Class 6

**Requirements in Scrum**

All Requirements are collected in the product backlog

The product owner can be seen as a requirements engineer:

* Elicit, define and prioritize requirements

**Types** of requirements:

* User stories, scenarios, use cases
* Functional and non functional requirements

**User stories**

A user story includes a sentence that describes what the user does or needs

Example: As <role>, I can <feature> so that <reason>

As **sales representative** I can **determine the scale of discount of a customer** so that **I am able to tell him a concrete offer with the correct price**

**Properties of a good user story: INVEST**

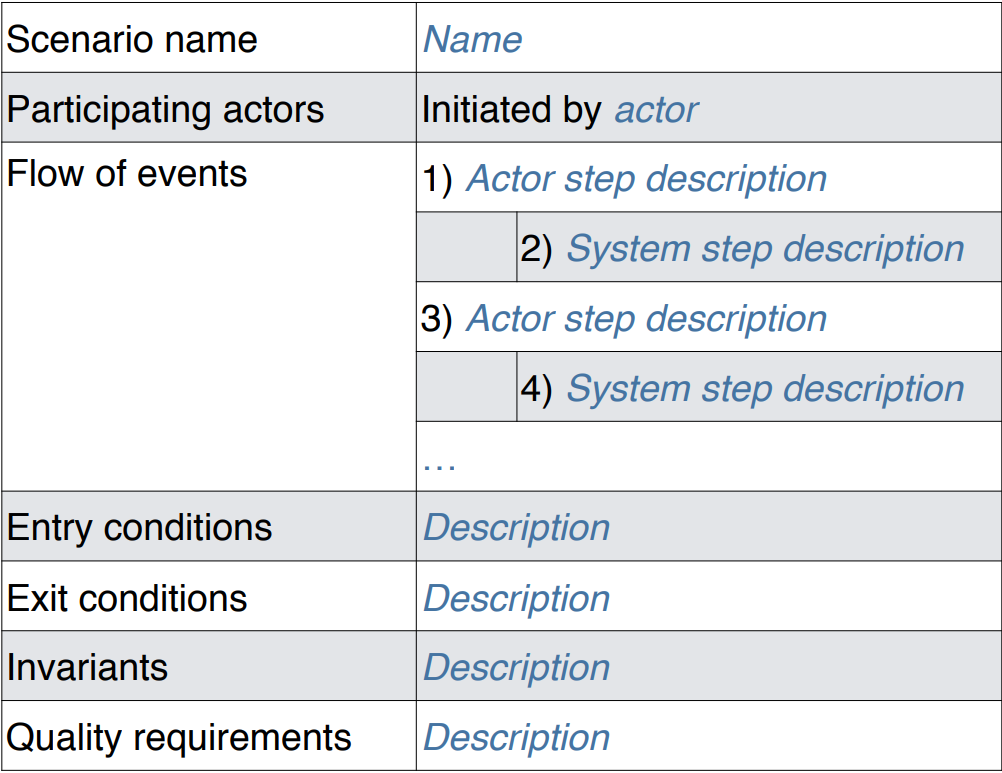
* **Independent** - avoid overlapping user stories
* **Negotiable** - a user story is not a contract, but a basis for discussion between development team and product owner
* **Valuable** - for the user and the business. And **Vertical**: plan and develop features, not layers
* **Estimable** - the stories in the product backlog represent the basis of the project plan
* **Small** - too large user stories must be partitioned into smaller ones to avoid an over-proportional increase of complexity
* **Testable** - if a user story is not testable, it might not be of real value for the product. This also implies realizability

**Acceptance criteria ????**

* Conditions that a software product must satisfy to be accepted by a user, customer or other stakeholder
* Pre-established standards or requirements a product or project must meet
* Set of statements, each with a clear pass/fail result, that specify requirements applicable at the current stage of project integration
  + Functional
  + Non-functional
* These requirements represent “conditions of satisfaction”
* There is no partial acceptance: either a criterion is met or it is not
* Acceptance criteria are typically written on the back of the user story

**Scenarios**

Instance of a use case. A scenario represents a concrete sequence of interactions between one or more actors and the system



**Relation between user stories, use cases and scenarios**

* All focus on functional requirements, but also reference non-functional requirements
* Scenarios are typically created during analysis
* Use cases and scenarios typically cover a larger scope and are more formal than user stories
  + Scenarios and use cases have participating actors, preconditions, event flow, post condition and special quality requirements
* User stories are more informal and typically created during requirements elicitation

**JIRA supports Scrum**

* Create and manage the product backlog
  + Create backlog items (e.g. scenario, user story)
  + Estimate the difficulty
  + Define sub-tasks for the unit of work during development
  + Prioritize backlog items and tasks
* Create and manage sprint backlogs
  + Plan sprints
  + Track progress in active sprints
  + View reports about finished sprints

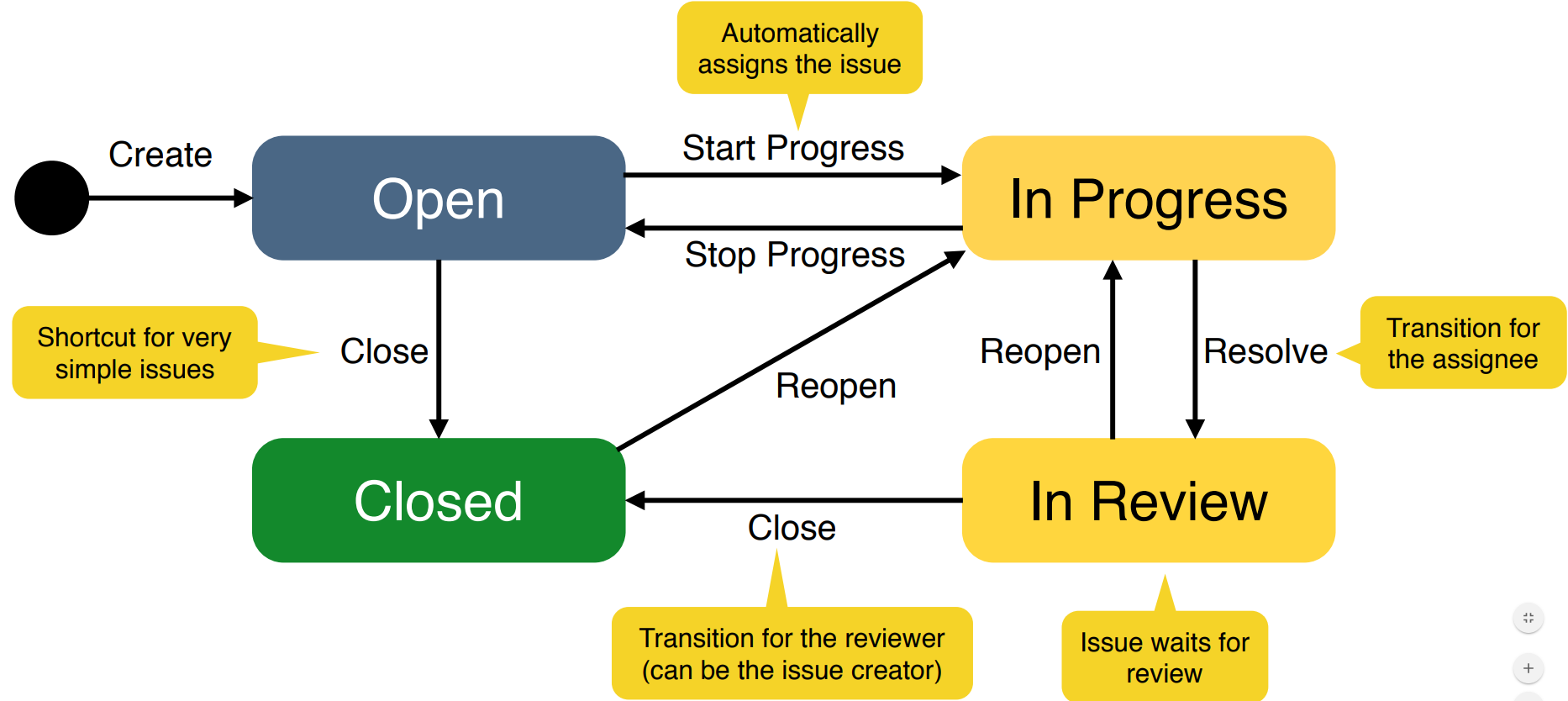
**Sprint planning meeting**

* The development team estimates the difficulty for the items in the product backlog
* Development team and product owner select product backlog items that can be realized in the sprint
* The development team negotiates with the product owner how many items it can realize in the Sprint
* The product owner defines when an item is accepted (e.g. using acceptance criteria)
* Important: The sprint backlog cannot be changed by the product owner within the Sprint to protect the team from too many changes

**Daily scrum meeting**

* Main purpose: Risk reduction by early information sharing and discussion
* 15 min **standup meeting** every day
* Every developer answers the following 3 questions:
  + **Status**: What did you do since the last meeting?
  + **Impediments**: Are there any impediments in your way? (also called blockers)
  + **Promises**: What do you promise to resolve until the next meeting?

**Complete issue workflow**



**Sprint review meeting**

* The development team delivers a product increment including the realized items from the sprint backlog
  + Send it before the review meeting to the product owner
* The development team demonstrates the product increment to the product owner
* The product owner or other stakeholders provide feedback and decide whether the items are realized completely
  + Only completely finished backlog items count towards this sprint
  + Realized items are ticked off in the sprint backlog
* Unrealized items move back to the product backlog
  + These are candidates for the next Sprint
  + Feedback of the product owner (or other stakeholders) is stored in JIRA
* The product owner can add new requirements to the product backlog or change existing ones
* The review meeting can include a sprint retrospective (can also be a separate meeting)

**Potentially shippable product increment**

* Each sprint focuses on the incremental creation of a working system
* The product increment can be thrown away or delivered ➡The product owner decides

**Sprint retrospective meeting**

* Scrum master and development team meet to discuss how the previous sprint worked out ➡ The product owner can participate if necessary
* There are different retrospective techniques ➡ Often it is most effective to brainstorm about things that worked well / did not work well
* Each team member is asked to identify specific things that the team should:
* Start, stop or continue doing

**Scrum as methodology**

**Involvement** **of** **the** **customer**: Onsite customer (“co-located”, product owner)

**Planning**: Checklists and incremental daily plans (“Daily Scrum”)

**Reuse**: Checklists from previous projects

**Modeling**: Models may or may not be used

**Process**: Iterative, incremental and adaptive process

**Control and Monitoring**: Risk management distributed across daily meetings

**Summary**

* Scrum does not define which type of requirements should be used
* Most Scrum teams prefer user stories
* JIRA supports Scrum artifacts and activities
* Create and manage the Product Backlog and its backlog items (e.g. User Stories / Scenarios)
* Plan the Sprint by choosing backlog items and by adding Sub-Tasks
* Track the status of a Sprint using a Taskboard and the progress of a Sprint with a Burn Down Chart
* Prioritize and estimate backlog items —> more about estimation in a later exercise
* Review the Sprint using acceptance criteria