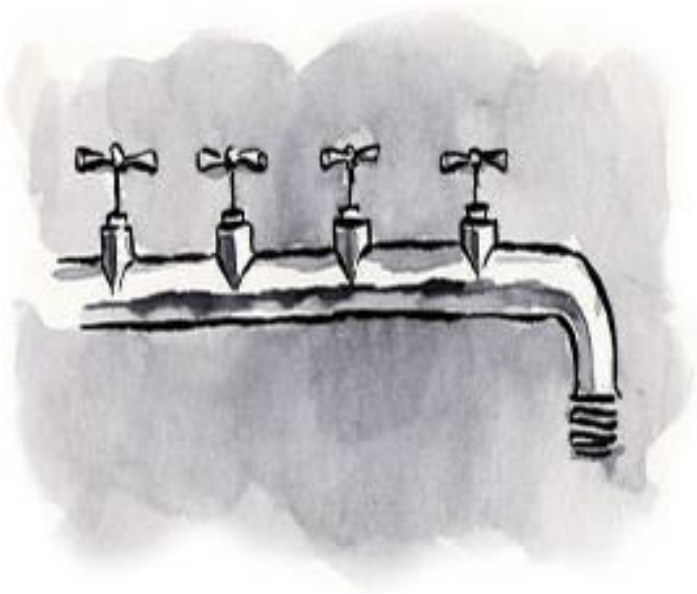
Part 2

AGILE & LEAN DEVELOPMENT



PRACTICES

OBJECTIVES

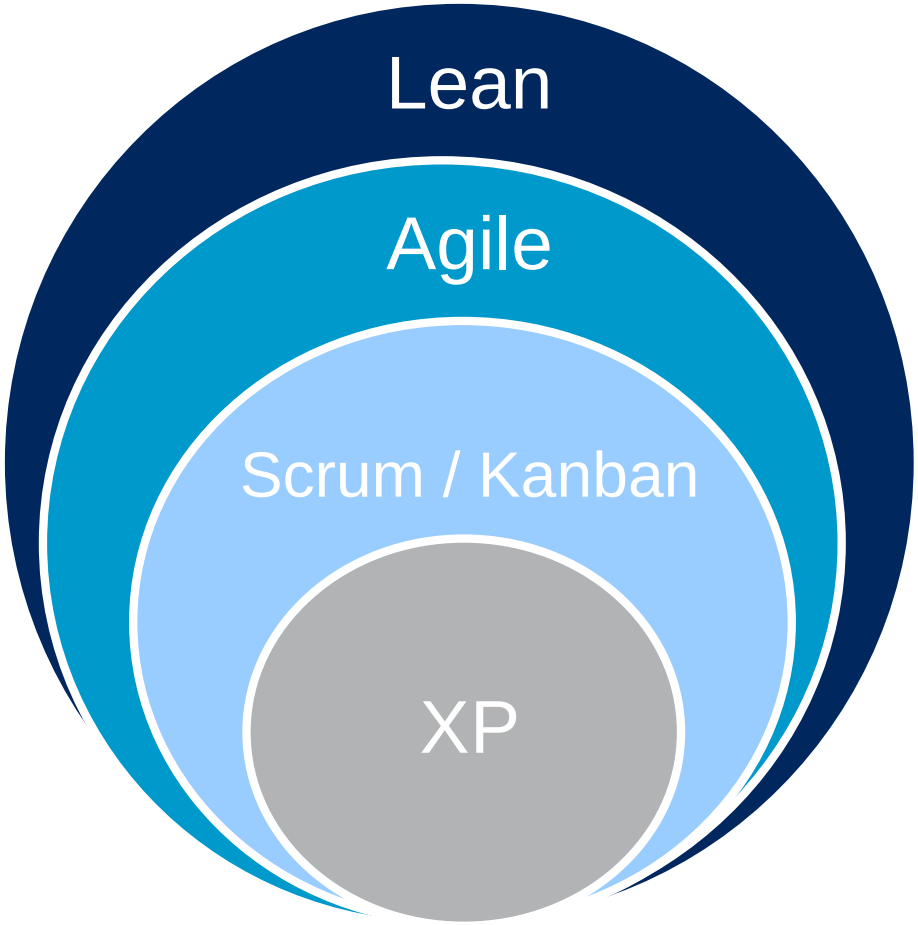
- Give introduction to “Agile concepts” to:
  - achieve a common understanding and terminology
  - information for further discussion and self-study

- Overview of Agile concepts:
  - Part 1: Agile, Lean
  - Part 2: Scrum, XP, Kanban
  - Part 3: User Stories, tasks, estimation and planning

- Disclaimer: - This is not a new mandatory Way of Working!
  - Just information and food for thought!

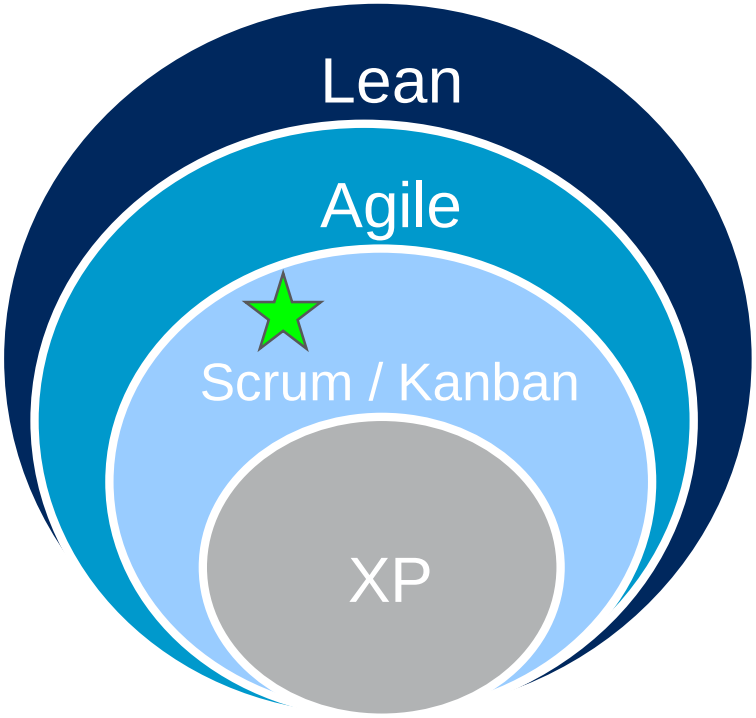
THEORY & METHODOLOGY

- Lean (theory)
  - Widely applicable for any kind of production
- Agile (theory)
  - is Lean
- Scrum (methodology)
  - is a planning method
- Kanban (methodology)
  - a planning philosophy
- XP (methodology)
  - Team specific values and practices



Reference: http://blog.crisp.se/henrikkniberg/

SCRUM

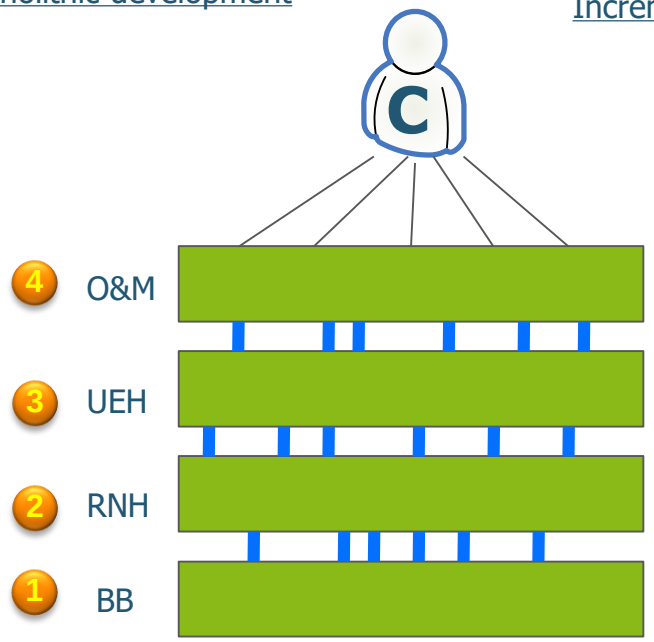

Lean
Agile
Scrum / Kanban
XP


Hooker
Loosehead prop
Tighthead prop
2nd row
2nd row
Blindside flanker
Openside flanker
Number 8
1
2
3
4
5
6
7
8

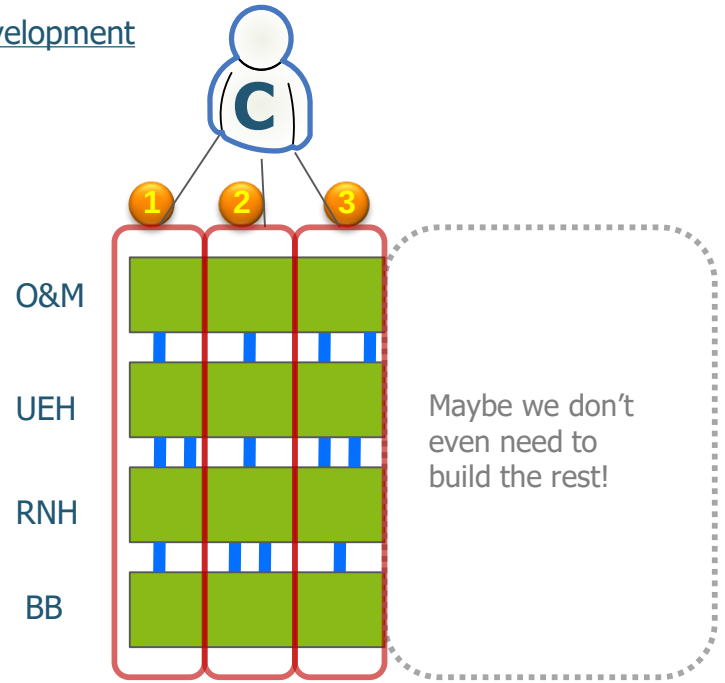
SCRUM IS A ITERATIVE, INCREMENTAL METHODOLOGY

ITERATIVE = DON'T EXPECT TO GET IT ALL RIGHT THE FIRST TIME
INCREMENTAL = BUILD IN "VERTICAL" SLICES (FEATURES) RATHER THAN "HORIZONTAL" (LAYERS)

Monolithic development



Incremental development



SCRUM

- First public appearance in 1995 when Jeff Sutherland and Ken Schwaber paper describing Scrum at OOPSLA.

- *“Scrum is a simple **"inspect and adapt" framework** that has three roles, four ceremonies, and three artifacts designed to deliver working software in Sprints, usually 30-day iterations”*

3 Roles

4 Cermonies

3 Artifacts

ROLES AND RESPONSIBILITIES

The Scrum Team

Product Owner

- Defines features and release dates
- Responsible for Return of Investment (ROI)
- Prioritizes features by business value
- Accepts or rejects work

Scrum Master

- Ensures team is functioning and productive
- Removes barriers (impediments)
- Shields team from external interference
- Ensures the process is followed
- Facilitates planning, not a traditional PjM

The Team

- Cross functional, 7 +/-2 members
- Self directed
- Organizes itself and assign tasks
- Commits to Sprint and Demos to Product Owner

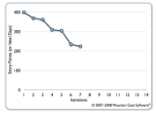
SCRUM FLOW

Product Owner

The team

Scrum Master

Release Burndown



Sprint Burndown

Product Backlog

Sprint Backlog

- Work Items assigned to a Sprint
- Estimated by the team
- Team Commitment

Time-boxed Test/Develop

Working Code Ready for Deployment

Sprint Planning Meeting

- Review Product Backlog
- Estimate Sprint Backlog
- Commit

Daily Scrum Meetings

- Done since last meeting
- Plan for today
- What is blocking us

Sprint Review Meeting

- Demo features to all

Sprint Retrospective

- Adjustments

Time boxed “Sprint” Cycles

THE SPRINT COMMITMENT

› Team's commitment to the product owner:

... we will do everything in our power, to reach the sprint goal and complete the stories,

... we will work on stories in priority order,

... we will display our progress and status on a daily basis and keep you informed,

... every story that we do deliver is complete.

RECIPROCAL COMMITMENTS

The product owner commits to leave priorities alone during the sprint

The team commits to delivering some amount of functionality – **forecasting** what

"No changes" during a sprint!

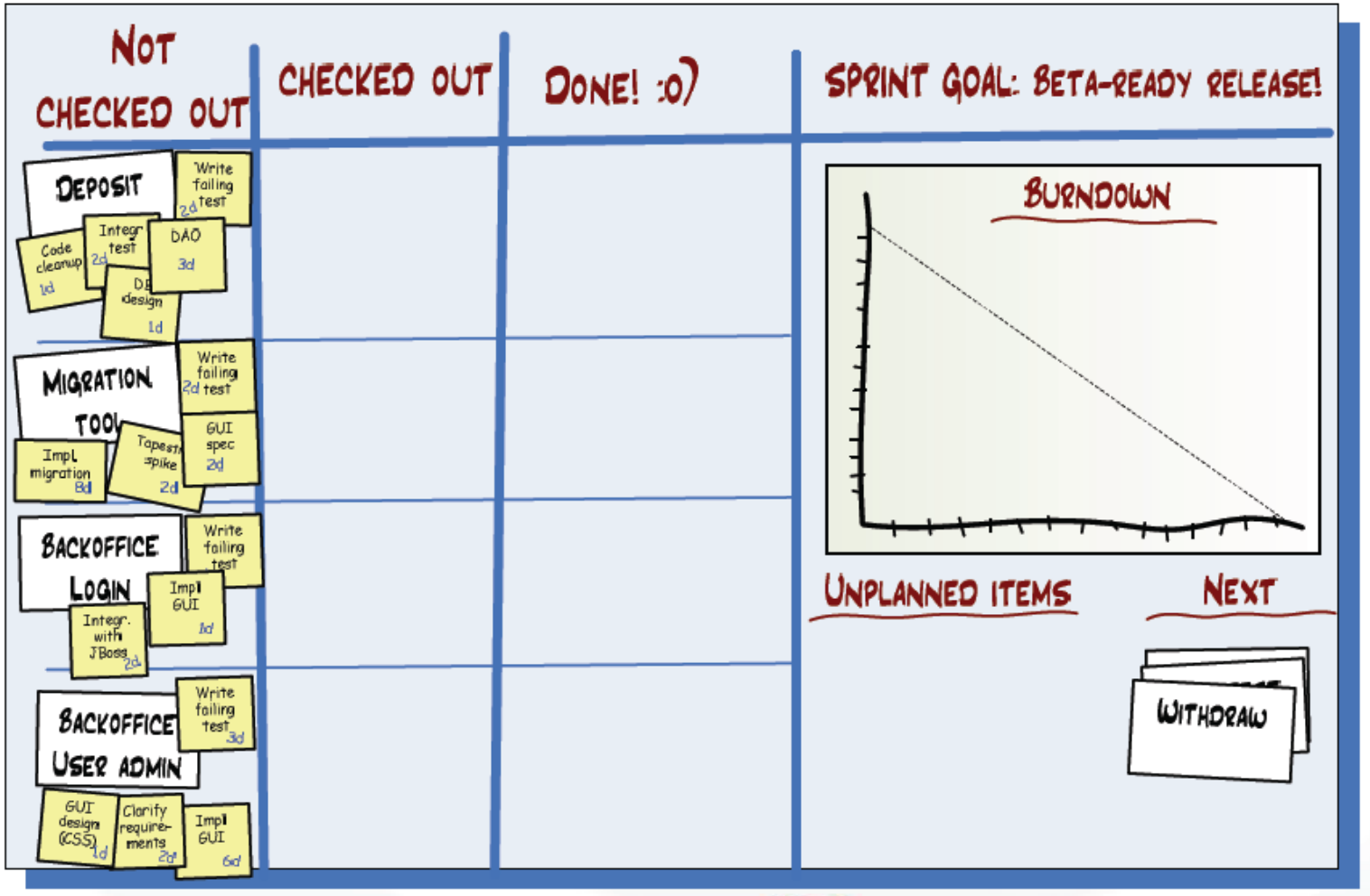
•What the team commits to – and what the product owner agrees to – during sprint planning, should be what is delivered.

However keep in mind that:

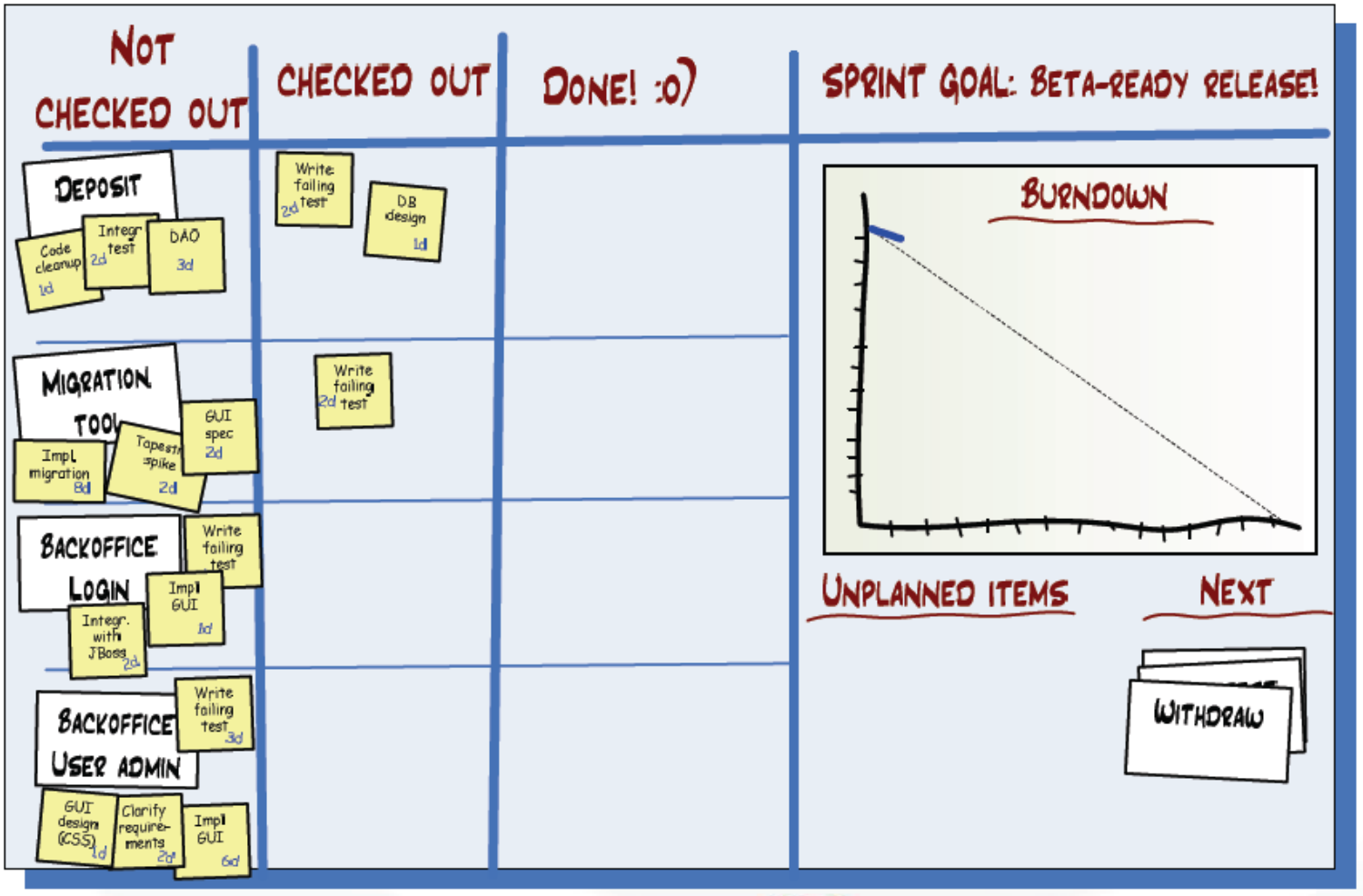
•We start with vague requirements

•Our understanding of those requirements is refined during the sprint

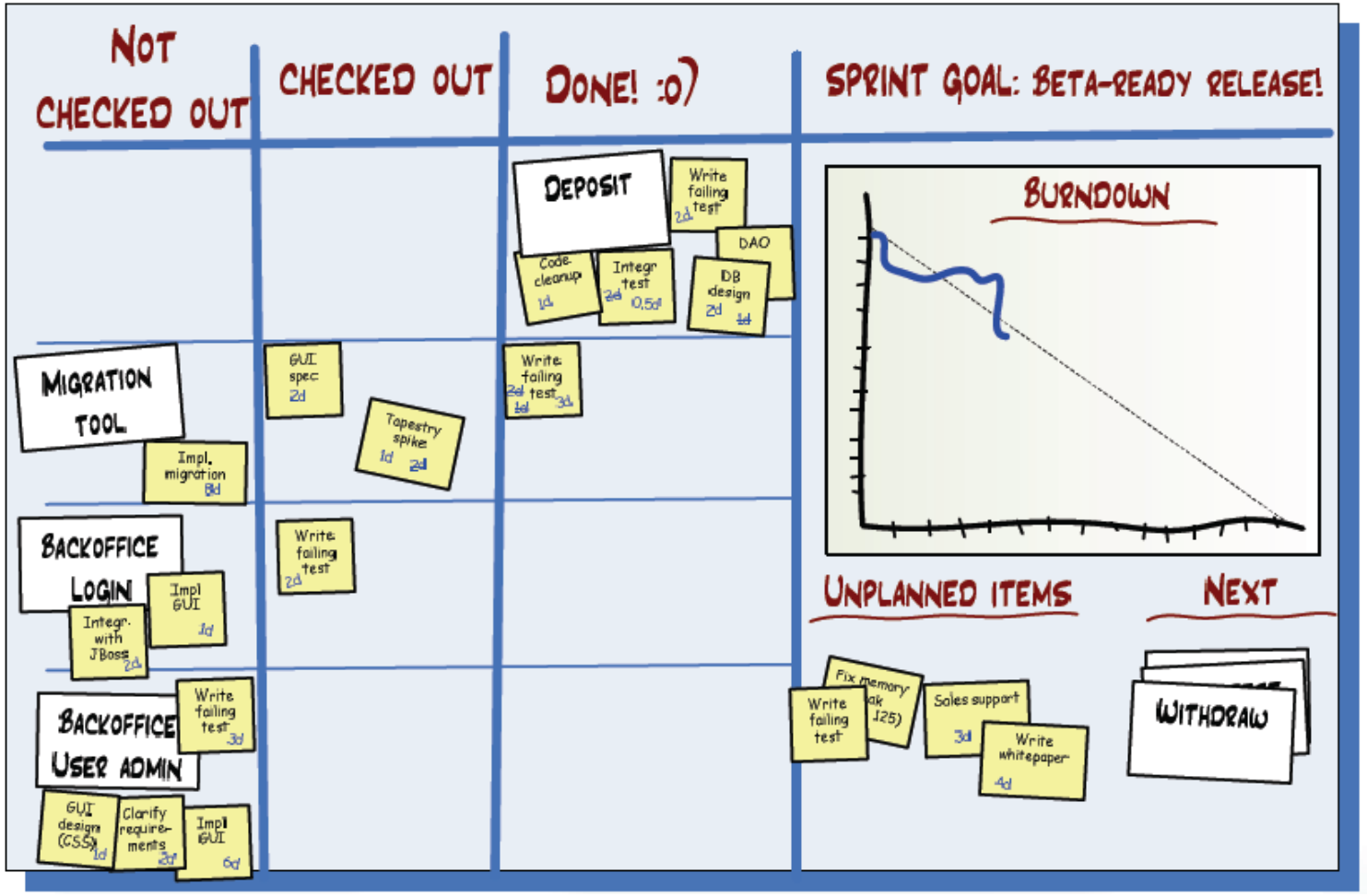
TASK BOARD WITH BURN DOWN



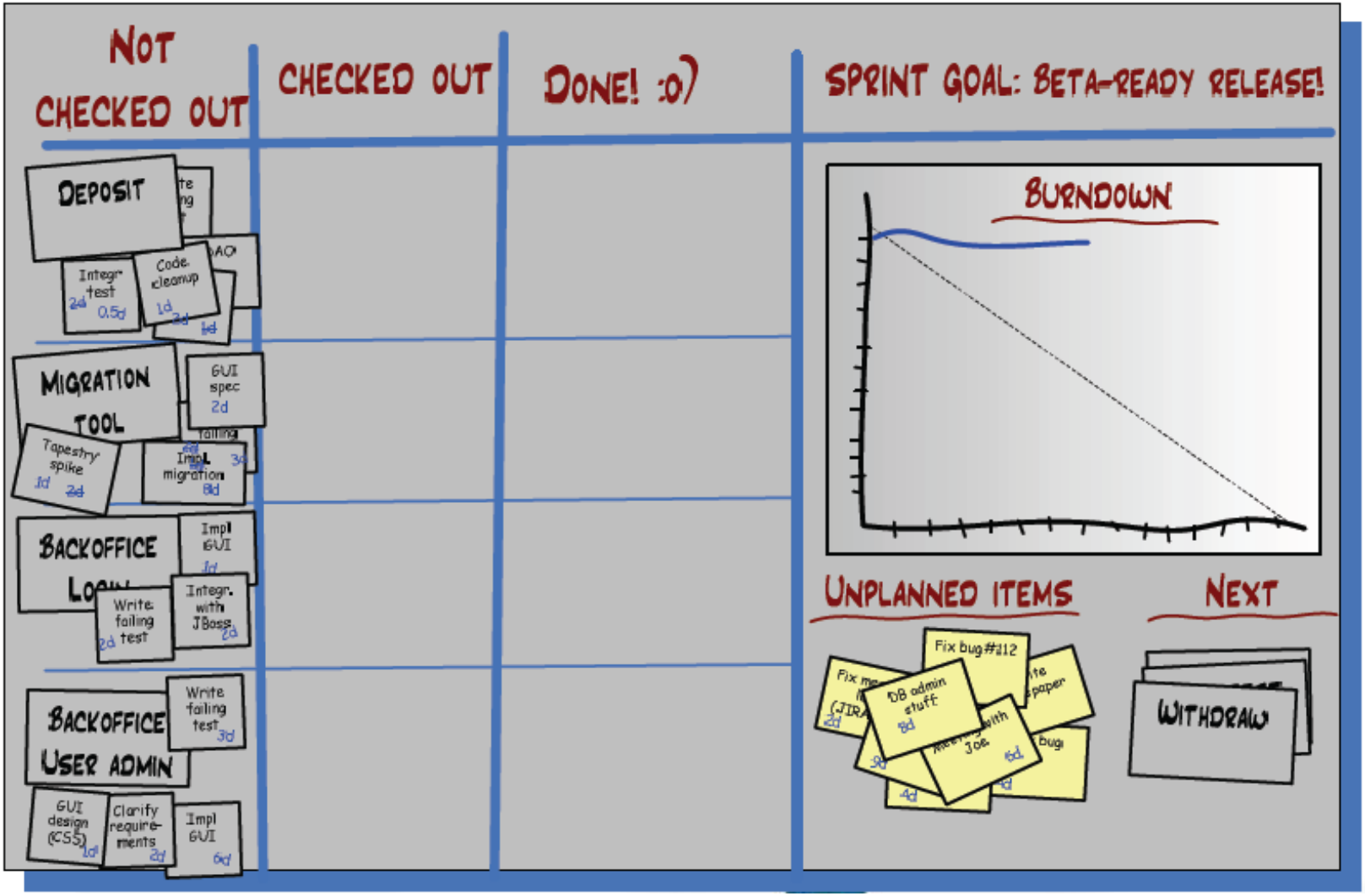
TASK BOARD AFTER 1ST MEETING



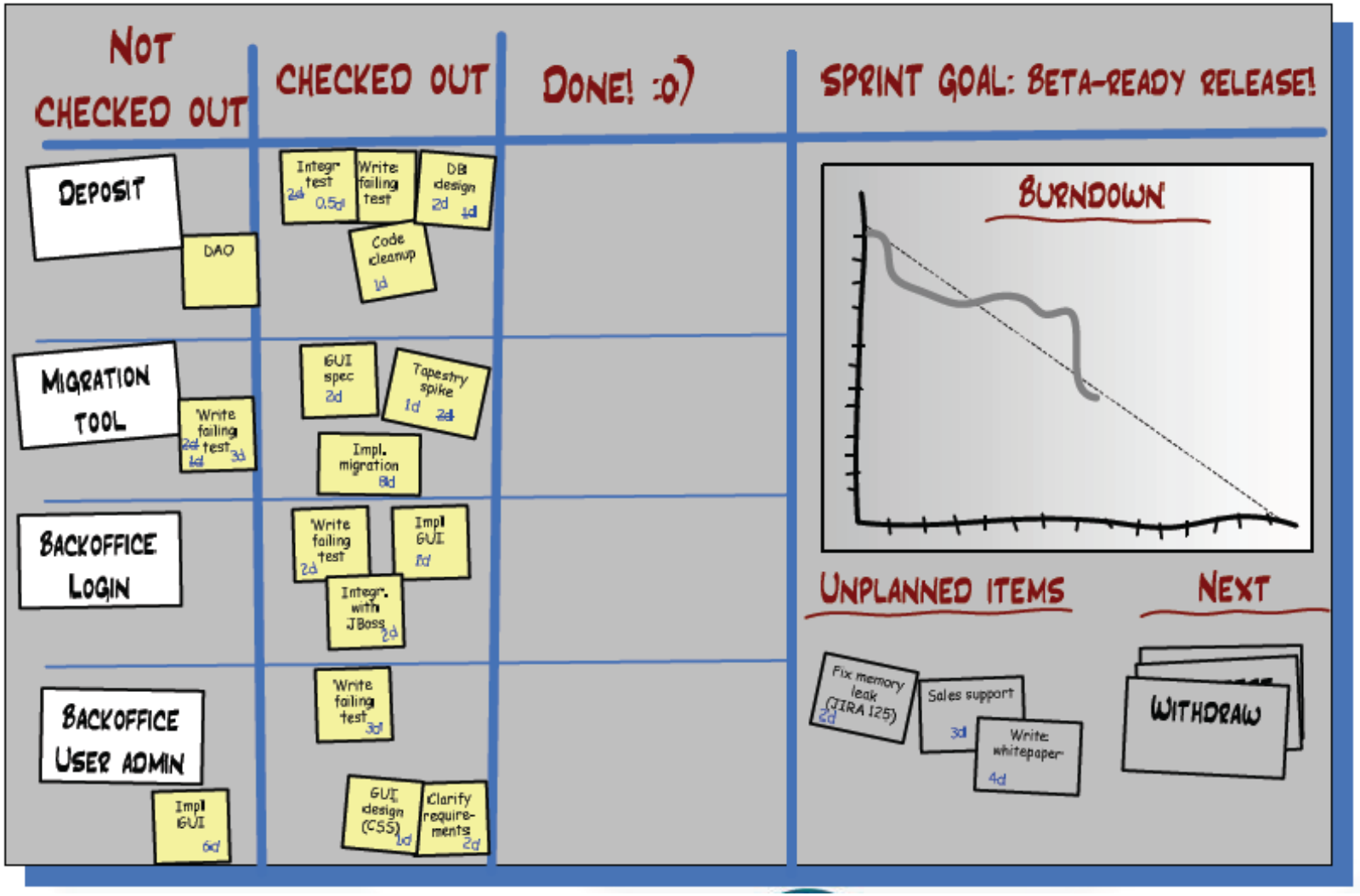
TASK BOARD AFTER X MEETING



WARNING SIGN #1



WARNING SIGN #2



THE SPRINT REVIEW - DEMO

- Team presents what it accomplished during the sprint
- Focus on ready stories
- Whole team participates
- Invite the world!
- **Get feedback**

THE IMPORTANCE OF DONE

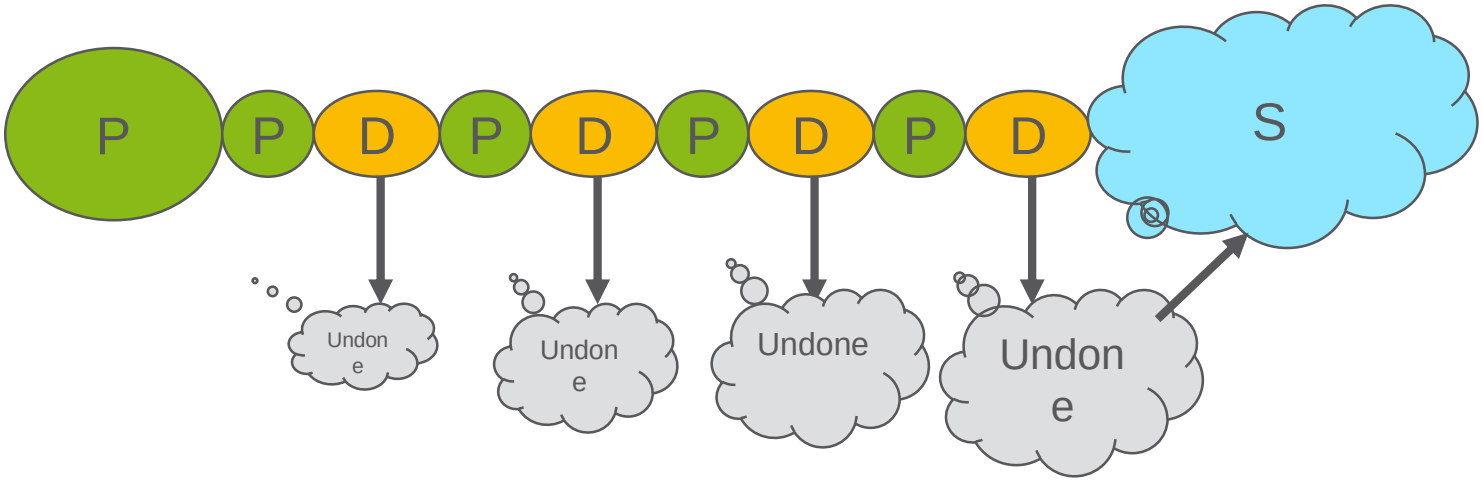
- Done must be clearly defined
- Strive for “No Work Left”
  - It might take some time, maybe years
- Understand what your present Done really means
  - What work is left before shipment?
  - When will that work be done?
- Done Done
  - 1st Done = up to a clearly defined step
  - 2nd Done = All work ready!

- You are not done if your are not Done!

Done = “No work left”

STABILIZATION IS WHEN YOU DO ALL THE UNDONE WORK

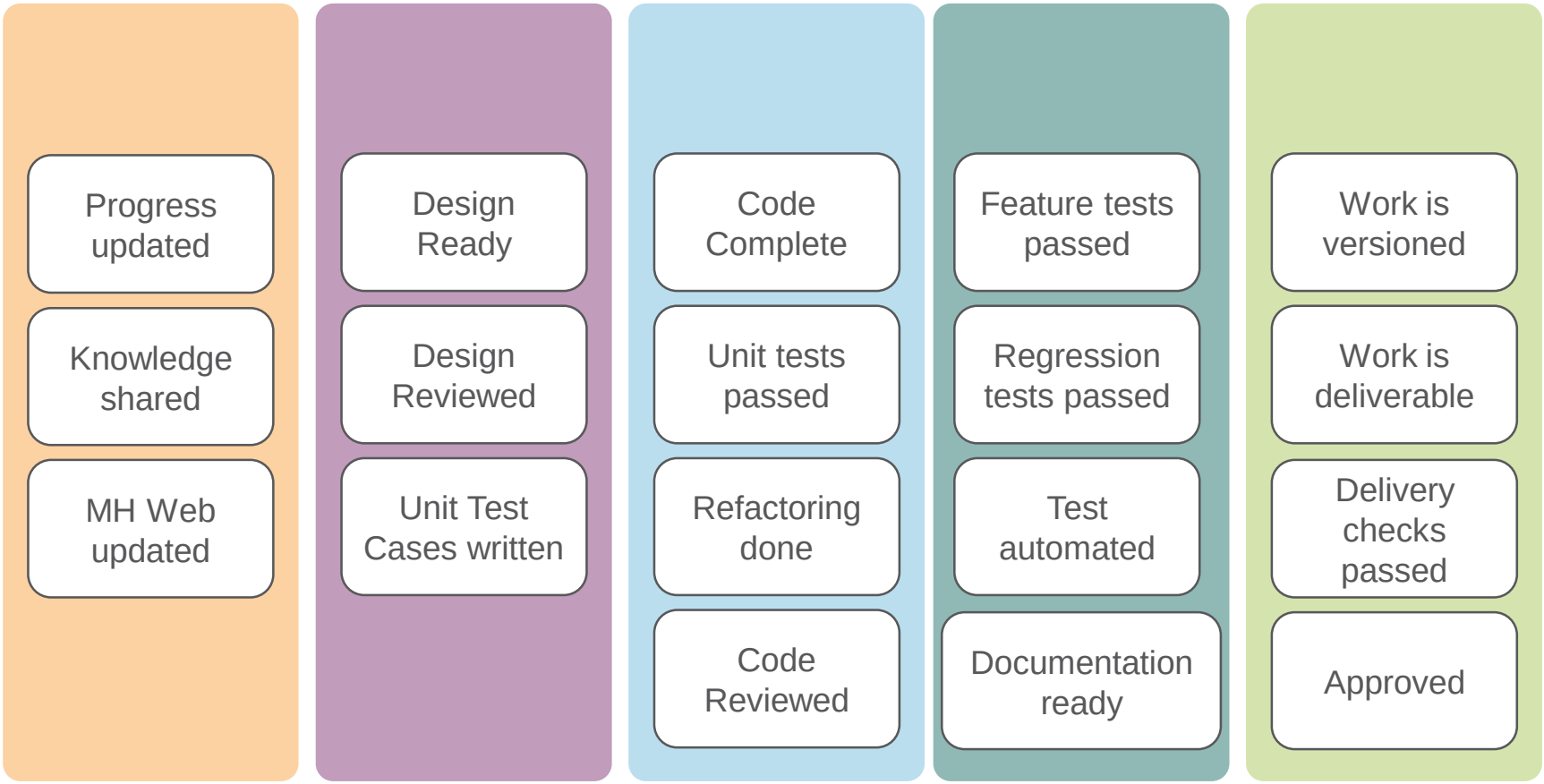
Scrum project with incomplete or variable “Done”



P Planning D Development S Stabilization

DOD - OVERVIEW

Example



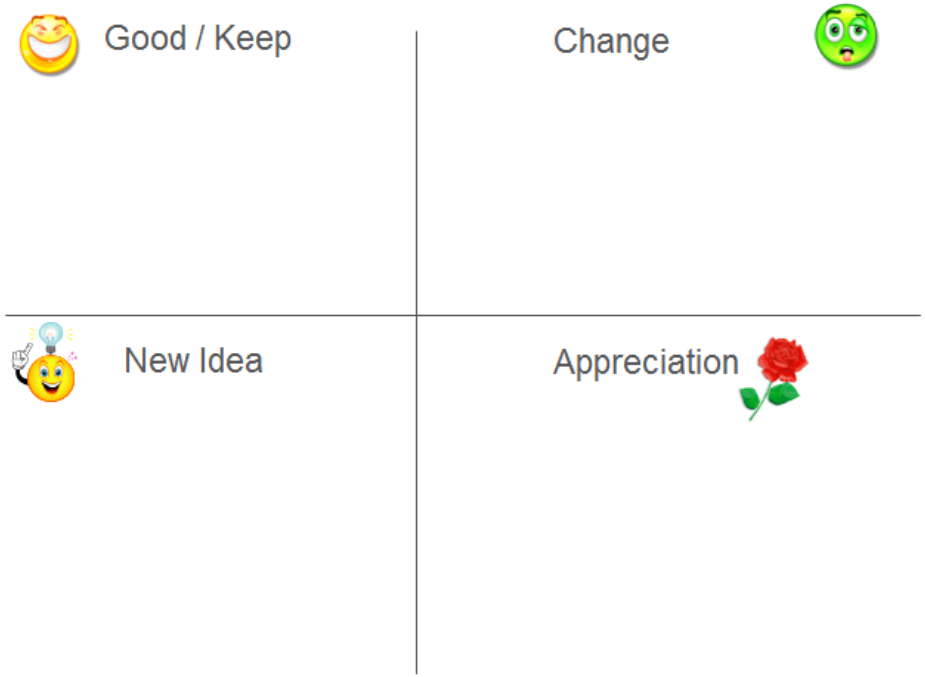
DEFINITION OF DONE - GENE

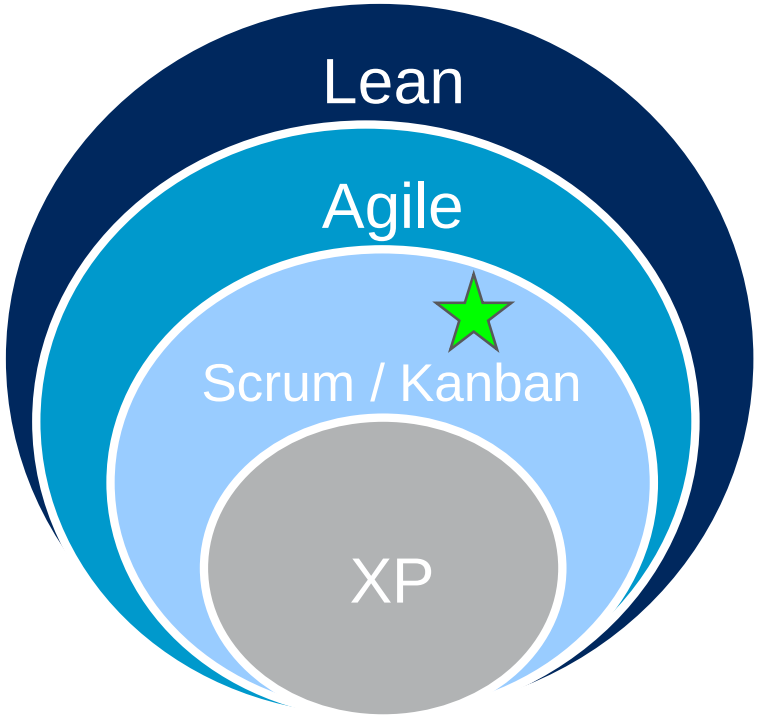
Example

- **Collaborated & Designed**
  - The team has, together with stakeholders discussed and described the function.
  - The team members has together discussed and defined the technical implementation and design solution.
- **Coded**
  - The work item is coded, i.e. all code necessary to implement the work item is written.
  - This also includes code for unit tests and functional tests, covering both new and old code, as well as updates of automated test suites
  - Refactoring is done.
- **Versioned**
  - All code (including test code) and documentation is checked in into the version handling system.
- **Tested**
  - The work item is tested. This includes
    - New/Added/Changed functionality is tested
    - Old functionality (legacy) is Regression tested
    - All design and code reviewed or desk checked according to team review plan
- **Documented**
  - Planned documentation is written or updated and reviewed. This includes Product Documentation, User Documentation and Trouble Shooting Guidelines.
- **Knowledge Shared**
  - Knowledge about what has been done is shared within the team. How has the solution been implemented and what artifacts has been affected. (Components, test, documentation etc)
  - This can bee done in several ways, e.g. by Pair Programming, reviews or workshops.
- **Deployable/Deliverable**
  - The work item is ready for delivery including all updates necessary for installation/integration.
  - Delivery checks is passed
- **Approved**
  - The Product Owner together with other stakeholders has given feedback and approved the solution.
  - Known defects are resolved. Remaining defects should be registered in MHWeb (TRs)
  - Progress is updated

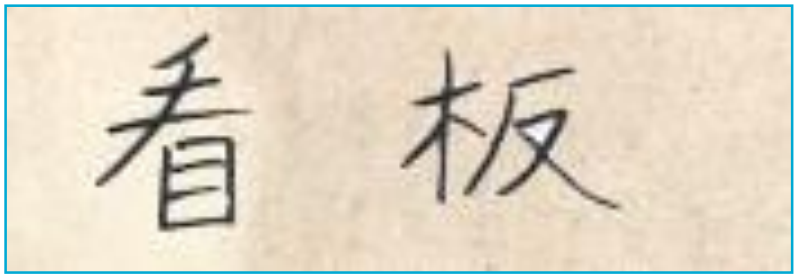
THE SPRINT RETROSPECTIVE

- Periodically take a look at what is and is not working
- Performed after every sprint
- Time boxed
  - decide duration in advance
- Whole team participates
  - Scrum Master
  - Team
  - Product Owner (optional)
  - Possible customer and others (by invitation of team)





KANBAN



WHAT IS KANBAN

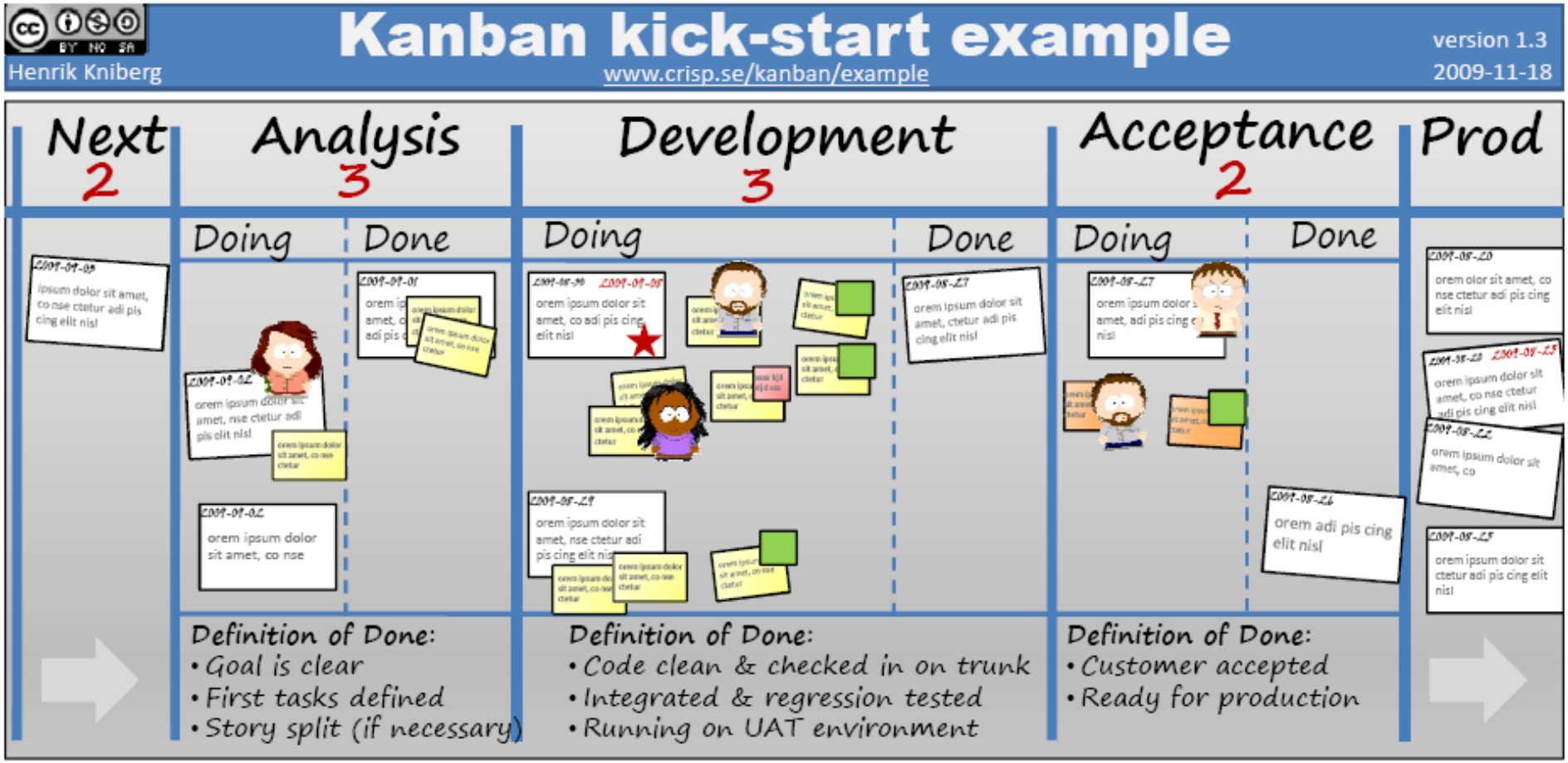
Kanban is most closely associated with Toyota's lean work

- Translates into "**visual card**"
- A signaling system used to trigger subsequent actions, illustrated by manufacturing pull systems
- Used originally in the production of products on a manufacturing line
- Supports the lean concept of "Just In Time" (JIT) production

THE 3 RULES OF KANBAN

- Visualize the workflow
  - Split the work into pieces, write each item on a card and put on the wall.
  - Use named columns to illustrate where each item is in the workflow.
- Limit Work in Progress (WIP)
  - assign explicit limits to how many items may be in progress at each workflow state
- Pull value through the system
  - Only start work when the need is created by the system
  - Measure the lead time and optimize to reduce leadtime

KANBAN BOARD



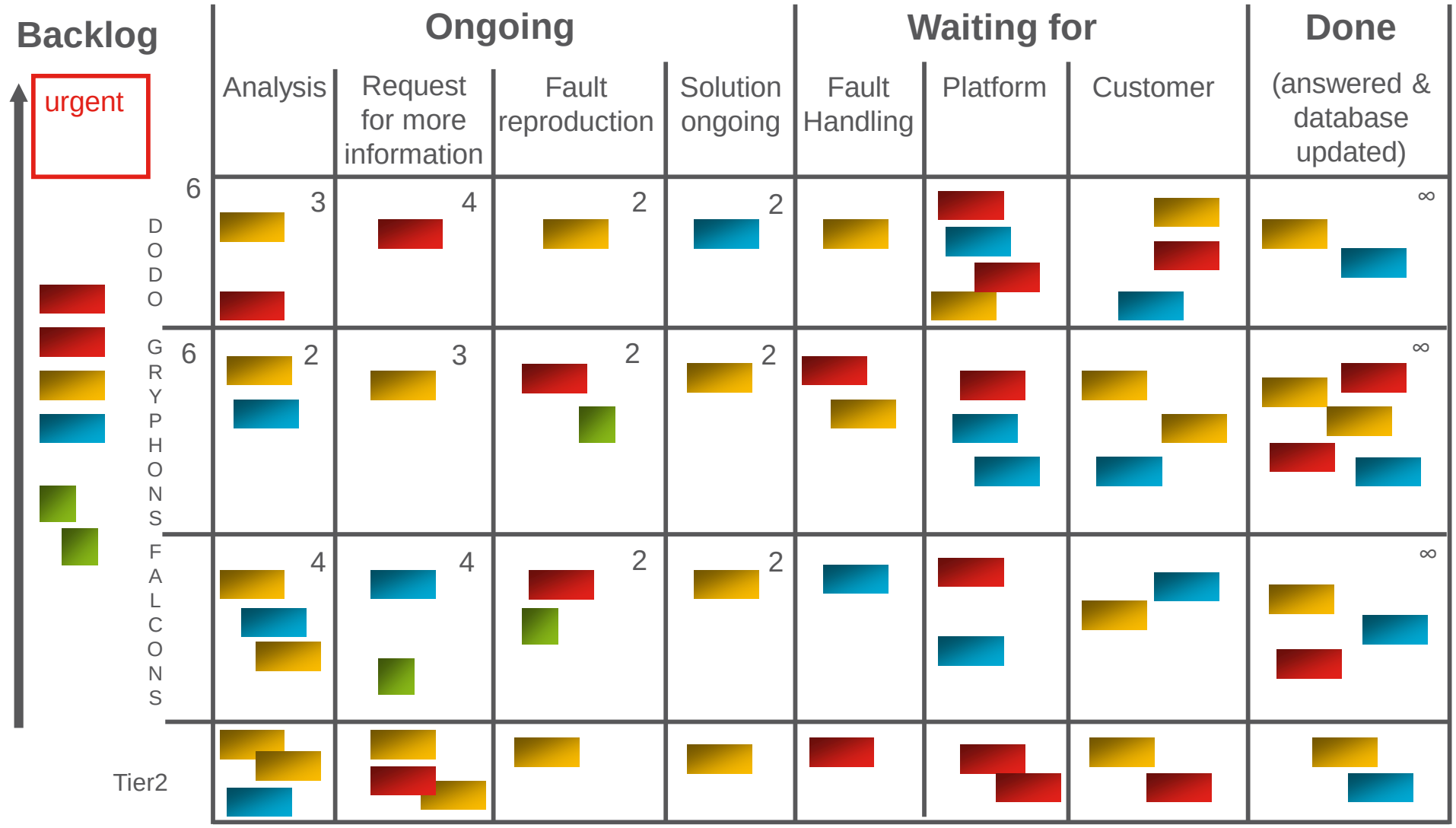
DIFFERENCES

Scrum	Kanban
Timeboxed iterations prescribed.	Timeboxed iterations optional.
Team commits to a specific amount of work for this iteration	Commitment optional.
Uses Velocity as default metric	Uses Lead time as default metric
Cross-functional teams prescribed.	Cross-functional teams optional.
Items must be broken down so they can be completed within 1 sprint.	No particular item size is prescribed.
Burndown chart prescribed	No particular type of diagram is prescribed
WIP limited indirectly (per sprint)	WIP limited directly (per workflow state)
Estimation prescribed	Estimation optional
Cannot add items to ongoing iteration.	Can add new items whenever capacity is available
A sprint backlog is owned by one specific team	A kanban board may be shared by multiple teams or individuals
Prescribes 3 roles (PO/SM/Team)	Doesn’t prescribe any roles
A Scrum board is reset between each sprint	A kanban board is persistent
Prescribes a prioritized product backlog	Prioritization is optional.

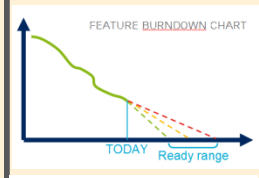
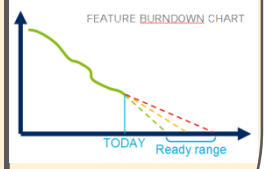
EXPERIENCES OF USAGE SCENARIOS

- Kanban good for maintenance and support
- For overview on “program” levels

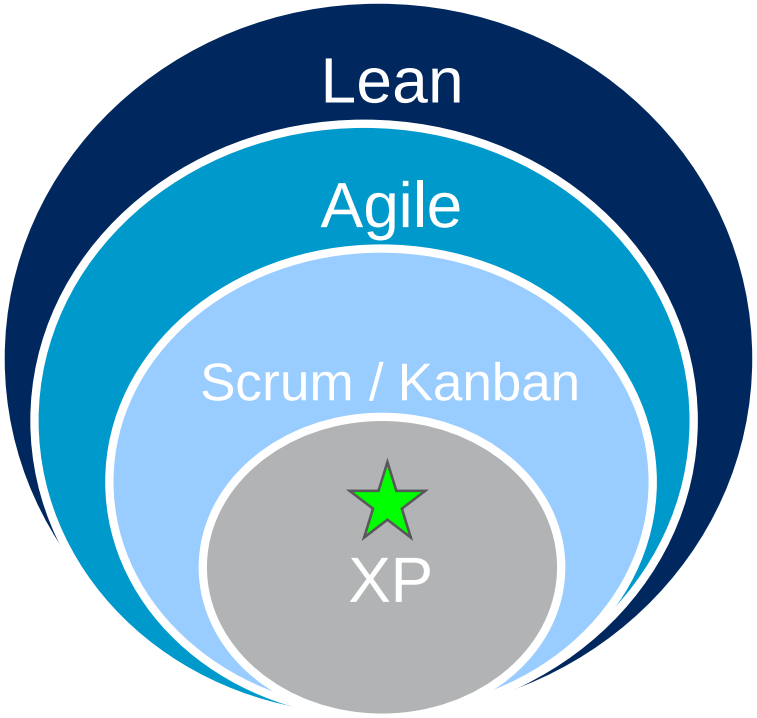
CSR HANDLING KANBAN BOARD



KANBAN BOARD FOR AREA

Next WIP Limit	Analysis WIP Limit		Development WIP Limit			Delivery WIP Limit	Done
	Doing	Done	ToDo	OnGoing	Done		
Story	Story 1	Story 5	*Team 1* Story	Story	Story	Story	TR 456
TR 899							TR 555
Story	Story		*Team 2* Story	Story Story		Story	Story
			Team n Story	TR 123	TR 234		
AVERAGE FLOW TIME: AVERAGE WAITING TIME: NO./WEEK:			*Other* Story	Story	Story		
		DoD			DoD		DoD

EXTREME PROGRAMMING



Kent Beck, 1999

XP-12 PRACTICES IN FOUR AREAS

1. Fine scale feedback
   - **Pair programming**
   - Planning game
   - **Test driven development**
   - Whole team
2. Continuous process
   - **Continuous integration**
   - Design improvement/Refactoring
   - Small releases
3. Shared understanding
   - Coding standard
   - Collective code ownership
   - Simple design
   - System metaphor
4. Programmer welfare
   - Sustainable pace

More emphasis on coding / programming techniques

THE RULES OF EXTREME PROGRAMMING - XP

Planning

› User stories are written.
› Release planning creates the release schedule.
› Make frequent small releases.
› The project is divided into iterations.
› Iteration planning starts each iteration.

Managing

› Give the team a dedicated open work space.
› Set a sustainable pace.
› A stand up meeting starts each day.
› The Project Velocity is measured.
› Move people around.
› Fix XP when it breaks.

XP DEVELOPMENT CONT.

Designing

› Simplicity.
› Choose a system metaphor.
› Use CRC cards for design sessions.
› Create spike solutions to reduce risk.
› No functionality is added early.
› Refactor whenever and wherever possible.

Coding

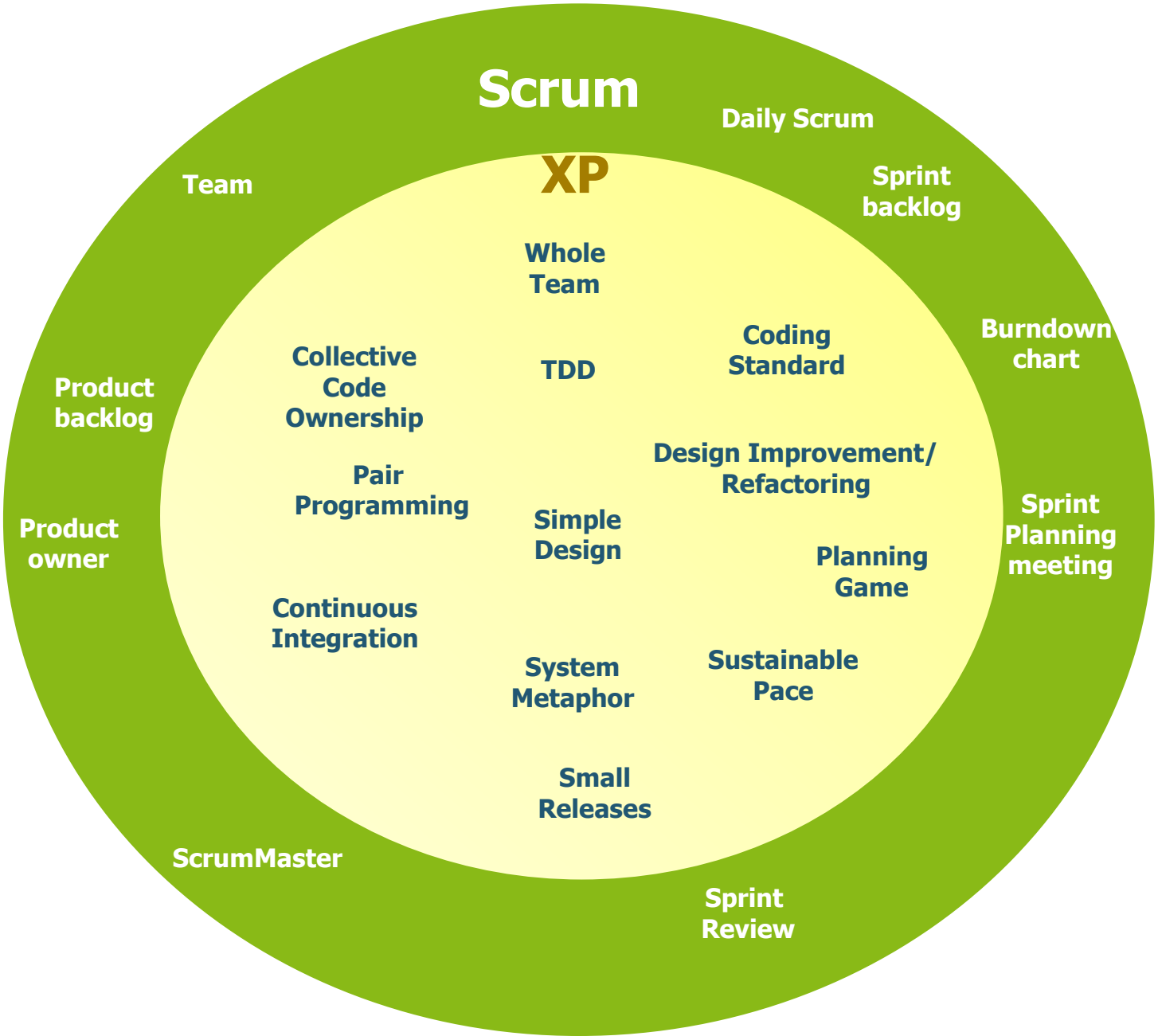
› The customer is always available.
› Code must be written to agreed standards.
› Code the unit test first.
› All production code is pair programmed.
› Only one pair integrates code at a time
› Integrate often.
› Set up a dedicated integration computer
› Use collective ownership.

XP DEVELOPMENT CONT.

Testing

- All code must have unit tests.
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance tests are run often and the score is published.

SCRUM "WRAPS" XP



SCALING AGILE

AVOID SCALING

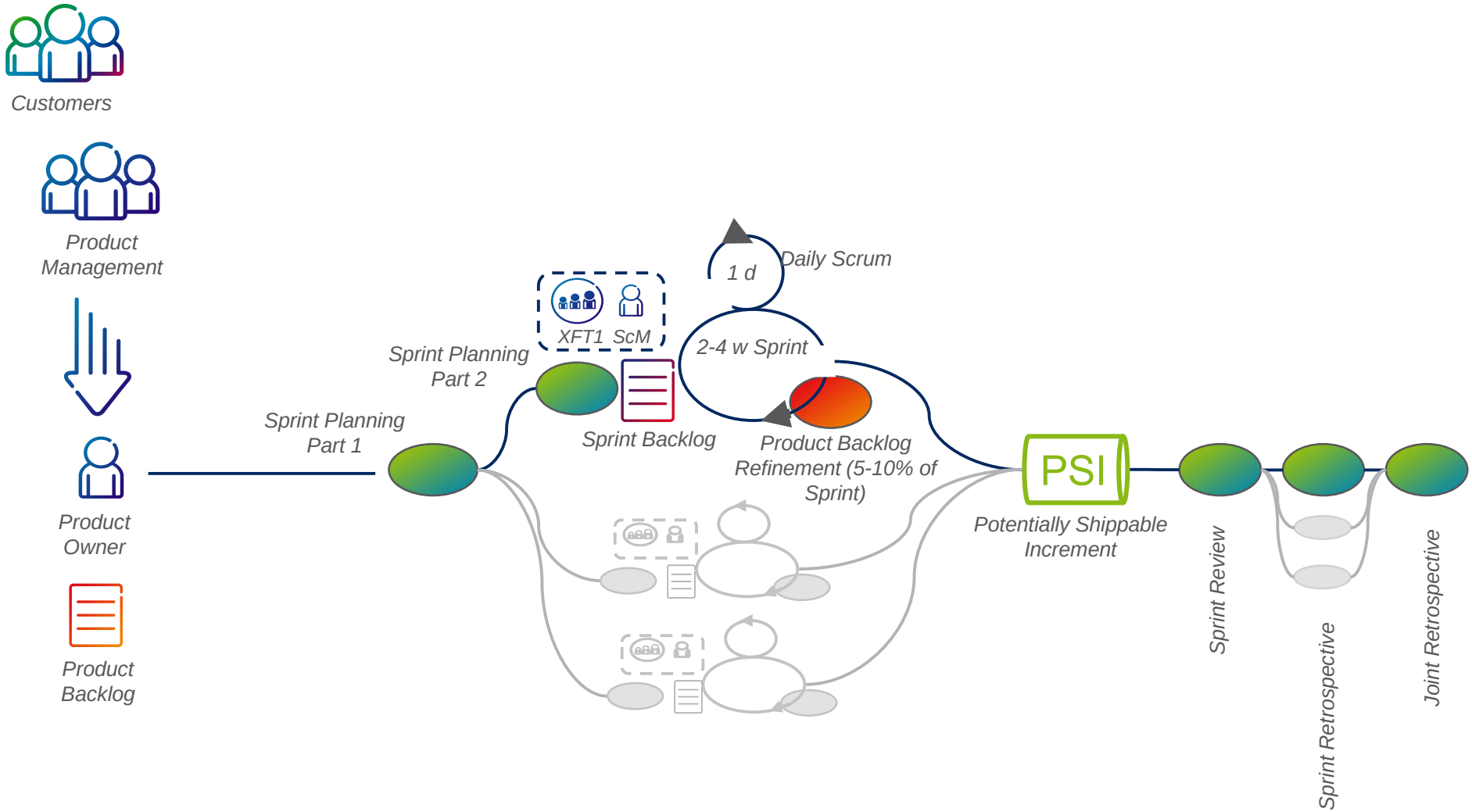
- Are there other alternatives?
- Architecture, infrastructure, …
- Be more efficient with the resources we have

LEARN FROM THE PAST

- How have we done scaling before?
- Can we reuse what worked well?

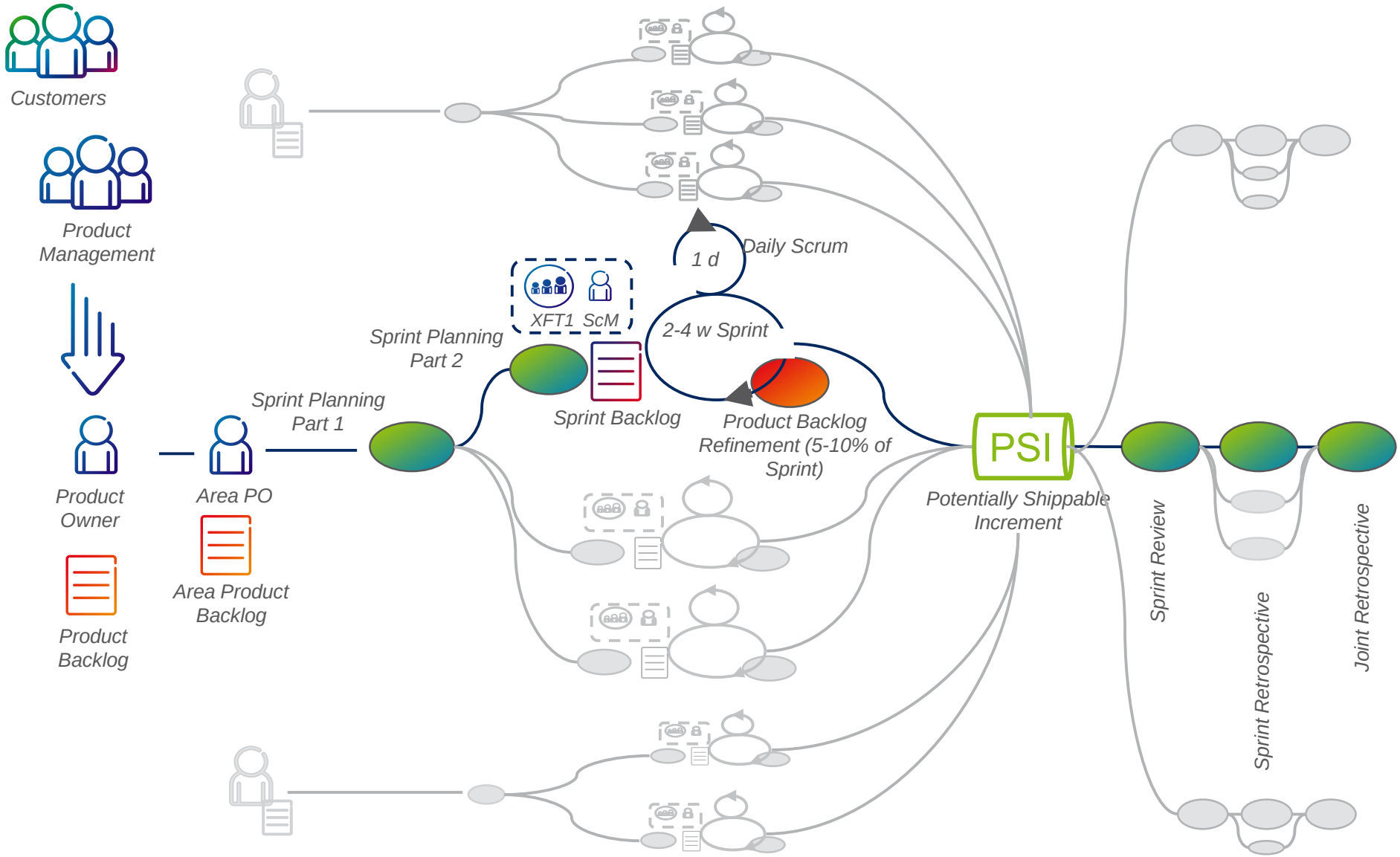
LIMITED NUMBER OF TEAMS;

ONE PRODUCT OWNER



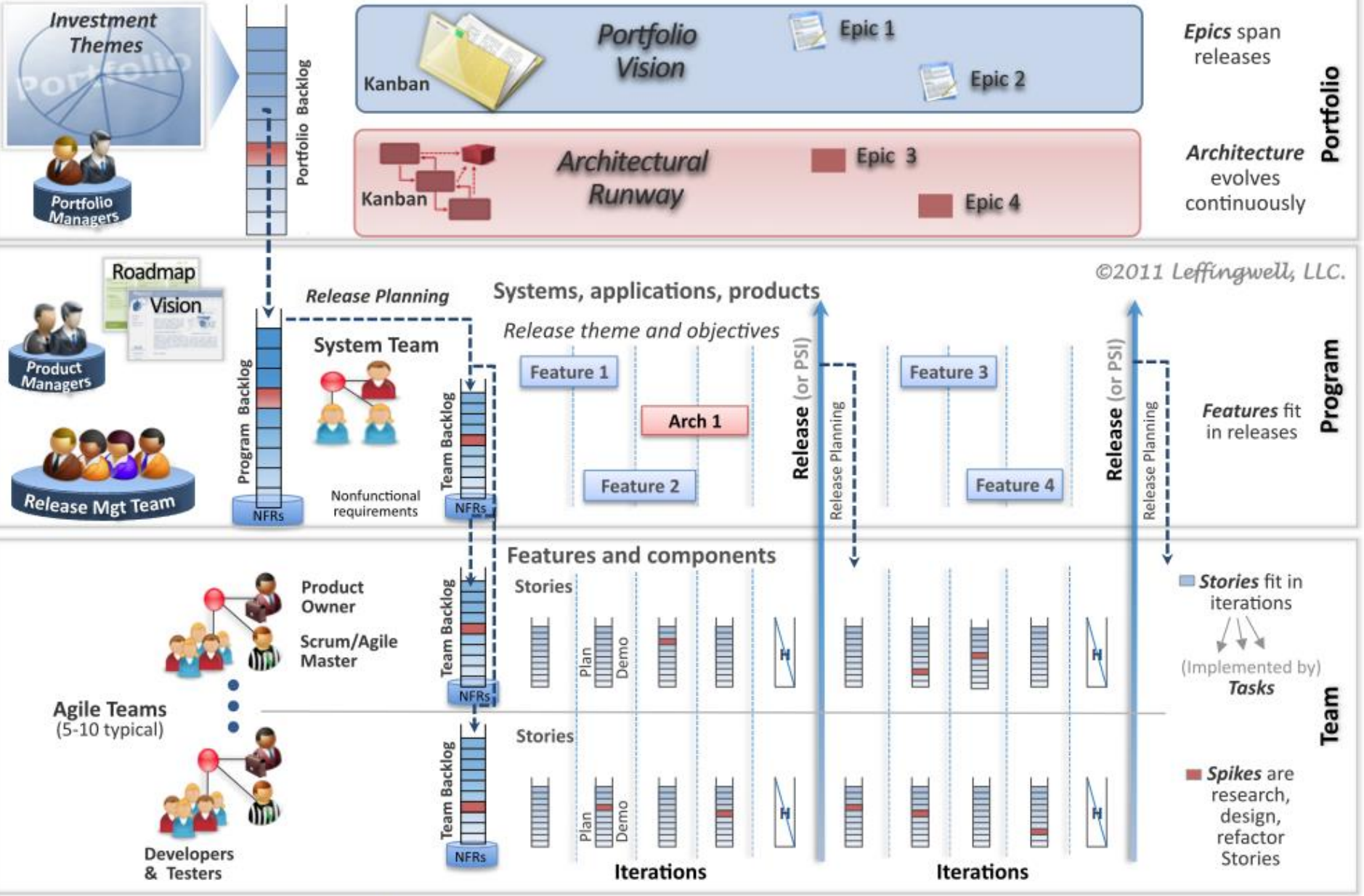
"MANY" TEAMS;

ONE PRODUCT OWNER AND AREA PRODUCT OWNERS



Scaled Agile Framework™ Big Picture

Leffingwell, LLC.

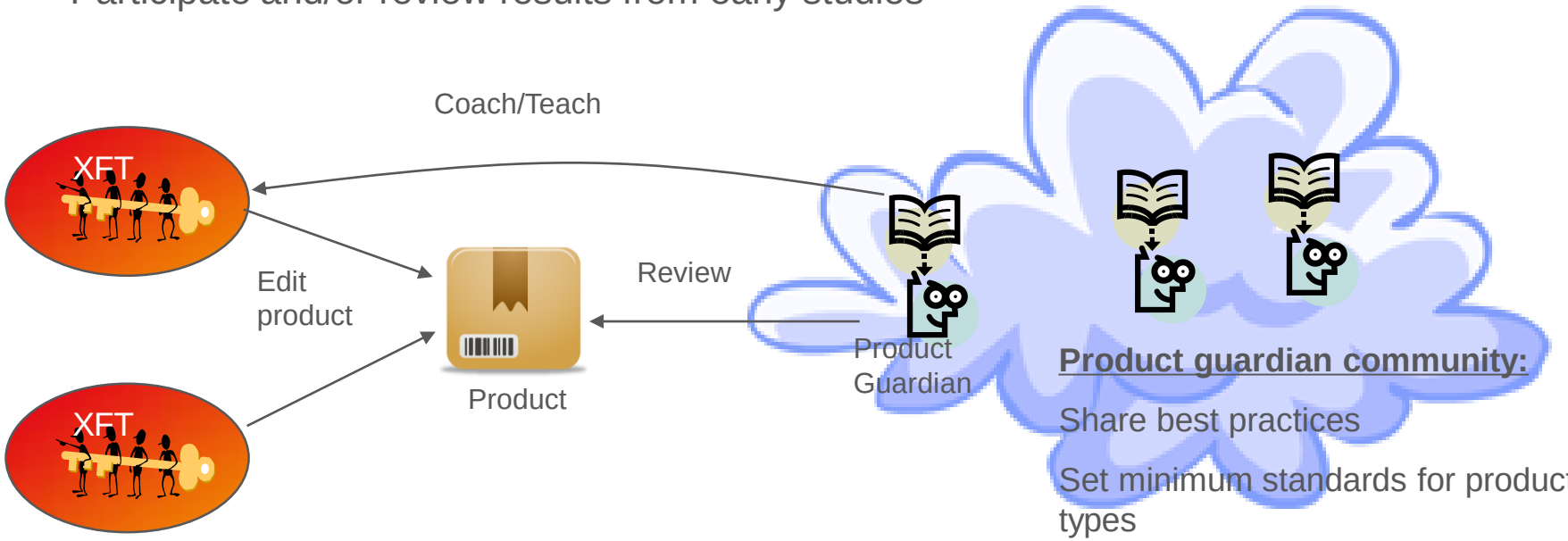


See *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise* and www.scalingsoftwareagility.wordpress.com

SYSTEM INTEGRITY THROUGH PRODUCT GUARDIANS

- Appoint a Product guardian for complex SW blocks, product document, system functionality or CPIs

- The role of the Product guardian is to:
  - Teach and coach XFT members
  - Participate in Design WS, Review code, product document or CPIs
  - Establish design guidelines
  - Collaborate with other PGs and Architects around technical issues and have a long-term vision for the product
  - Participate and/or review results from early studies



COMMUNITIES FOR COLLABORATION

- Communities of Practice (CoP)
- Use communities of practice to collaborate around certain interest and domain areas across teams. E.g. test, products, architecture
- Re-use already existing structures and forums, if applicable

ScM CoP

Impediment handling between teams

Knowledge sharing.

Tech CoP

Technical coordination, Design and Arch guidelines and support

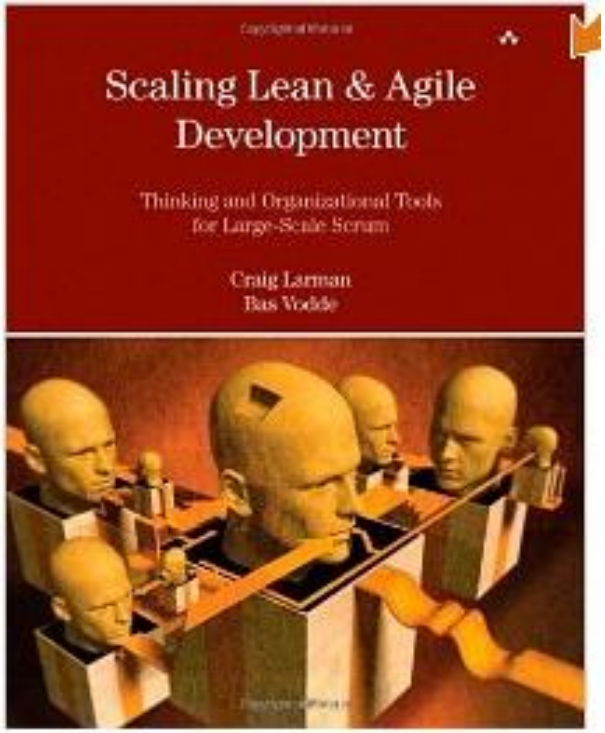
Knowledge sharing.

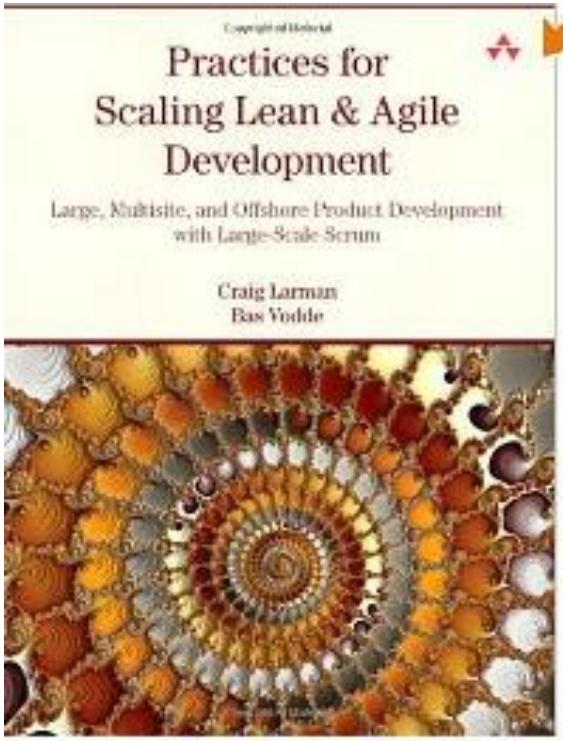
Test CoP

Test strategy and test coordination.

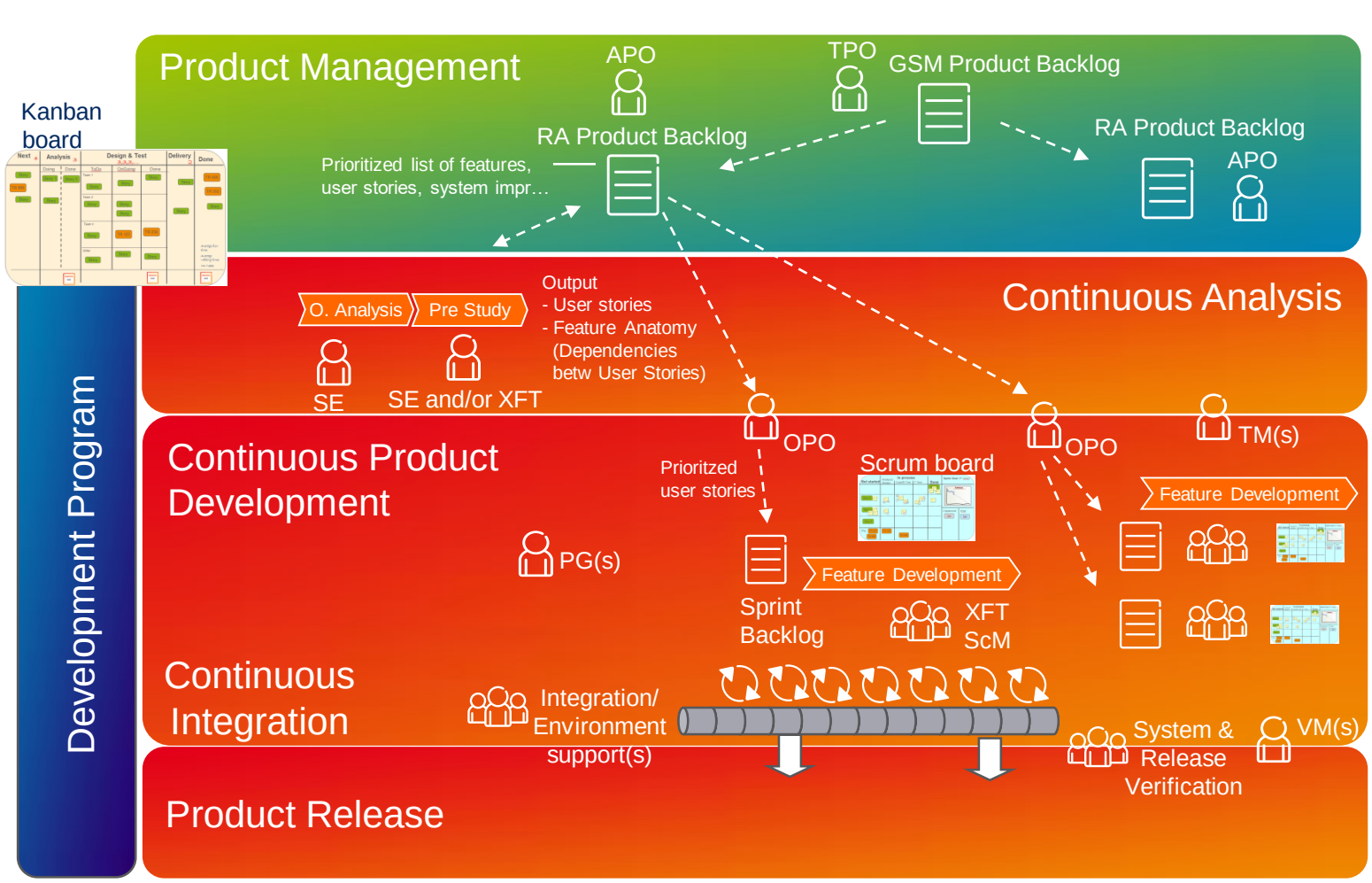
Knowledge sharing.

SOME READING TIPS





BACKLOGS, ROLES & FORA, PROGRESS VISUALIZATION



APO – Area Product Owner
OPO – Operational Product Owner
TM – Technical Manager
VM – Verification Manager
ScM – Scrum Master
XFT – Cross functional team
SE – System Engineer
PG – Product Guardian