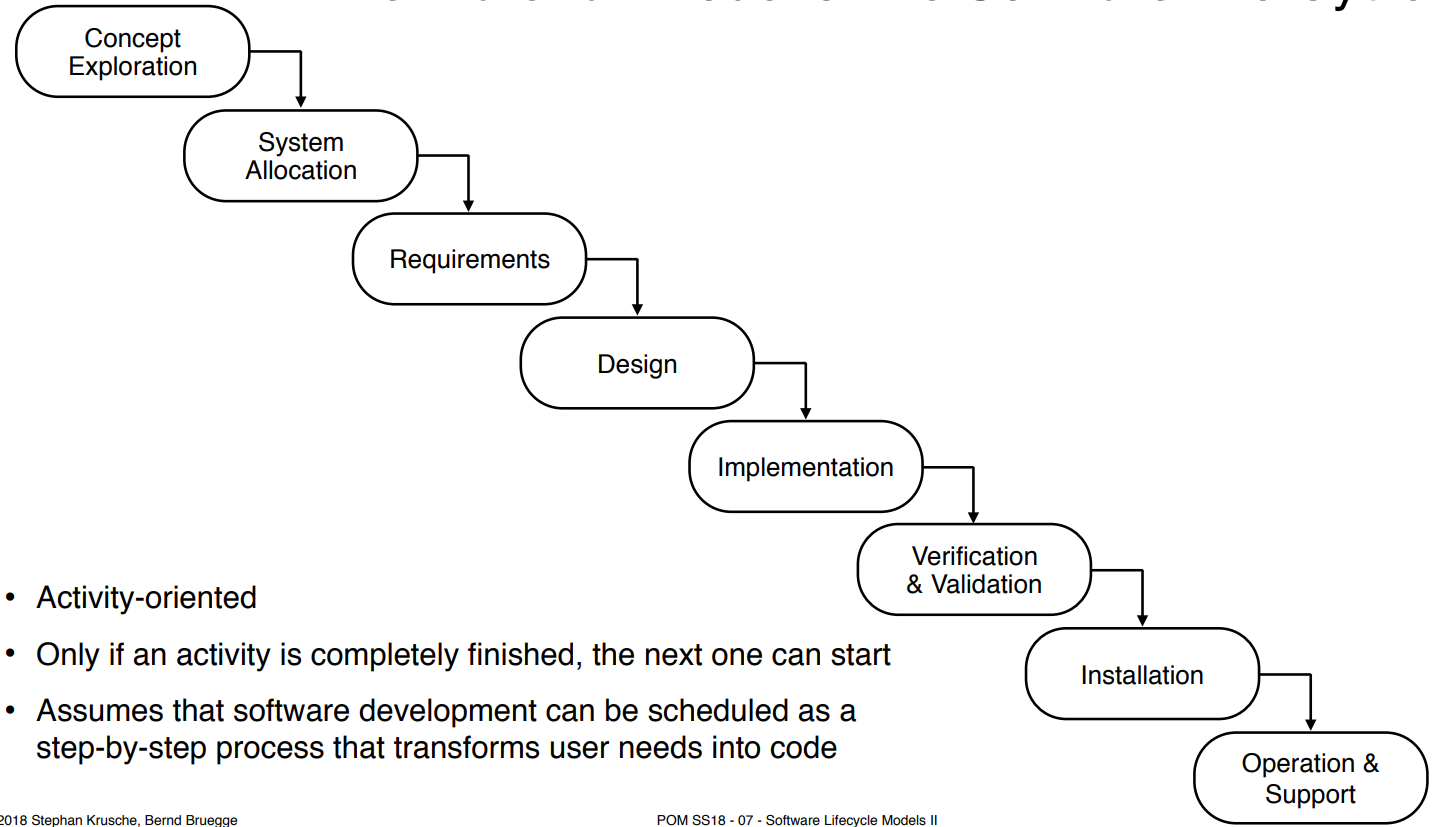
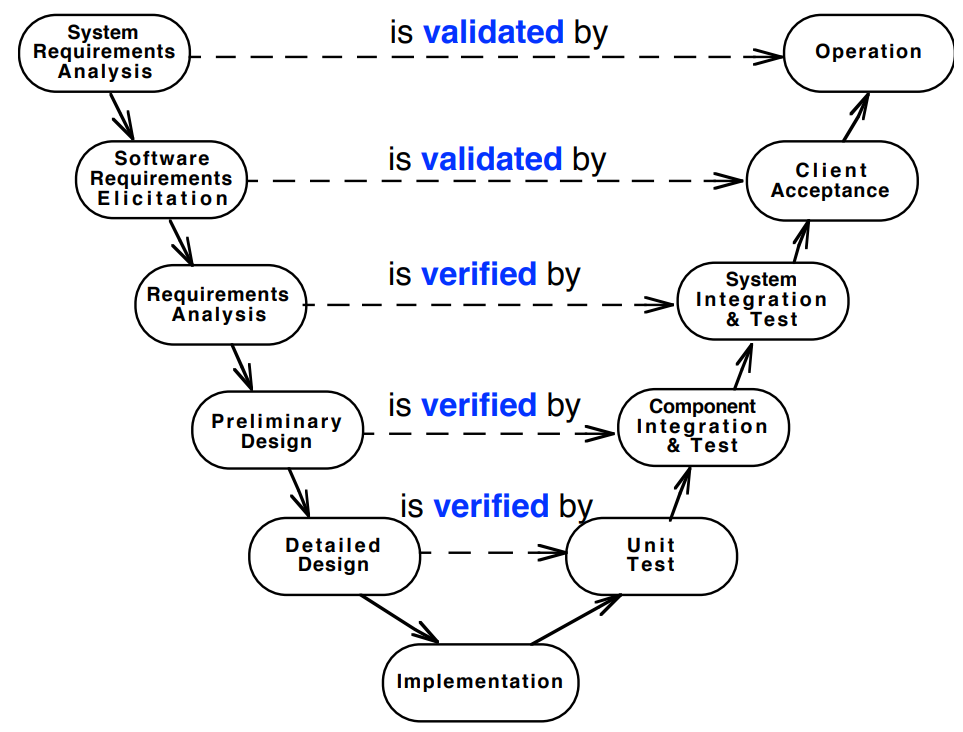
**POM7**

The Waterfall Model of the Software Life Cycle



* The model uses a prescribed sequential execution of activities
* The software development can be scheduled as a linear step by step process
* The goal is to never turn back once an activity is completed
* The key feature of the model is the **verification** activity at the end of each activity that ensures that the activity does not introduce unwanted or deletes mandatory requirements

**Activity Diagram of the V Model**



* The horizontal object flow describes the information flow between activities of same abstraction level

1. **Higher levels** of abstractions of the V-model deal with the requirements in terms of elicitation and operation

➡The client acceptance activity validates the understanding of the user against the requirements

1. **The middle-part** of the V-model focuses on mapping the understanding of the problem into a software architecture

➡The Component Integration and Test activity verifies functional components against the preliminary design

1. **The lower level** of the V-model focuses on details such as the assembly and coding of software components

➡Unit Test activity verifies units against their description in the detailed design

**Validation vs. Verification**

Validation**:**

* Assurance that a product meets the needs of the customer
* Involves acceptance and suitability with external customers
* **Informally: "Are you building the right thing?"**

Verification:

* Evaluation whether or not a product complies with a regulation, requirement, specification or imposed condition
* Often an internal process
* **Informally: "Are you building it right?"**

**Properties of sequential models (Waterfall and V Model)**

* Nice milestones
* No need to look back (linear system)
* Always one activity at a time
* Easy to check progress during development, e.g. 90% coded, 20% tested
* But software development is non-linear!

**Definition: iterative vs. Incremental**

* **Iterative** means “re-do” or “re-work”

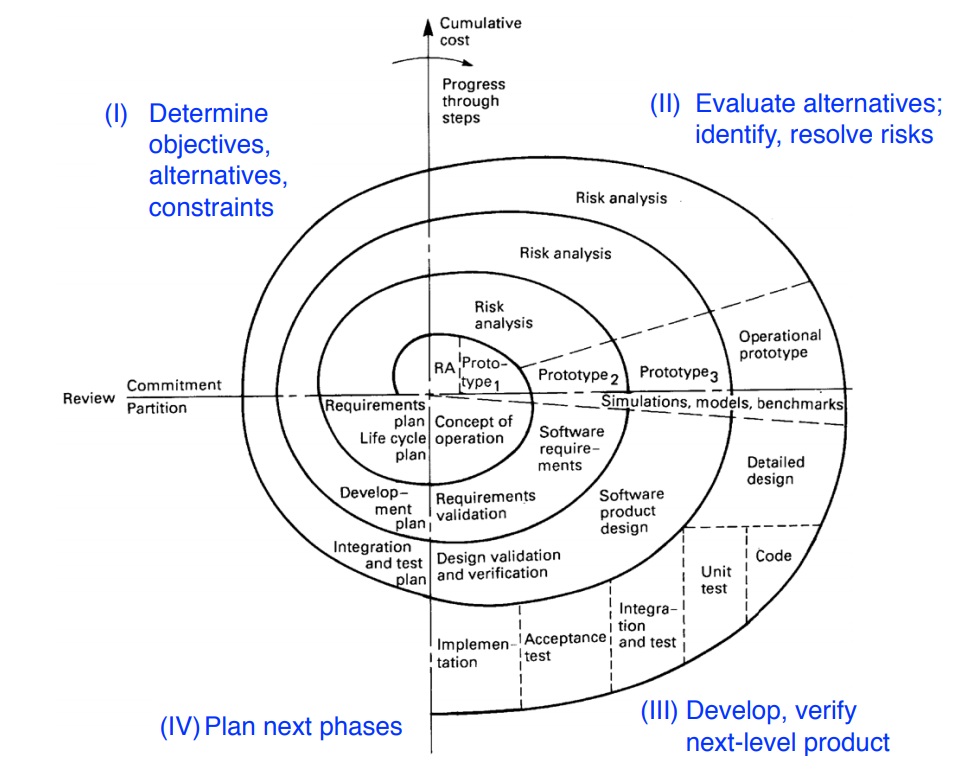
➡ Iterative development helps you to improve your product

* **Incremental** means “to add onto something”

➡ Incremental development helps you improve your process

**Spiral model**

* 9 iterations (cycles)
* 4 basic activities that must be applied in each iteration
  + Determine objectives, alternatives and constraints (start of project, begin of iterations)
  + Identify and evaluate alternatives (risk Analysis)
  + Identify and resolve risks (Concept of Operation Activity)
  + Develop a prototype and obtain approval for the next iteration (Requirements and life cycle planning



**3 types of prototypes**

1. Illustrative prototype

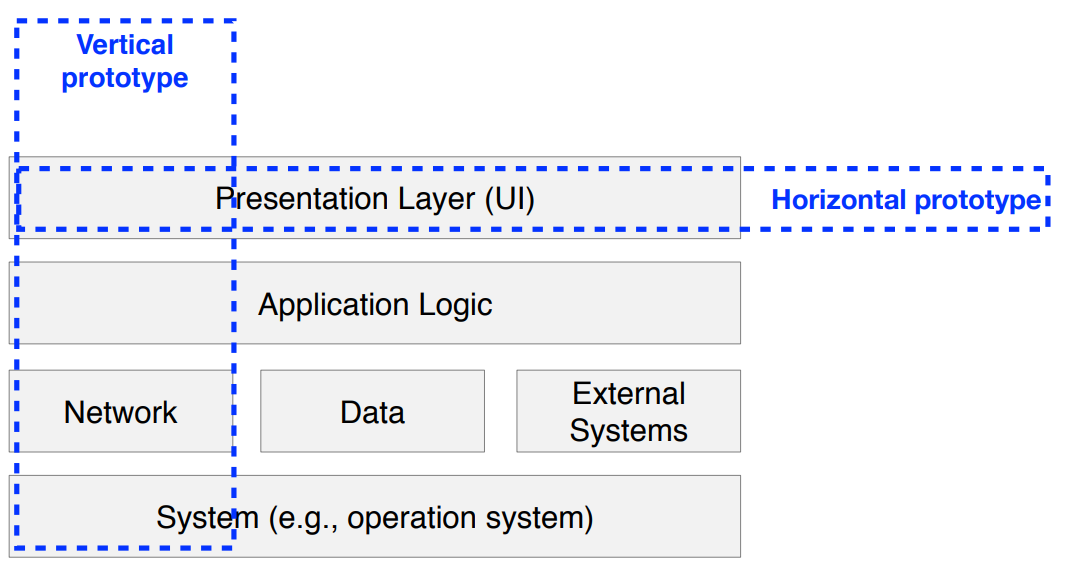
* Develop the user interface with a set of storyboards
* Implement them on a napkin or with a user interface builder
* Good for a first dialog with the client

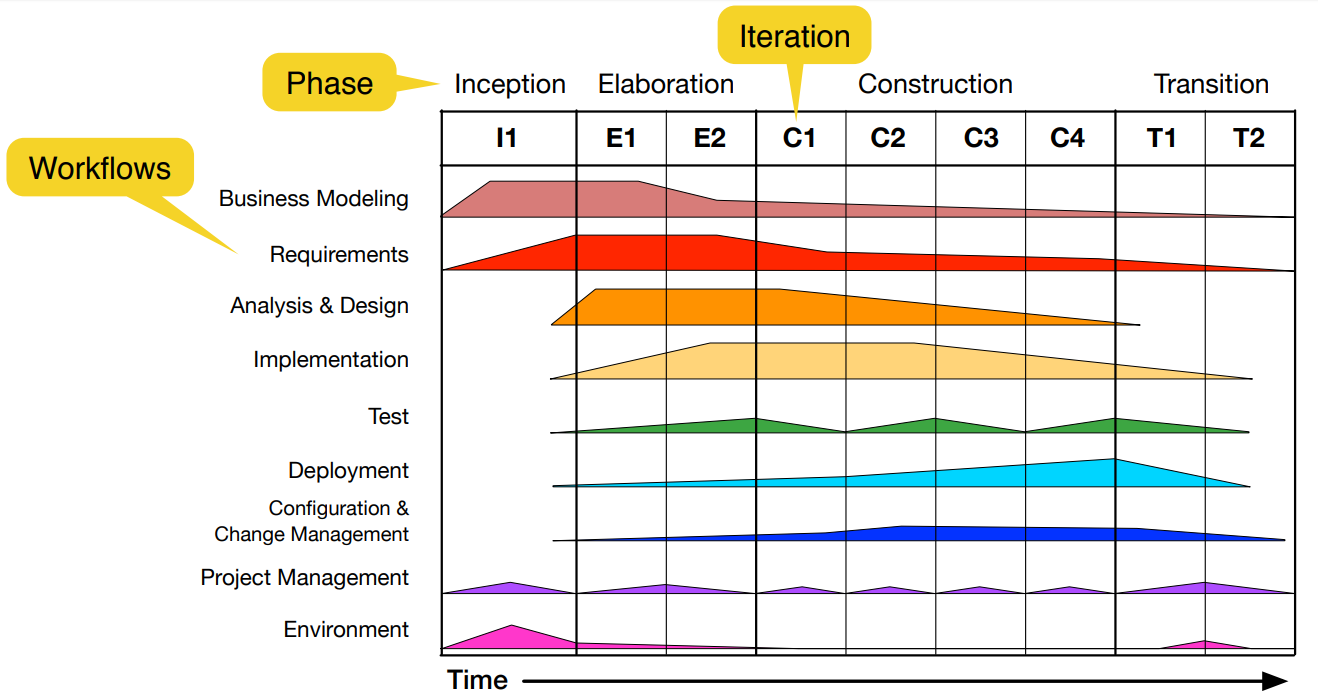
1. 2) Functional prototype

* Implement and deliver an operational system with minimum functionality
* Then add more functionality
* Good for incremental development

1. 3) Exploratory Prototype ("Hack")
   * Implement part of the system to learn more about the requirements

**Vertical vs. horizontal prototype**

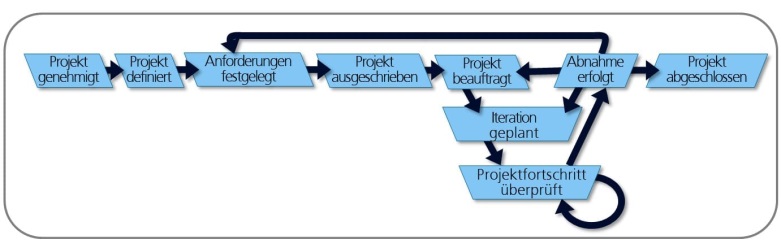


**Unified process**

* An iterative and incremental lifecycle model built on the idea of **cycles** in the lifetime of a software system
* Each cycle consists of 4 **phases**: inception, elaboration, construction, transition
* Each phase can be iterated. During the duration of an iteration, several **workflows** are performed in parallel
  + **6** **core** **workflows**: business modeling, requirements, analysis & design, implementation, test, deployment
  + **3** **supporting** **workflow**: configuration & change management, project management, environment

**V-Model XT**

* The V-Model XT is the successor of the V-Model, XT = eXtreme Tailoring
* Standard for German government contracts
* Goals:
  + Minimization of project risk
  + Improvement and guarantee of quality
  + Reduction of cost
  + Improvement of communication
* Focus on products instead of activities
  + No defined chronology of activities
* V-Model XT describes who has to do what and when in a project
* Each type has a project execution strategy
  + Each strategy is roughly itself a process model, consisting of milestones and decision points

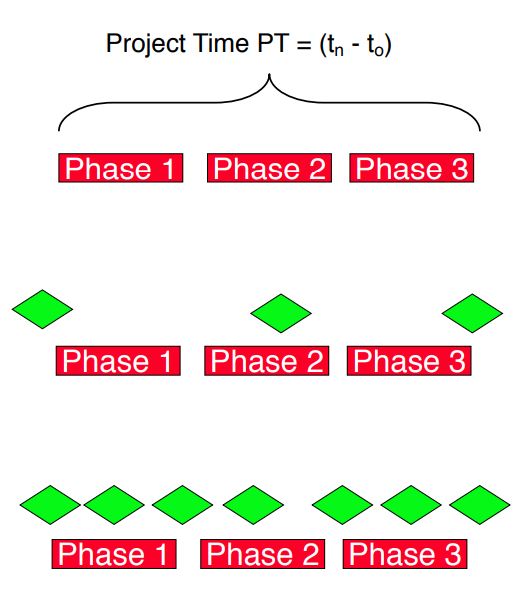


**V-Model XT Tailoring**

* Tailoring means selecting the process components to be used in the project
* Static tailoring occurs at project definition time (before the project actually starts)
  + Project type and project characteristics together suggest a set of process components
  + Project-specific adaptations may involve e.g. using different strategies for prototypes vs. final development etc.
* Dynamic tailoring may occur during project execution
* when architectural design decides to realize some functionality in HW, the process components 'HW development' will be included dynamically

**Frequency of change and choice of software lifecycle model**

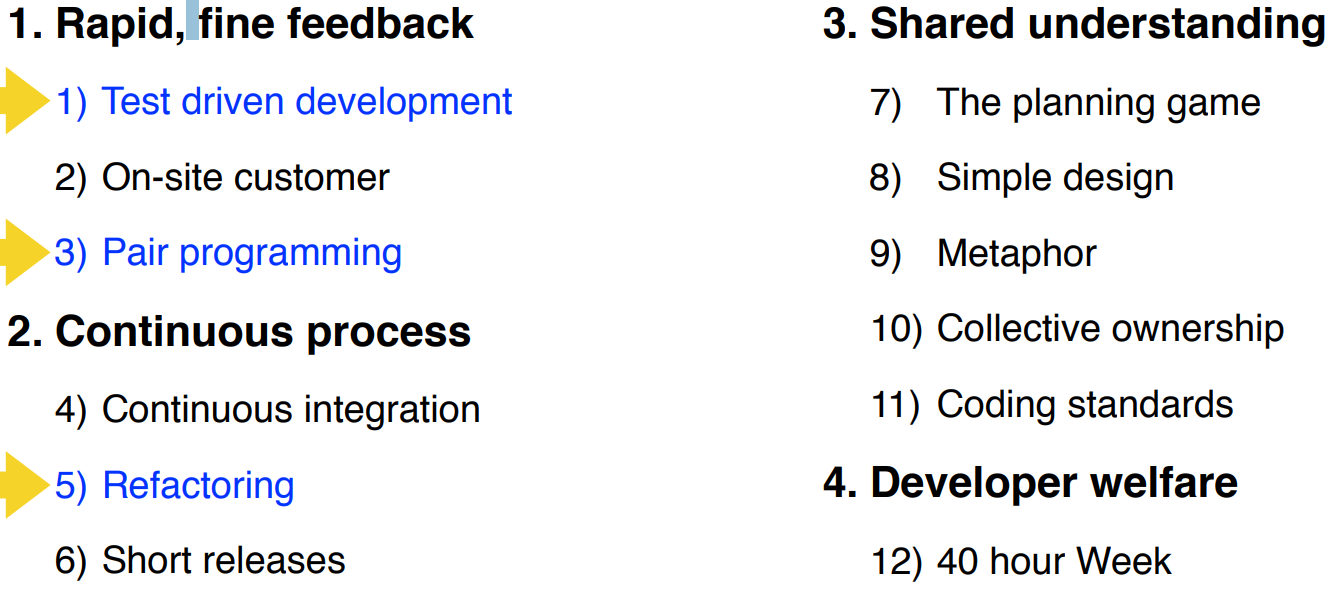
PT = Project

Time MTBC = Mean Time Between Change

* **Change rarely occurs** (MTBC >> PT)
  + **Sequential** model: Waterfall Model, V-Model
  + Open issues are closed before moving to next phase
* **Change occurs sometimes** (MTBC ≈ PT)
  + **Iterative** model: Spiral Model, Unified Process, V-Model XT
  + Change occurring during phase may lead to the iteration of a previous phase or cancellation of the project
* **Change is frequent** (MTBC << PT)
  + **Agile** model: Scrum, Kanban, Extreme Programming (XP)
  + Change during a phase can lead to reengineering the requirements or the design

**Extreme programming (XP)**

* Main goals
  + Avoid over-planning
  + Improve software quality
  + Improve responsiveness to changing customer requirements
* **Terminology**: iteration, deliverable, release
* 5 fundamental principles: rapid feedback, assume simplicity, incremental change, embracing change, quality work
* 4 roles: developer, customer, manager, coach
* 12 practices: how to approach the development process



**XP practice: test driven development (TDD)**

* XP is based on test-driven development (TDD)
* Do not code before you have tests
* Also called **Test first**
* Tests are not only for testing functional requirements, they are also for testing nonfunctional requirements and adherence to standards
* Write only tests that could possibly fail

**XP practice: refactoring**

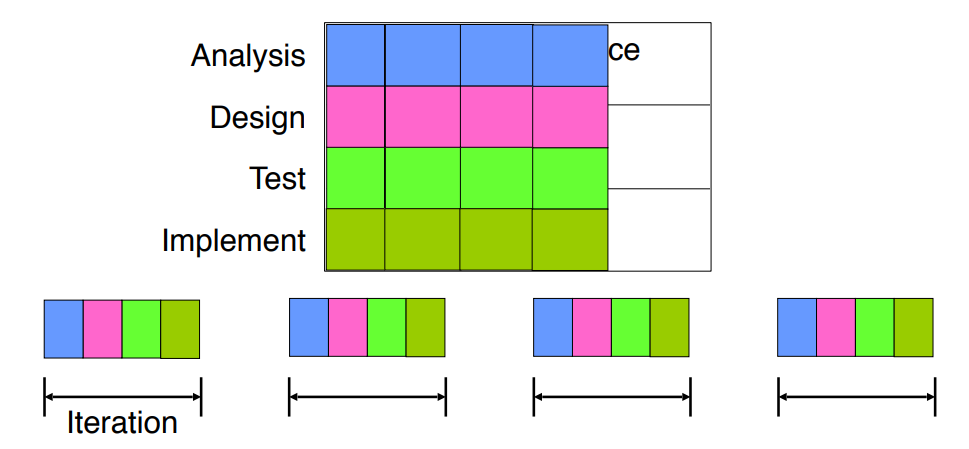
* Refactoring: A change that leaves system behavior unchanged, but enhances simplicity, flexibility, understandability, and/or performance
* You refactor only when the system requires you to do so
* Keep all tests running after refactoring

**XP practice: pair programming**

* + All production code is written with two people looking at one machine
  + There are two roles in each pair:
  1. Driver - Thinks tactically: worries about the implementation, writes the tests and the production code
  2. Navigator (“Observer”) - Thinks strategically: Is this whole approach going to work? What test cases may fail? Can we simplify the system to make this problem go away?

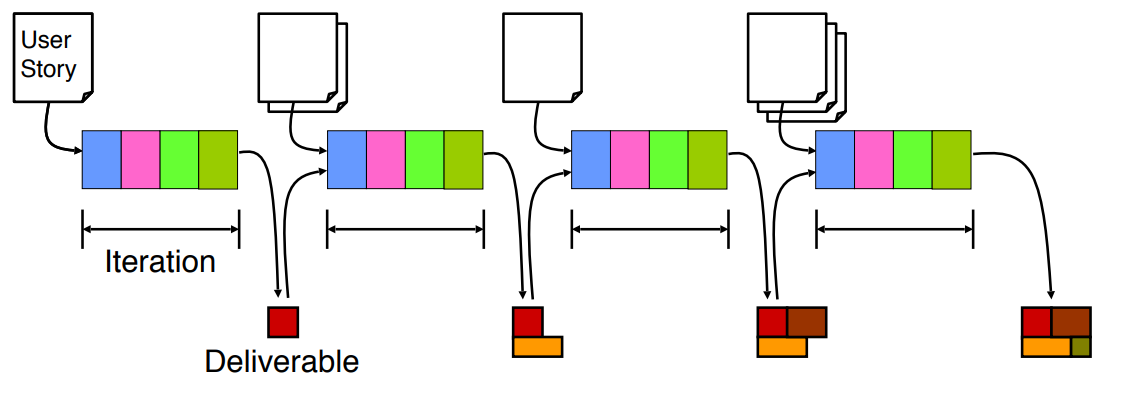
**XP terminology: iteration**

* An iteration implements one or more user stories
  + Iterations are “time-boxed”
  + Iteration finishes always on a fixed date
  + Typical duration of an iteration: 2 to 4 weeks
* Number of implemented features is variable
* (Jeder Iteration alle 4 Bereiche abdecken)



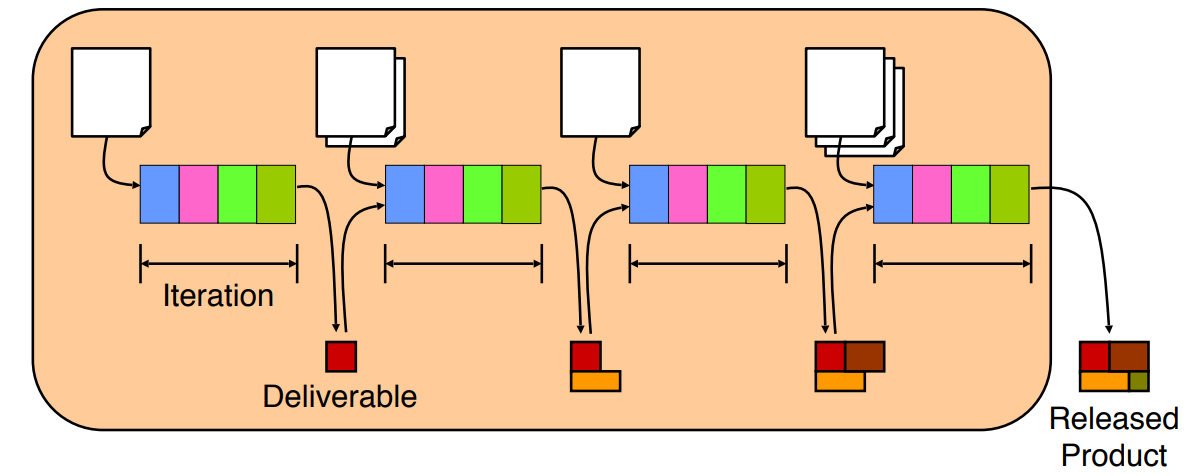
**XP terminology: deliverable**

* Deliverable: result of an iteration
  + Coded, tested, and potentially shippable
* Small addition of functionality



**XP terminology: release**

* A release consists of several iterations that add related functionality
  + Usually every 2 to 6 months
* When should you release?
  + When the customer sees a big improvement over the last release

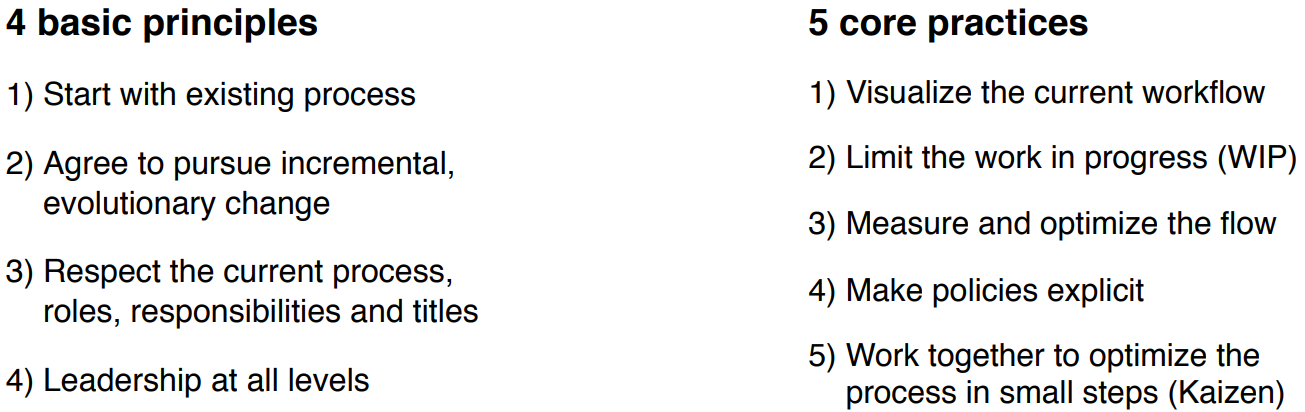


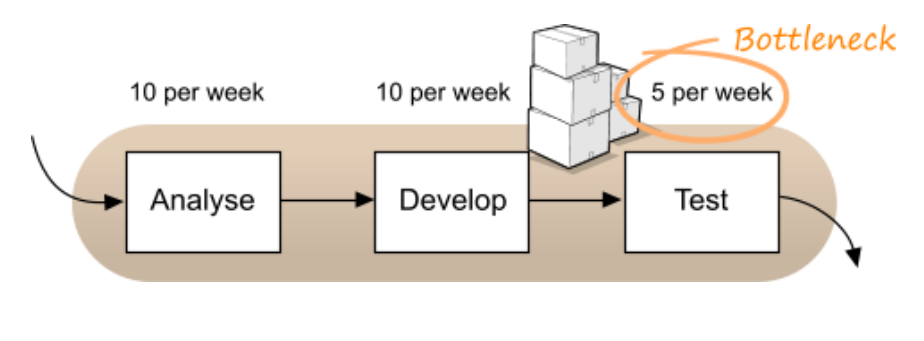
**Kanban**

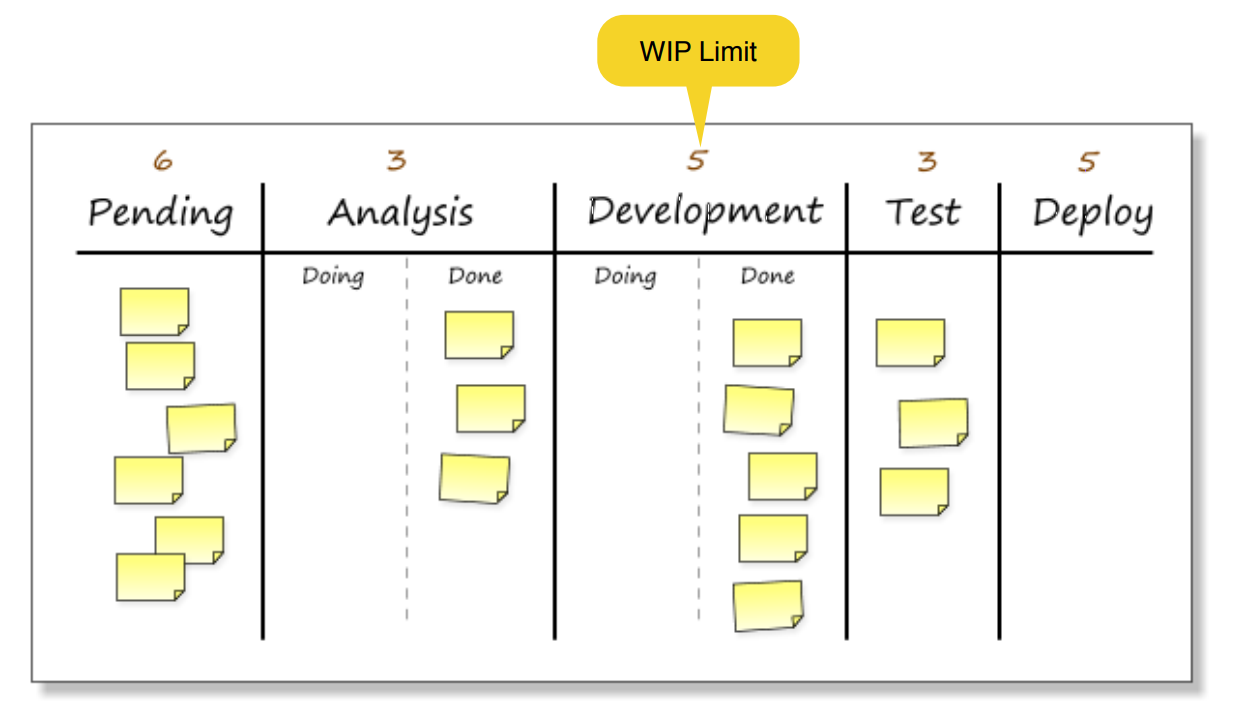
• Stands for transparency, optimization and personal responsibility

• Does not solve problems, it simply makes them visible!

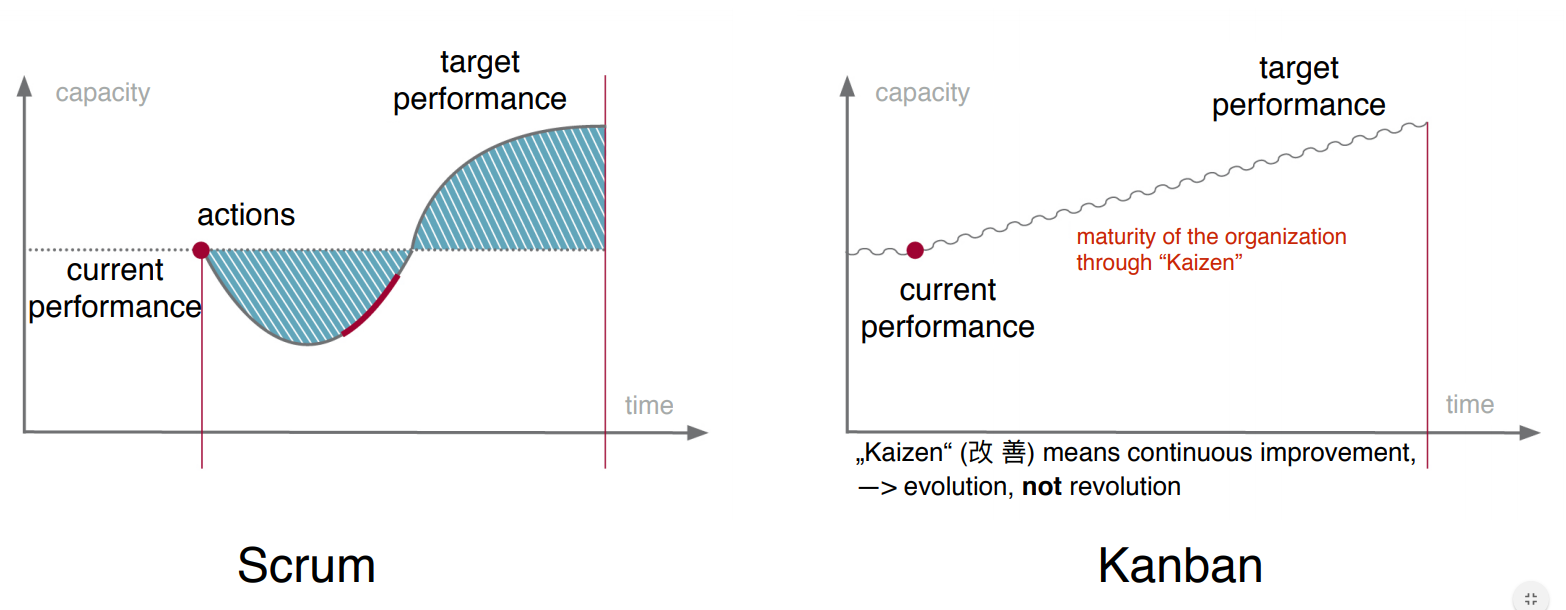
• Begins with the current state and process of a system





????

**Scrum is revolutionary, Kanban is evolutionary (sollte man eig nur einmal gesehen haben)**



**Situations where agile methods might not be appropriate**

* When it is not supported by the company culture
* Project too big for regular complete integration
* Where it inherently takes a long time to get feedback
* Where you can’t realistically test