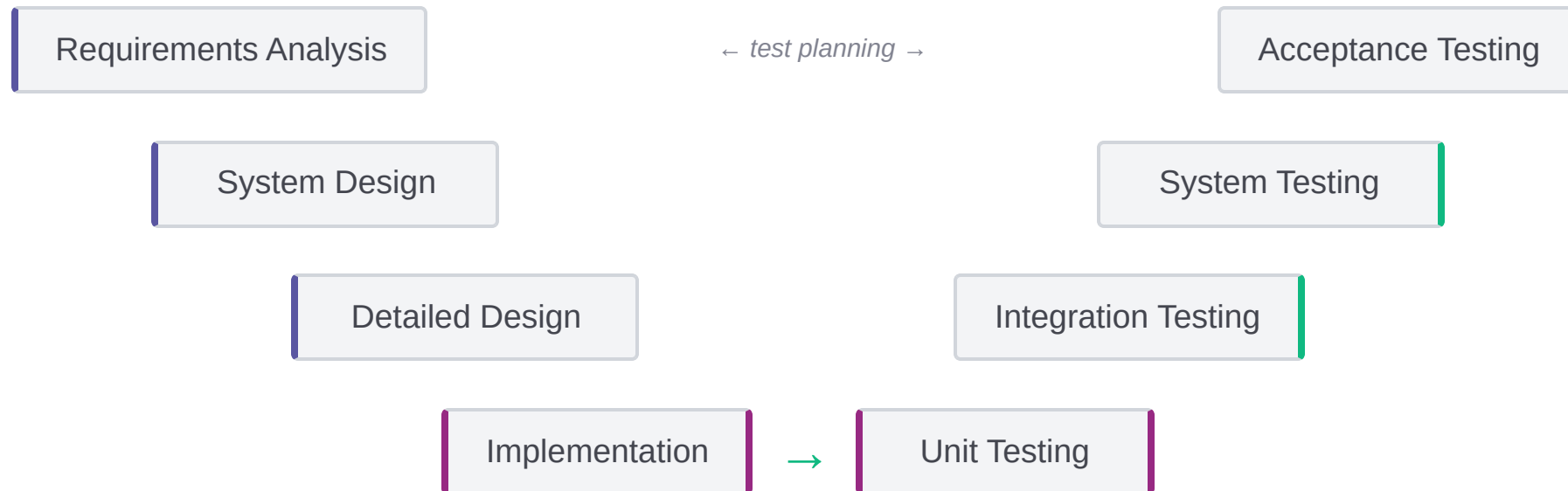


Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Why Traditional Methods Fail

1

The V-Model: A Quick Recap



Key idea: Each development phase has a corresponding test phase.

V-Model: The Theory Sounds Great!

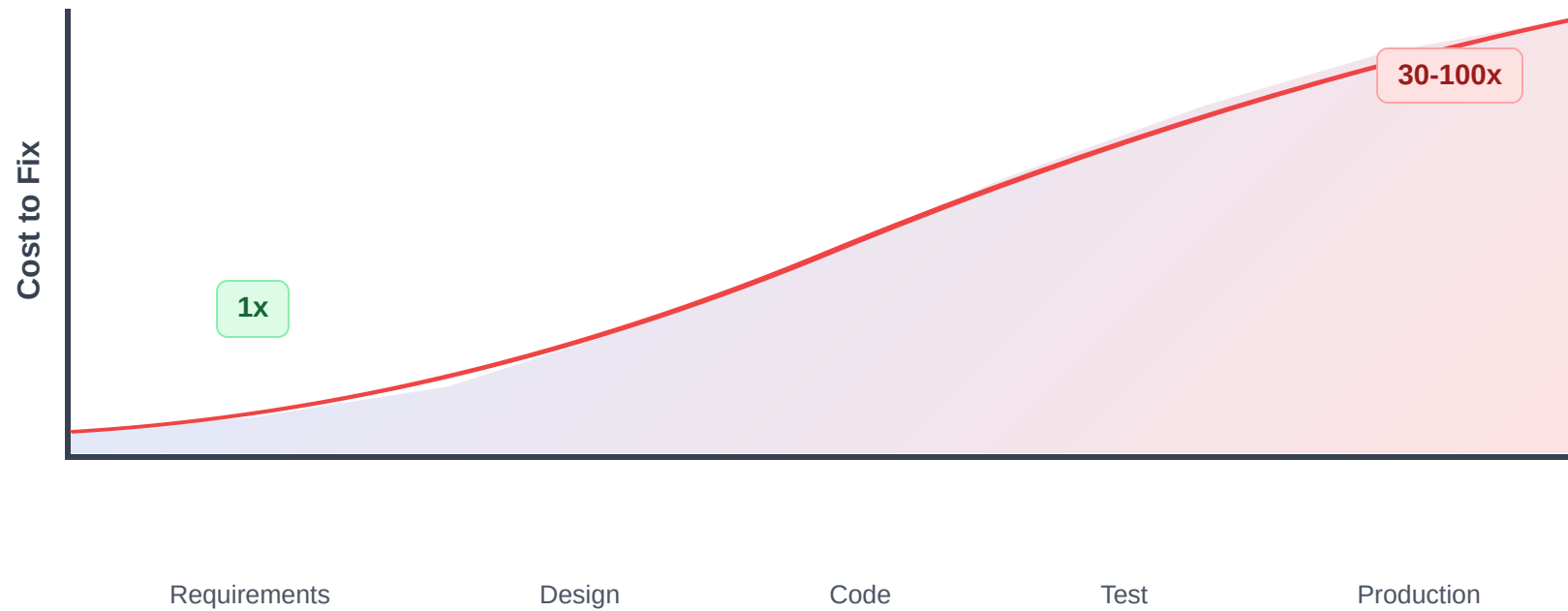
Promises:

- Clear phases with defined deliverables
- Testing planned from the start
- Traceability from requirements to tests
- "We'll catch problems before they're expensive"

Reality:

- First working software appears months after requirements
- Requirements change, but the plan doesn't
- Problems found late = expensive fixes

The Cost of Change Curve



Barry Boehm (1976): Fixing a bug in production costs **30-100x** more than fixing it during requirements.

The Three False Assumptions

The V-Model assumes:

1. **"We can know all requirements upfront"**
 - Reality: Users don't know what they want until they see it
2. **"We can design completely before coding"**
 - Reality: Design problems only emerge when you build
3. **"We can build it right the first time"**
 - Reality: Learning happens through iteration

These aren't just hard to achieve - they're impossible.

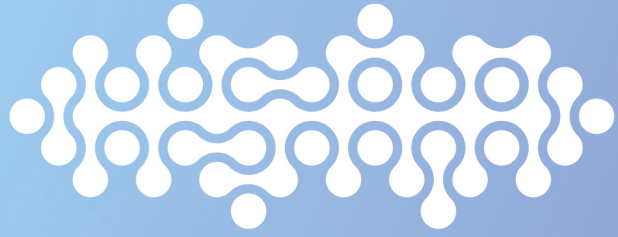
The Planning Paradox

*If we plan more, we waste time on plans that will change.
If we plan less, we build the wrong thing.*

Traditional Response: Plan even MORE carefully!

- More documentation
- More review meetings
- More sign-offs

Result: Slower delivery, same problems.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Exercise 1: The Telephone Game

2

Exercise 1: The Telephone Game

Goal: Experience why the V-Model's phase separation fails.

 **Join the Web App** - Code will be shown on screen

 **Key rule:** No asking questions between phases!

Topic: Study Room Booking App 

Exercise 1: Example (Mensa App)

Round	Role	Output
1	CUSTOMER	"Show today's menu, prices, veggie filter, and how long the queue is"
2	ANALYST	"REQ-1: Menu. REQ-2: Prices. REQ-3: Diet filter. REQ-4: Queue status"
3	DEVELOPER	"Dropdown, price column, checkbox, traffic light for queue"
4	TESTER	✗ "how long" (wait time) vs "traffic light" (busy level)

What went wrong? Ambiguous "queue" requirement → different interpretations!

Exercise 1: Your Turn!

Round	Your Role	Your Task
1	CUSTOMER	Write a feature request - be specific!
2	ANALYST	Read someone's request → write formal spec
3	DEVELOPER	Read a spec → describe your implementation
4	TESTER	See your original vs. what was built

 **The app automatically rotates documents between rounds!**

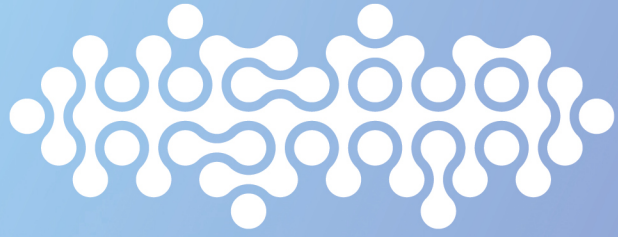
 **Remember:** No asking questions between phases!

Exercise 1: Debrief

Discussion Questions:

1. Where did information get lost?
2. What assumptions did each group make?
3. When did you first realize there was a mismatch?
4. How could continuous feedback have helped?

Key Insight: Late feedback = expensive fixes



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

The Scrum Framework

3

The Agile Response

Instead of fighting uncertainty with more planning...

Traditional	Agile
Complete upfront planning	Short planning cycles (1-4 weeks)
Heavy documentation	Working software as primary artifact
Change control boards	Welcome change as competitive advantage
Big bang releases	Frequent small releases
Testing at the end	Continuous testing throughout

The 3-5-3 Structure

Scrum provides structure without sacrificing flexibility.

The **3-5-3** Structure:

- **3 Roles:** Product Owner, Developers, Scrum Master
- **5 Events:** Sprint, Planning, Daily, Review, Retrospective
- **3 Artifacts:** Product Backlog, Sprint Backlog, Increment

The Three Roles

Role	Decides	Does NOT
Product Owner	WHAT gets built, priority	Tell devs HOW to build
Developers	HOW to build, estimates	Prioritize the backlog
Scrum Master	Process, removes blockers	Assign tasks, manage people

Critical: Clear separation of concerns!

Product Owner

The voice of the customer

- Defines and prioritizes the Product Backlog
- Represents stakeholders
- Accepts or rejects completed work
- Communicates the product vision

Key principle: One person, one voice, one priority.

Not a committee!

Developers (Development Team)

The people who do the work

- Self-organizing: decide HOW to do the work
- Cross-functional: have all skills needed
- Deliver a potentially releasable increment each sprint
- Typically 3-9 people

Key principle: Work is PULLED, not ASSIGNED.

Team members pick tasks from the board!

Scrum Master

Servant-leader for the team

- Facilitates Scrum events
- Removes impediments blocking the team
- Coaches the team on Scrum practices
- Protects the team from outside interference

Key principle: NOT a project manager!

Doesn't assign work or make decisions for the team.

Quick Quiz: Which Role?

Scenario 1: A customer wants to add a new feature mid-sprint.

- Who decides if it's worth interrupting the sprint?

Scenario 2: The team needs server access but IT won't respond.

- Who escalates this impediment?

Scenario 3: The team debates React vs. Vue for the frontend.

- Who makes the final technical decision?

Quick Quiz: Answers

Scenario 1: Customer wants new feature mid-sprint

- **Product Owner** - They control priorities

Scenario 2: Team blocked by IT

- **Scrum Master** - They remove impediments

Scenario 3: React vs. Vue debate

- **Development Team** - Technical decisions are theirs

The Five Events

Event	Duration	Purpose
Sprint	1-4 weeks	Container for all events
Sprint Planning	2-4 hours	Select work, define goal
Daily Scrum	15 minutes	Synchronize, identify blockers
Sprint Review	1-2 hours	Demo to stakeholders
Sprint Retrospective	1-2 hours	Improve the process

The Sprint Cycle



The Scrum Board

Why visual management matters

TO DO	IN PROGRESS	IN REVIEW	DONE
US-4	US-3	US-2	US-1
US-5			

At a glance: What's done, what's blocked, who needs help.

Healthy vs. Unhealthy Boards

Red Flag 1: 8 items "In Progress", 0 "Done"

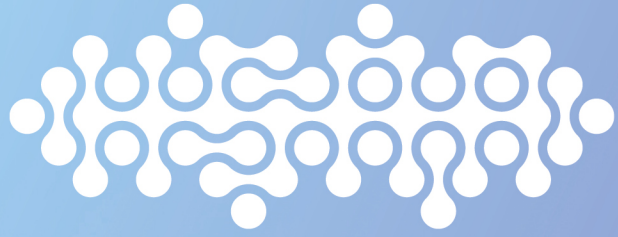
- **Problem:** Too much work in progress (WIP)
- **Fix:** Finish things before starting new things

Red Flag 2: Board unchanged for 3 days

- **Problem:** Hidden blockers or no updates
- **Fix:** Daily Scrum to surface issues

Red Flag 3: PO moving cards directly

- **Problem:** Role violation
- **Fix:** Team owns their workflow



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Exercise 2: The Sprint Game

4

Exercise 2: The Sprint Game

Goal: Experience a complete Scrum sprint in 30 minutes.

Your Team:

- 1 Product Owner
- 1 Scrum Master
- 3-4 Developers

Materials:

- Product Backlog cards
- Empty Sprint Board
- Timer

Exercise 2: Sprint Game Rules

Sprint Planning (5 min):

- PO presents top stories from backlog
- Team estimates and commits to sprint

Sprint Execution (15 min):

- Developers work on tasks
- Move cards on board as you progress
- SM watches for blockers

Sprint Review (5 min):

- Demo completed work to PO

Retrospective (5 min):

- What went well? What to improve?

Sprint Game: The Flow

Round 1 → Round 2 → Execution → Review → Results

Round 1: Backlog Prioritization (13 min)

- 1.1 Importance Voting: Rate each story ★★★★★
- 1.2 Priority Voting: Set execution order (0-3)
- 1.3 Final Prioritization: PO reorders, team votes confidence

Round 2: Story Assignment (10 min)

- 2.1 Developer Preferences: Check stories you want
- 2.2 PO Assignment: Stories assigned to developers
- 2.3 Satisfaction Voting: Rate your assignments 👍 😐 👎

Round 1: Prioritizing the Backlog

Phase 1.1: Importance Voting (5 min)

Everyone rates each story's importance: ★ to ★★★★★

Phase 1.2: Priority Voting (5 min)

Top stories selected → Vote on execution order (0-3)

Phase 1.3: Final Prioritization (3 min)

- PO can reorder the backlog (drag & drop)
- Team votes confidence: 👍 Ready | 😐 Unsure | 👎 Concerns

Round 2: Assigning Stories

Phase 2.1: Developer Preferences (3 min)

Check the stories you'd like to work on

Phase 2.2: PO Assignment (5 min)

PO assigns each story to a developer
(Sees who wanted each story)

Phase 2.3: Satisfaction Voting (2 min)

Rate your assignment: 👍 Happy | 😐 OK | 👎 Unhappy

PO can reassign based on feedback

Execution & Review

Execution Phase

Work on assigned stories using the Kanban board:

To Do → In Progress → Review → Done

Review Phase

Stories in "Review" need team approval:

- Everyone votes: 👍 Approve or 👎 Reject
- All approve → Story moves to Done
- Any reject → Back to In Progress for fixes

Sprint Game: Tips for Success

- **Vote honestly** - Your opinion shapes the sprint
- **Communicate** - Use the board, talk to teammates
- **Pull work** - Don't wait to be assigned tasks
- **Move cards** - Keep the board updated in real-time
- **Time matters** - Each phase has a timer!

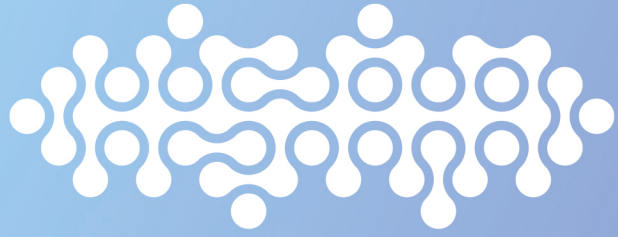
Ready? Let's start the Sprint Game!

Exercise 2: Debrief

Discussion Questions:

1. Did you complete everything you committed to?
2. What blocked you during the sprint?
3. How did the board help (or not help)?
4. What would you do differently next sprint?

Remember: Incomplete work returns to the backlog!



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Agile in the Real World

5

Real World: US Military Goes Agile

March 2024: US Army mandates Agile for all software

*"We're learning from current conflicts - including in Ukraine - that the Army's success on future battlefields will depend on our ability to **rapidly update software**."*
— Secretary of the Army Christine Wormuth

If it works for **mission-critical military systems**, it can work for your project.

Traditional vs. Agile Military Software

Traditional (5-10 years)	Agile (weeks-months)
Detailed requirements docs	Continuous soldier involvement
Rigid acquisition process	Modular contracting
Separate testing phases	Continuous integration
Deploy once, maintain forever	Continuous improvement

The lesson: "Agile doesn't scale" is a myth.

Key Takeaways

1. **V-Model fails** because its assumptions are impossible to meet
2. **Agile accepts uncertainty** and builds short feedback loops
3. **Scrum provides structure** through roles, events, and artifacts
4. **The board is critical** - transparency enables collaboration
5. **Work is PULLED, not assigned** - teams self-organize

Apply This Today

For your Road Profile Viewer:

1. Create a Product Backlog as GitHub Issues
2. Plan a 1-week sprint with a clear goal
3. Set up a GitHub Project board
4. Hold Daily Standups (even async on Slack)
5. Do a Sprint Review to demo your progress
6. Run a Retrospective to improve

The best way to learn Agile is to practice it.

Resources

Official Sources:

- [The Scrum Guide](#) - Free, 13 pages
- [US Army Agile Policy \(2024\)](#)

Tools:

- GitHub Projects - Free, integrates with Issues
- Jira - Enterprise-grade
- Linear - Modern, developer-friendly



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Questions?

THANK YOU

Next: Take the Scrum Quiz to test your understanding!