Software Engineering II: Project Organization and Management

Exercise 02 - Agile Methods

Stephan Krusche

08 May 2015

Outline

- Agile Methods: Scrum
  - Artifacts
  - Meetings
  - Roles
  - **Exercise 1:** Creation of the Product Backlog: User Stories and Scenarios
- JIRA Agile
  - **Exercise 2:** Creation of the Product Backlog in JIRA
  - **Exercise 3:** Sprint Planning Part I - Creation of the Sprint Backlog in JIRA
  - **Exercise 4:** Sprint Planning Part II - Refinement of the Sprint Backlog in JIRA
  - **Exercise 5:** Working on the Taskboard during the Sprint
  - **Exercise 6:** Sprint Review

Controlling Software Development with a Process

- How do we control software development? Two opinions:
- Through organizational maturity (Humphrey 1989)
  - Repeatable process, Capability Maturity Model Integration (CMMI)
- Through agility (Schwaber 2001):
  - Large parts of software development is empirical in nature; they cannot be modeled with a defined process
- There is a difference between a defined and an empirical process
- How can software development better be described?
  - with a defined process control model or
  - with an empirical process control model?

more in Lecture 11
CMMI on June 24

Defined vs. Empirical Process

Defined Process

Planned

Follows strict rules

Avoids deviations

Empirical Process

Not entirely planned

inspect and adapt

Defined Process Control Model

- Requires that every piece of work is completely understood
- Deviations are seen as errors that need to be corrected
- Given a well-defined set of inputs, the same outputs are generated every time
- Precondition to apply this model:
  - All the activities and tasks are well defined to provide repeatability and predictability
- If the preconditions are not satisfied:
  - Lot of surprises, loss of control, incomplete or wrong work products.

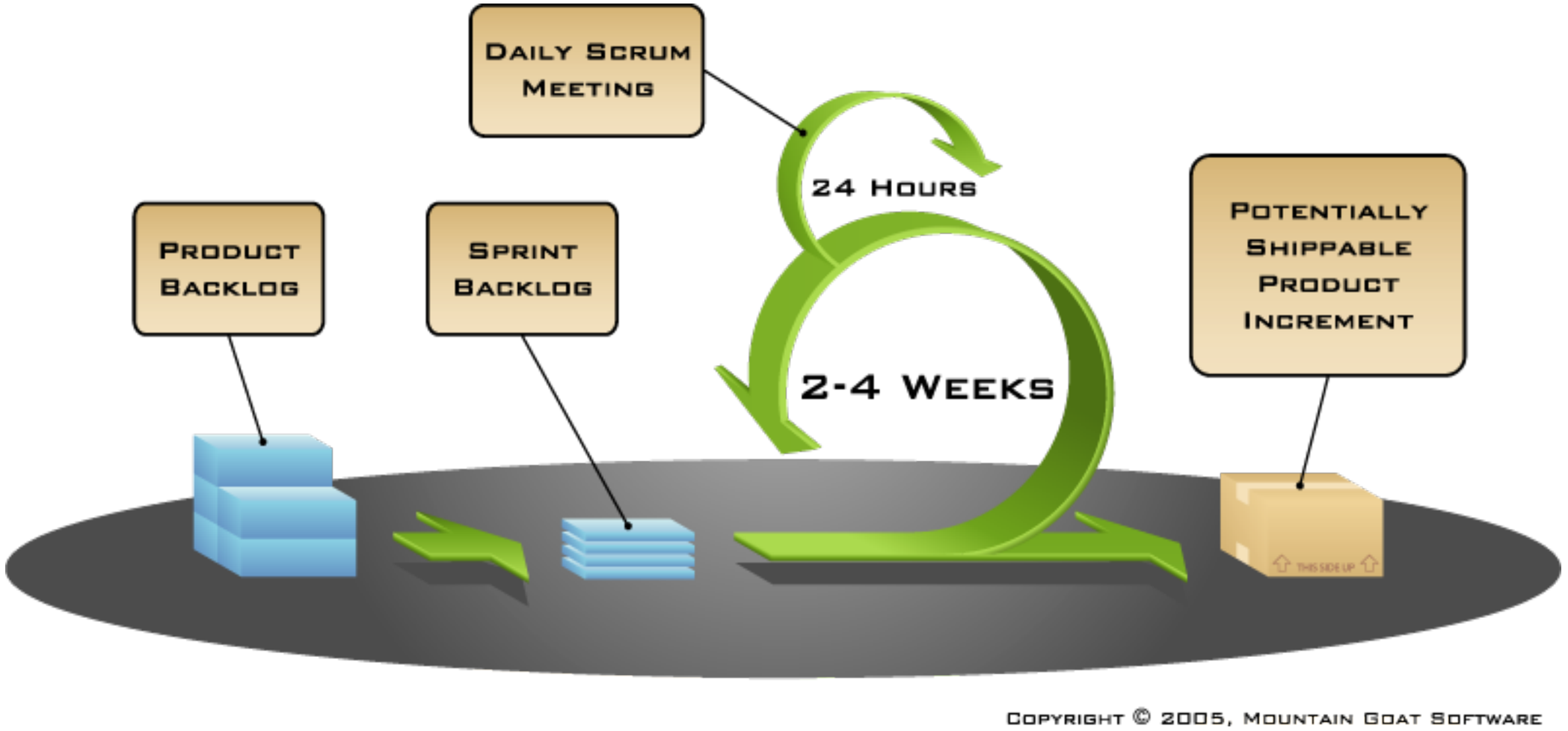
Empirical Process Control Model

- Empirical process: An imperfectly defined process, not all pieces of work are completely understood
- Deviations, errors and failures are seen as opportunities that need to be investigated
  - The empirical process “expects the unexpected”
- Control and risk management is exercised through frequent inspection
- Condition when to apply this model:
  - Change is frequent and cannot be ignored
  - Change of requirements, change of technology, change in the organization, people change too.

Scrum is based on an Empirical Process Control Model

- Original definition (from Rugby): A Scrum is a way to restart the game after an interruption
  - The forwards of each side come together in a tight formation and try to get the ball when it is tossed in among them
- Definition in agile processes: Scrum is a technique that deals with interruptions (change)
  - Manages and controls software and product development with rapidly changing requirements
  - Improves risk management by improved communication and cooperation and the delivery of product increments.

Example of an Empirical Process: Scrum



Scrum Artifacts

- Product Backlog: List of requirements for the whole product
- Sprint Backlog: List of requirements and tasks for one iteration (“Sprint”)
- Potentially Shippable Product Increment: Release to the Product Owner that contains all results of the current Sprint

Scrum Meetings

- Project Kickoff Meeting (Start of the Project): Create and prioritize Product Backlog
- Sprint Planning Meeting (Start of each Sprint): Create Sprint Backlog
- Daily Scrum Meeting (Every Morning,15min): Standup meeting to share status, impediments and promises
- Sprint Review Meeting (End of each Sprint): Demonstration of realized backlog items to the Product Owner (and other stakeholders)

Scrum-Team

Development Team
Self-organizing and cross-functional
Realizes the product increment

Product Owner
Defines the product
Responsible for results

Scrum Master
Resolves impediments
Responsible for the process

Sprint - Definition

„A time-box of one month or less during which a “Done”, useable, and potentially releasable product increment is created. Sprints best have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint.“

[scrumguides.org]

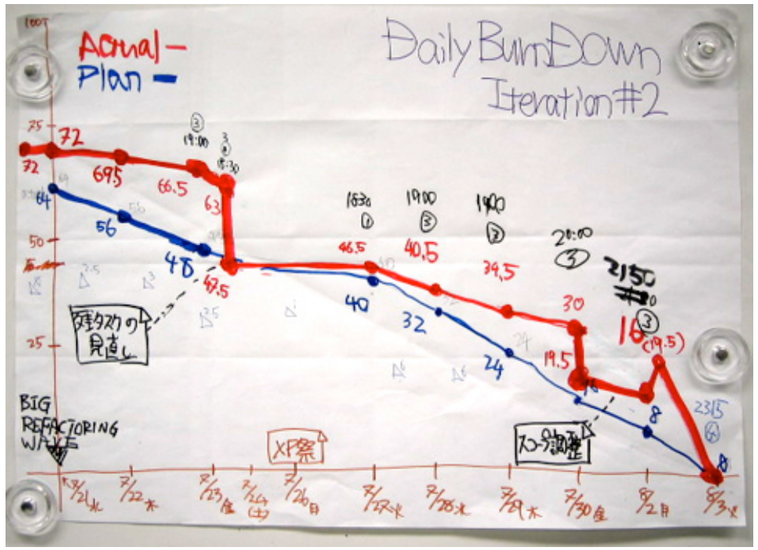
Sprint

- Typically 2-4 weeks long
- Starts with Sprint Planning Meeting
  - **Create the Sprint Backlog:** Selection of items to be implemented in the Sprint
  - **Important:** Development Team and Product Owner select the items together
- Ends with Sprint Review Meeting
  - Release and deliver the application (product increment)
  - Demo of the application during the meeting
  - **Important:** Product Owner gives feedback
- Sprint Review Meeting and Sprint Planning Meetings can be combined into a single meeting with the Product Owner
- The Scrum Team can additionally perform a Sprint Retrospective Meeting

Sprint

- The Development Team
  - Works on the items in the Sprint Backlog
  - Uses e.g. a Task Board to visualize the status of these items
- The Scrum Master visualizes the progress, e.g. in a Burn Down Chart

Items	Todo	In Progress	In Review	Done
User Story 1	Task	Task	Task	Task
User Interface Design	Task	Task	Task	Task
Scenario 1	Task	Task		



Product Backlog

- Collection of items (e.g. user stories, scenarios, etc.) prioritized by the Product Owner
- The Product Backlog can always be changed and reprioritized during the projects
- Created on the basis of the problem statement during the Kickoff Meeting or in the phase before the actual project starts

Problem Statement

Kickoff Meeting

Prioritized Product Backlog

ID	Name	Priority
1	Backlog Item 1	Critical
2	Backlog Item 2	Critical
3	Backlog Item 3	Major
4	Backlog Item 4	Minor
5	Backlog Item 5	Minor

Priority

- The priority describes the importance of the requirement for the software to be developed
- If the requirement has a high value for the product (e.g. because it is the core functionality of an application), it has a high priority
- Examples for a priority scheme (that we also use in JIRA)
  - **Prio 1 = Critical** (Candidates for the first development sprints, must be part of the first product increment)
  - **Prio 2 = Major** (Must be realized within in the project, can be realized in one of the following product increments)
  - **Prio 3 = Minor** (Desirable, if there is enough time)
  - **Prio 4 = Not Important** (Might not be realized)

➡ Priorities of requirements can change during the project in Scrum

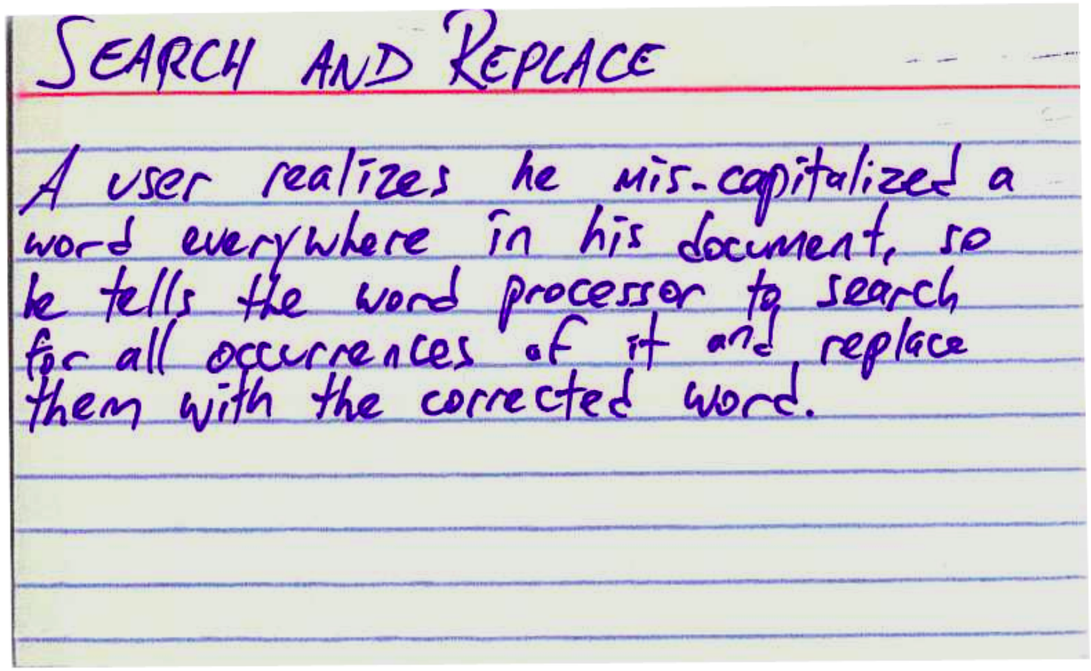
➡ Prioritization is done by the Product Owner

User Stories

A promise for conversation

Example for backlog items: User Stories

- A User Story includes a sentence that describes what the user does or needs
- Often written on a card
- Users (roles) are the actual users of the system



Properties of a good User Story: INVEST

- **I**ndependent - avoid overlapping User Stories
- **N**egotiable - A User Story is not a contract, but a basis for a discussion between Development Team and the Product Owner
- **V**aluable - for the user and the business. And **V**ertical: we plan and develop features and not layers
- **E**stimable - the stories on the Product Backlog represent the basis of our project plan
- **S**mall - too large user stories should be partitioned into smaller ones, because the complexity increases over-proportional otherwise
- **T**estable - if a User Story is not testable, it might not be a real value for the product, this also implies **realizability**

User Story Template

As <role>,

I want to <feature>

so that <reason>

Main advantages of the template:

- Using the “I” phrasing, developer can identify with the given user role
- Defines a structure and simplifies the prioritization process
- Facilitates categorization according to user roles

Example of a User Story

As	customer
I want	a flexible possibility to store data in my database
so that	That's clear

Independent	
Negotiable	
Valuable	
Estimable	
Small	
Testable	

➡ **bad example**

Example of a User Story (2)

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

Independent	
Negotiable	✅
Valuable	✅
Estimable	✅
Small	✅
Testable	✅

➡ good example

Example of a User Story (3)

As	student
I want	to find my grades online
so that	I don't have to wait until the next day to know whether I passed the exam

Independent	
Negotiable	✅
Valuable	✅
Estimable	✅
Small	✅
Testable	✅

➡ good example

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
- Acceptance Criteria are a set of statements, each with a clear pass/fail result, that specify requirements applicable at the current stage of project integration
  - Functional (e.g. minimal marketable functionality)
  - Non-functional (e.g. minimal quality)
- These requirements represent “conditions of satisfaction.” There is no partial acceptance: either a criterion is met or it is not.

Definition of Done

- The Definition of Done (DoD) is usually a clear and concise list of requirements that a software Increment must adhere to for the team to call it complete
- Until this list is satisfied, a product Increment is not done

➡ Before a piece of functionality is in a potentially releasable state, it must adhere to a common understanding of completion by those working on the Increment

- Examples:
  - End-User Documentation is updated
  - Unit testing is complete and passed
  - Developer has peer reviewed code
  - Integration Testing is complete and passed
  - Configuration Management has added the functionality to the build and captured release notes
  - Product Owner has reviewed the demo and approved the User Story
  - User Acceptance Testing is complete and passed

Scenarios

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

Scenario Template

Scenario name	*Name*
Participating actors	Initiated by *actor*
Flow of events	1) *Actor step description*
	2) *System step description*
	3) *Actor step description*
	4) *System step description*
	…
Entry conditions	*Description*
Exit conditions	*Description*
Invariants	*Description*
Quality requirements	*Description*

Quality requirements are also called non-functional requirements

Scenario Example

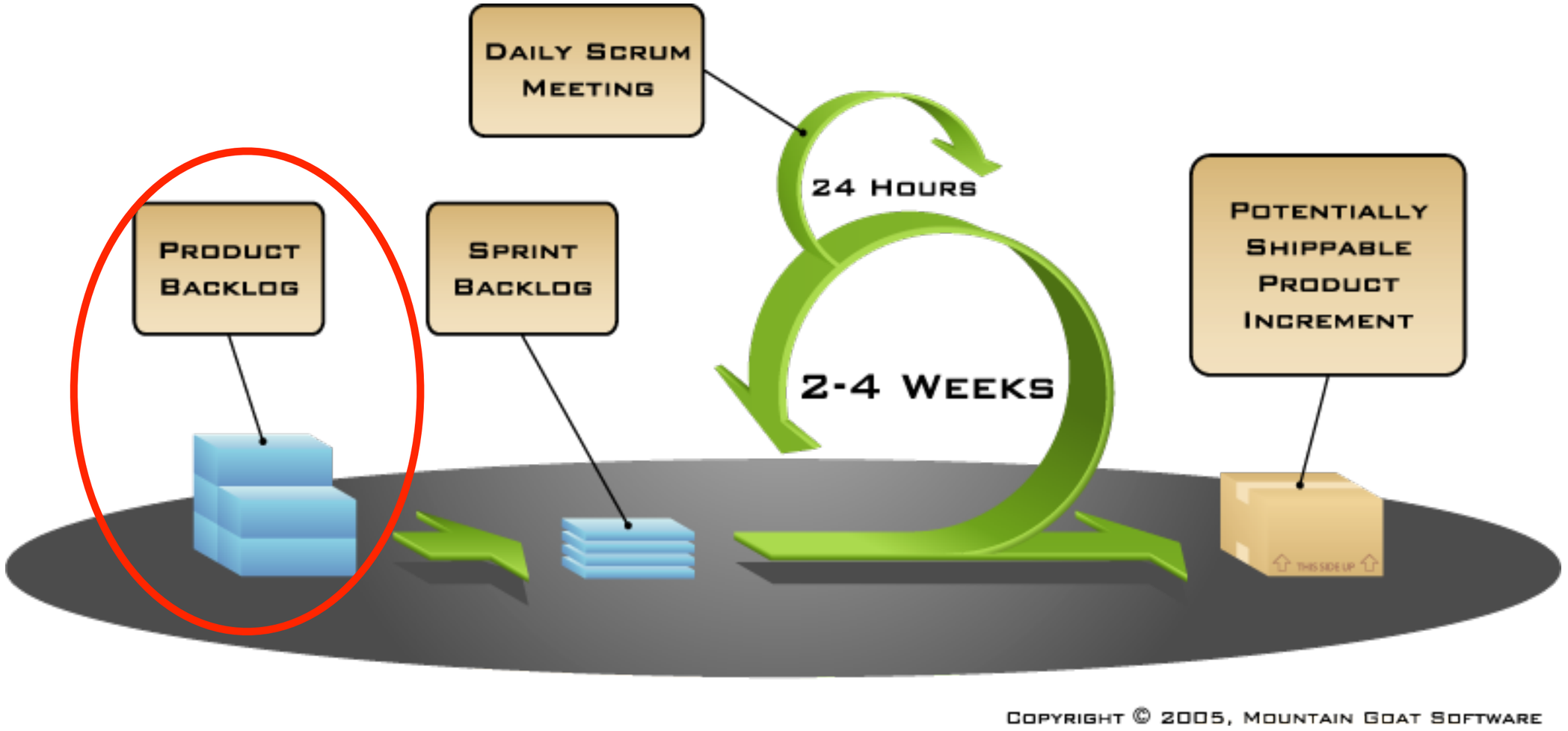
Scenario name	*Select Pedelec*
Participating actors	Initiated by *Marie, employee*
Flow of events	1) *Marie opens the Pedelec App*
	2) *The system shows a map with Marie's current location in the center of the screen and 3 Pedelecs near to Marie*
	3) *Marie chooses the Pedelec that is closest to her*
	4) *The app shows the range (working radius) of the chosen Pedelec as a dashed circle with the Pedelec as center*
Entry conditions	*3 Pedelecs are near to Marie*
Exit conditions	*The Pedelec is selected*
Invariants	*-*
Quality requirements	*Marie can select the Pedelec in 1 click*

Relation between User Stories, Use Cases and Scenarios

- 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
  - Scenarios are instantiated use cases
  - More details about scenarios and use cases in chapter 4 of Object Oriented Software Engineering using UML by Bruegge and Dutoit
- User Stories are usually created during requirements elicitation

Source: Bernd Bruegge, Allen H. Dutoit, Object Oriented Software Engineering using UML, Design Patterns and Java

Exercise 1: Creation of the Product Backlog



The Pedelec App will be our running example problem statement for the exercises. **Notice:** You cannot choose it for the team project!

Time: 10 min

Exercise 1: Creation of the Product Backlog: User Stories and Scenarios

Task 1: Read the Pedelec App Problem Statement and use it for the following tasks
Task 2: Create 3 User Stories and write them into a Text Editor
Task 3: Create 1 small Scenario and write it into a Text Editor
Hint: Write user stories and scenarios for the functional requirements
Task 4: Prioritize these Product Backlog Items

Product Backlog: Example Solution - User Stories

ID	Summary	User Story	Priority
1	☐ Search available Pedelecs	As employee, I can search Pedelecs on a map so that I know where free Pedelecs are located	Critical
2	☐ Check working radius	As employee, I can check the working radius of a Pedelec so that I know whether I can reach my destination with it	Critical
3	☐ Reserve available Pedelec	As employee, I can reserve an available Pedelec for a certain time-slot so that I am sure that I can use it during this time-slot	Critical
4	☐ Return Pedelec	As employee, I can return a Pedelec after my trip so that others can again reserve it	Major
5	☐ Contact colleague	As employee, I can contact a colleague to obtain the reservation of his bike so that he knows that I need the bike urgently	Minor
6	☐ Pass reservation	As employee, I can pass my reservation of a Pedelec to my colleague so that he can obtain it	Minor
7	☐ Report damage	As employee, I can report the damage of a Pedelec so that the Pedelec can be repaired	Minor
8	☐ Unlock Pedelec	As employee, I can see a PIN so that I can unlock my reserved Pedelec	Major

Product Backlog: Example Solution - Scenarios (1)

Scenario name	*Search available Pedelec*	
Participating actors	Initiated by *Marie, employee*	
Flow of events	1) *Marie opens the Pedelec App*	
		2) *The system shows a map with Marie's current location (Marienplatz) in the center of the screen and 3 Pedelecs near to Marie*
	3) *Marie browses the map by zooming and moving the map*	
		4) *The app zooms and moves the map. The map shows the Pedelecs that are located on the map*
Entry conditions	-	
Exit conditions	-	
Invariants	-	
Quality requirements	*Marie can zoom the map by pinching*	

Product Backlog: Example Solution - Scenarios (2)

Scenario name	*Check working radius*
Participating actors	Initiated by *Marie, employee*
Flow of events	1) *Marie selects one available Pedelec*
	2) *The system shows a dashed 20km radius around the Pedelec as center of the circle. It also shows a reserve button at the bottom of the page*
Entry conditions	*The app is already open and Pedelecs are visible on the map*
Exit conditions	*The Pedelec is selected and the range of the Pedelec is shown*
Invariants	*-*
Quality requirements	*Marie can select a Pedelec with one click*

Product Backlog: Example Solution - Scenarios (3)

Scenario name	*Reserve available Pedelec*
Participating actors	Initiated by *Marie, employee*
Flow of events	1) *Marie selects the reserve button at the bottom of the page*
	2) *The system shows a dialog for the start and end time of the reservation*
	3) *Marie chooses 14:30 as start and 16:30 as end time and selects the confirm button*
	4) *The app confirms the reservation for this time-slot*
Entry conditions	*The app is already open and one available Pedelec is selected*
Exit conditions	*The Pedelec is reserved for the chosen timeslot*
Invariants	*-*
Quality requirements	*Marie can reserve a Pedelec with four clicks*

Product Backlog: Example Solution - Scenarios (4)

Scenario name	*Return Pedelec*	
Participating actors	Initiated by *Marie, employee*	
Flow of events	1) *Marie parks the Pedelec in the parking slot and locks it. Then she opens the app.*	
		2) *The app automatically displays the Pedelec that is currently in use (based on Marie's reservation time-slot)*
	3) *Marie chooses the return button*	
		4) *The app confirms that Marie return the Pedelec and that it is now available for other reservations*
Entry conditions	*The Pedelec is reserved and in use*	
Exit conditions	*The Pedelec is free for reservations again*	
Invariants	*-*	
Quality requirements	*Marie can return the Pedelec with one click*	

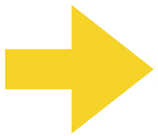
Product Backlog: Example Solution (Summary)

Prioritized Product Backlog

ID	Summary	Priority
1	☐ Search available Pedelecs	Critical
2	☐ Check working radius	Critical
3	☐ Reserve available Pedelec	Critical
4	☐ Return Pedelec	Major
5	☐ Contact colleague	Minor
6	☐ Pass reservation	Minor
7	☐ Report damage	Minor
8	☐ Unlock Pedelec	Major

Outline

- Agile Methods: Scrum
  - Artifacts
  - Meetings
  - Roles
  - **Exercise 1:** Creation of the Product Backlog: User Stories and Scenarios



JIRA Agile

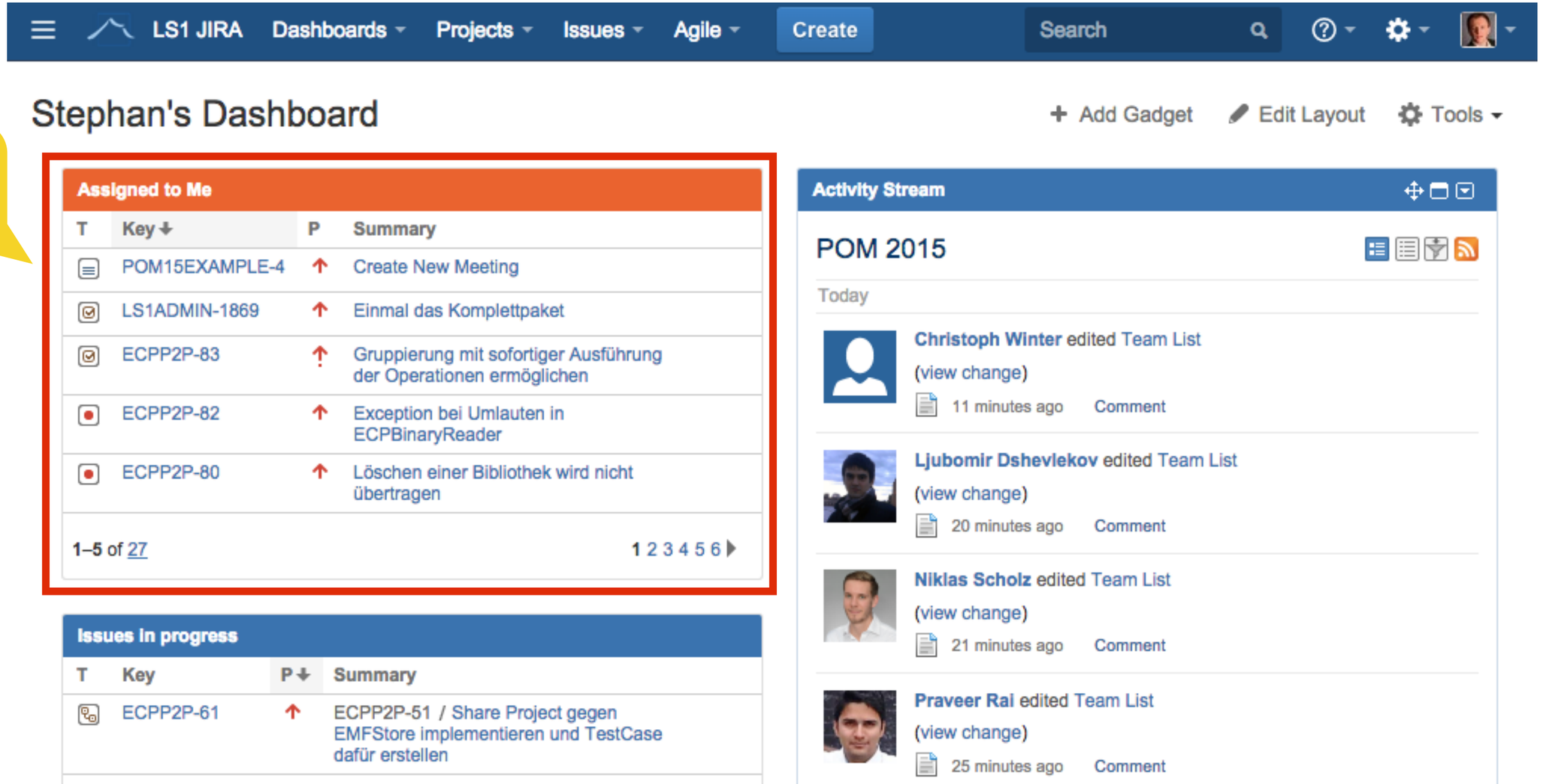
  - **Exercise 2:** Creation of the Product Backlog in JIRA
  - **Exercise 3:** Sprint Planning Part I - Creation of the Sprint Backlog in JIRA
  - **Exercise 4:** Sprint Planning Part II - Refinement of the Sprint Backlog in JIRA
  - **Exercise 5:** Working on the Taskboard during the Sprint
  - **Exercise 6:** Sprint Review

JIRA Agile

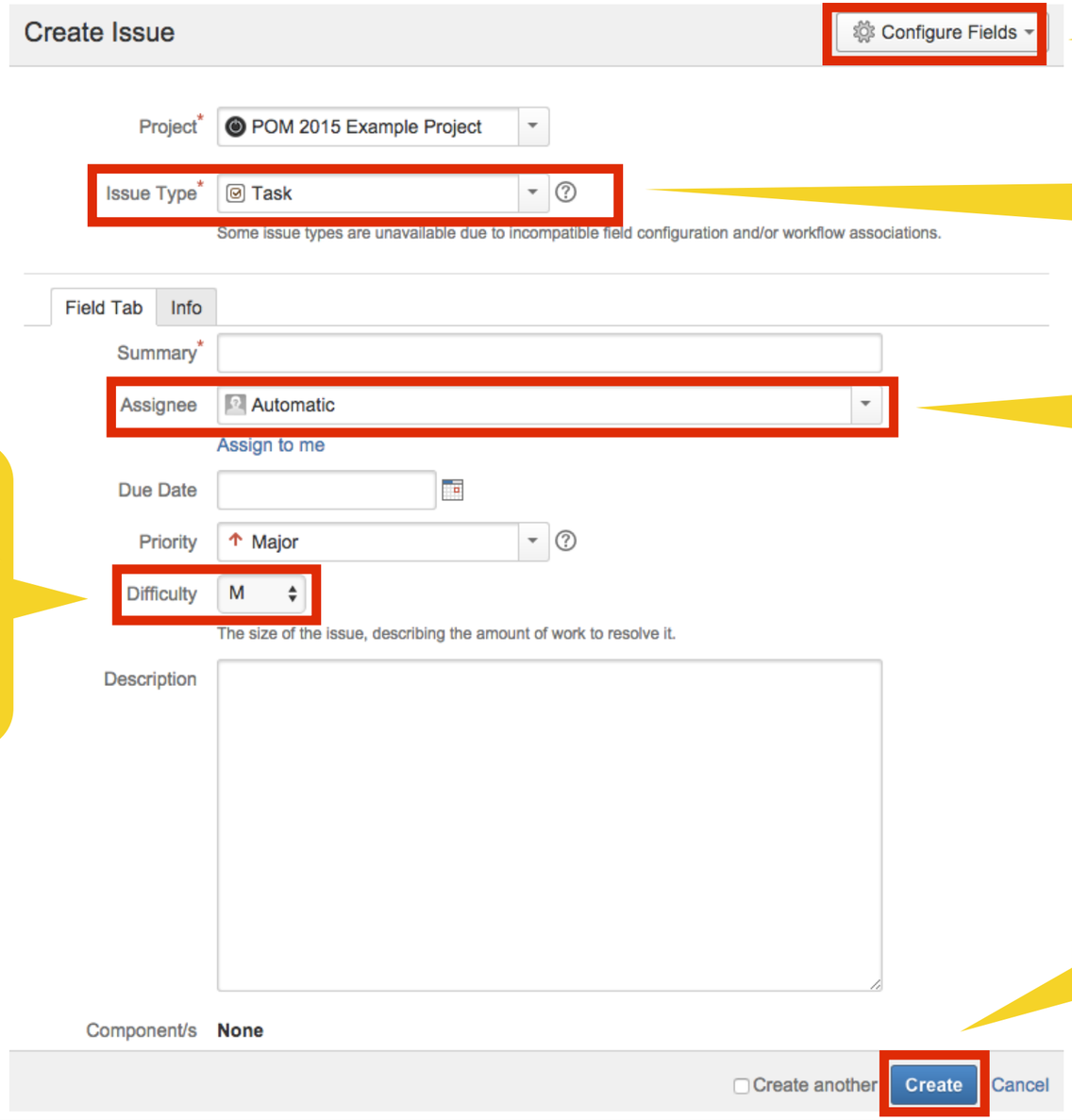
- Create and Manage the Product Backlog
  - Create Backlog Items (e.g. Scenario, User Story)
  - Estimate the difficulty
  - Define sub-tasks for the unit of work during development
  - Prioritization of backlog items and tasks
- Create and Manage Sprint Backlogs
  - Plan Sprints
  - Track progress in active Sprints
  - View reports about finished Sprints

JIRA Dashboard

All issues currently assigned to you.



Create Issue Screen



Configuration of the fields visible on your screen

Issue Type (e.g. Task, Bug, User Story, Scenario)

The Person responsible for the resolution of the issue

The difficulty of the issue describes the amount of work estimated to resolve it.
It is measured with t-shirt sizes: `S, M, L, XL`

'`Create`' button

Level of Difficulty

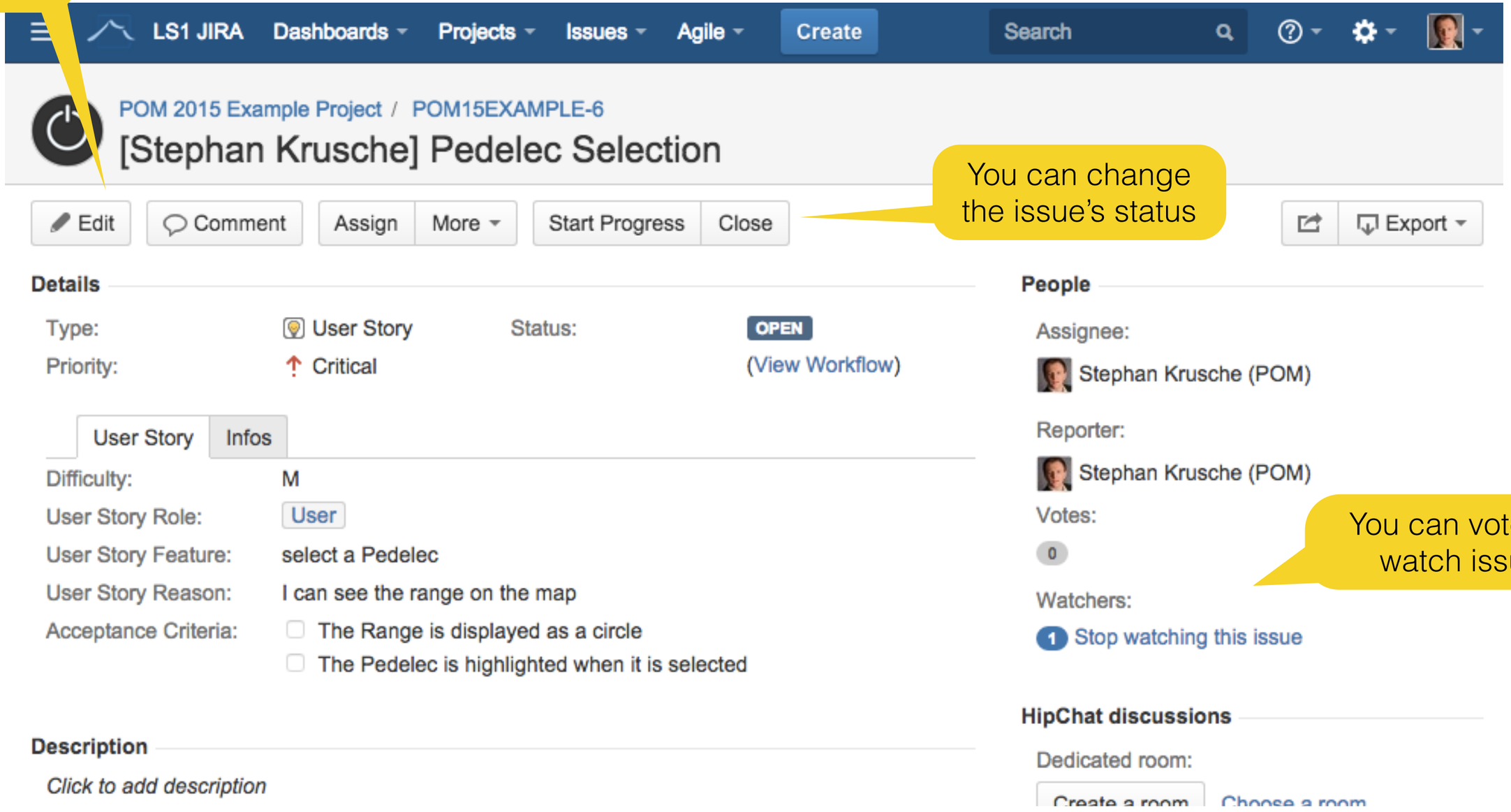
- In our projects, T-Shirt sizes represent the rough difficulty of the realization of the issue, i.e. how much effort / time the Development Team needs to resolve the issue
  - Small (**S**)
  - Medium (**M**)
  - Large (**L**)
  - Extra Large (**XL**) —> too big, split it into smaller issues!
- In the Sprint Planning Meeting the team decides on the difficulty of the backlog items by estimating Story Points (1/2, 1, 2, 3, 5, 8, 13, 21, 100) using e.g.:
  - Poker Planning
  - Team Estimation Game

➡ Both are agile estimation techniques: we introduce more techniques in the ***Estimation and Scheduling*** lecture and exercise

Lecture on June 03
Exercise on June 05

Issue Browser

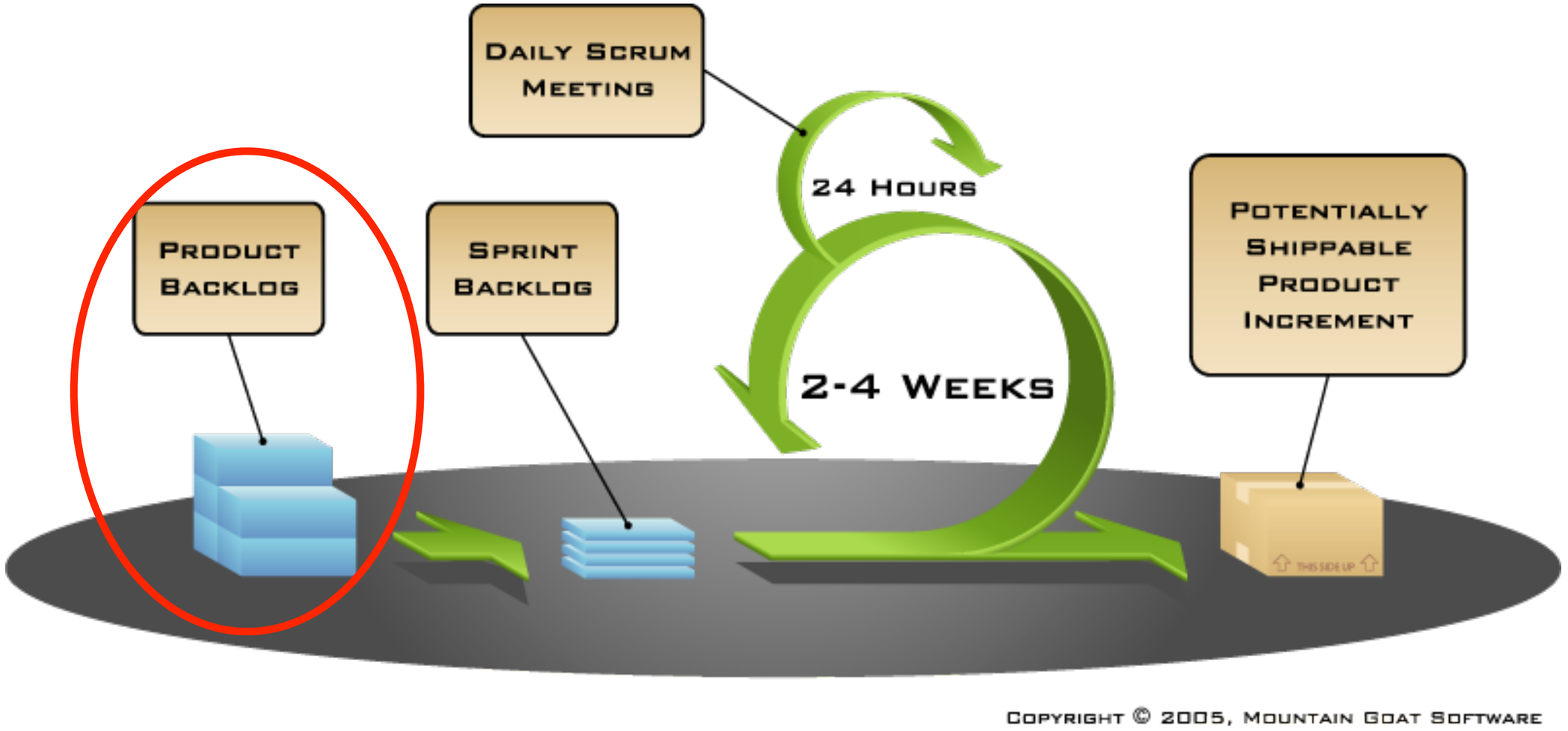
You can edit the issue



You can change the issue's status

You can vote and watch issues

Exercise 2: Creation of the Product Backlog in JIRA



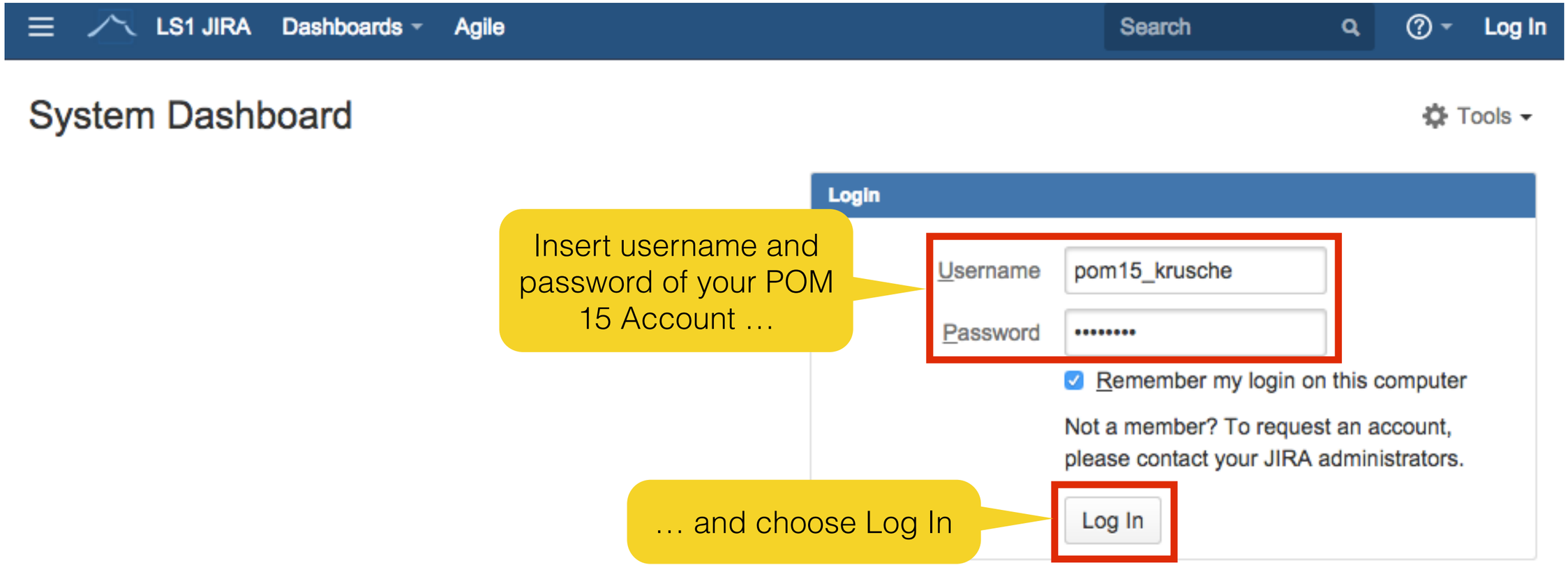
Time:
8 min

Exercise 2: Creation of the Product Backlog in JIRA

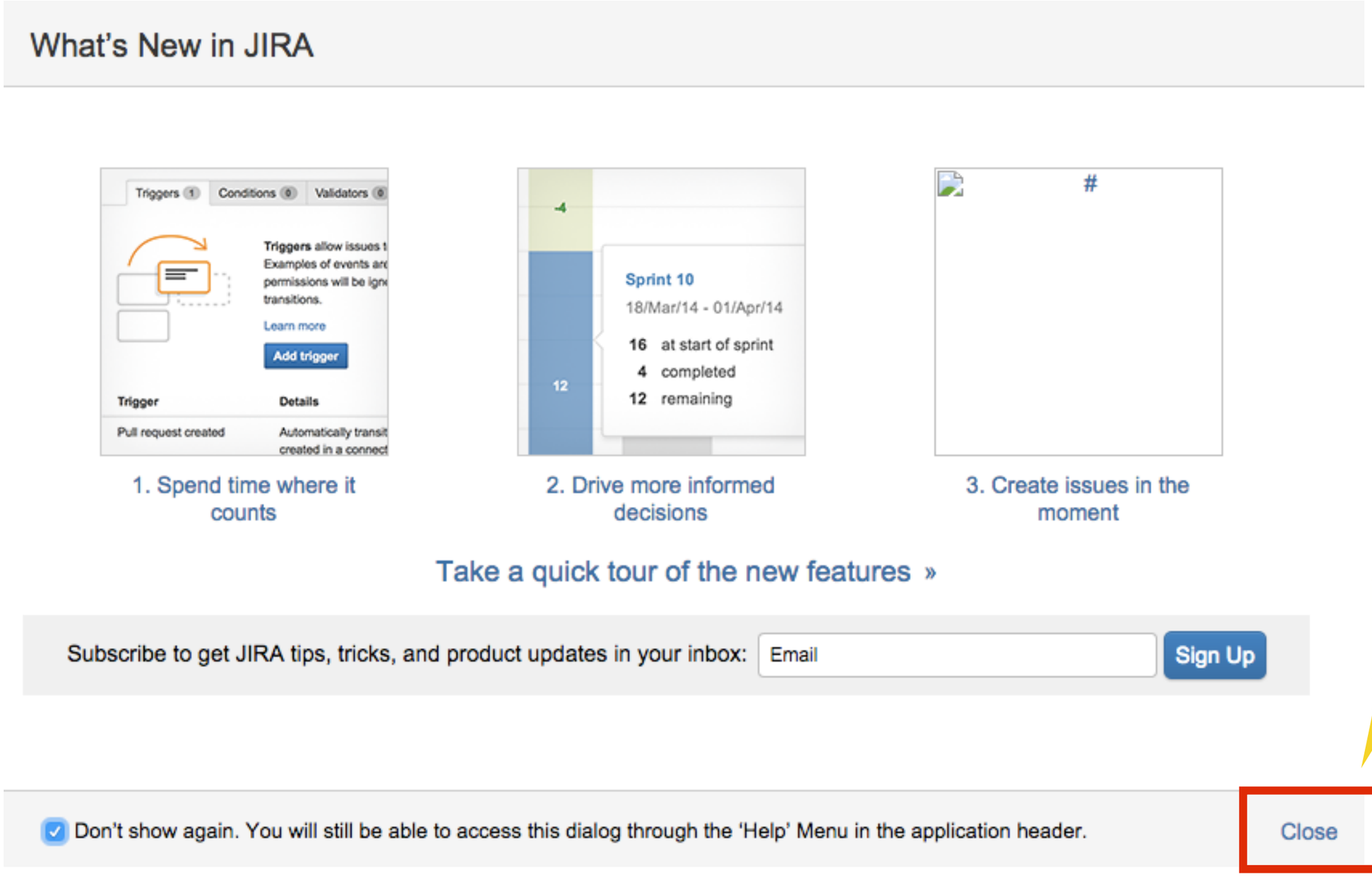
Task 5: Login to JIRA and open the POM 2015 Example Project
Task 6: Create 3 User Stories of **Exercise 1** in JIRA and **assign it to yourself**
Task 7: Create 1 Scenario of **Exercise 1** in JIRA and **assign it to yourself**
Hint: Use the Prefix [Your Full Name] to easily find the issue later (just for exercise purposes)
Task 8: Open and inspect the Product Backlog in JIRA Agile

Task 4: Login into JIRA and open the POM 2015 Example Project

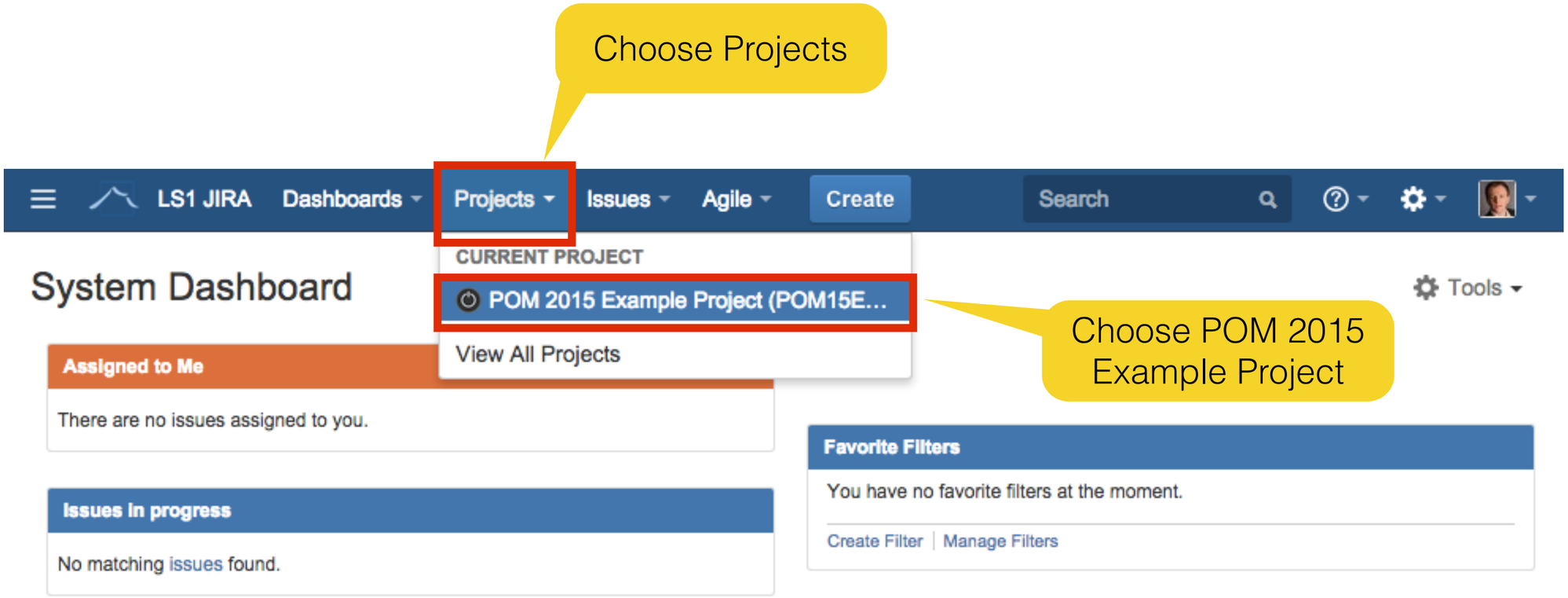
https://jirabruegge.in.tum.de



Task 4: Login into JIRA and open the POM 2015 Example Project

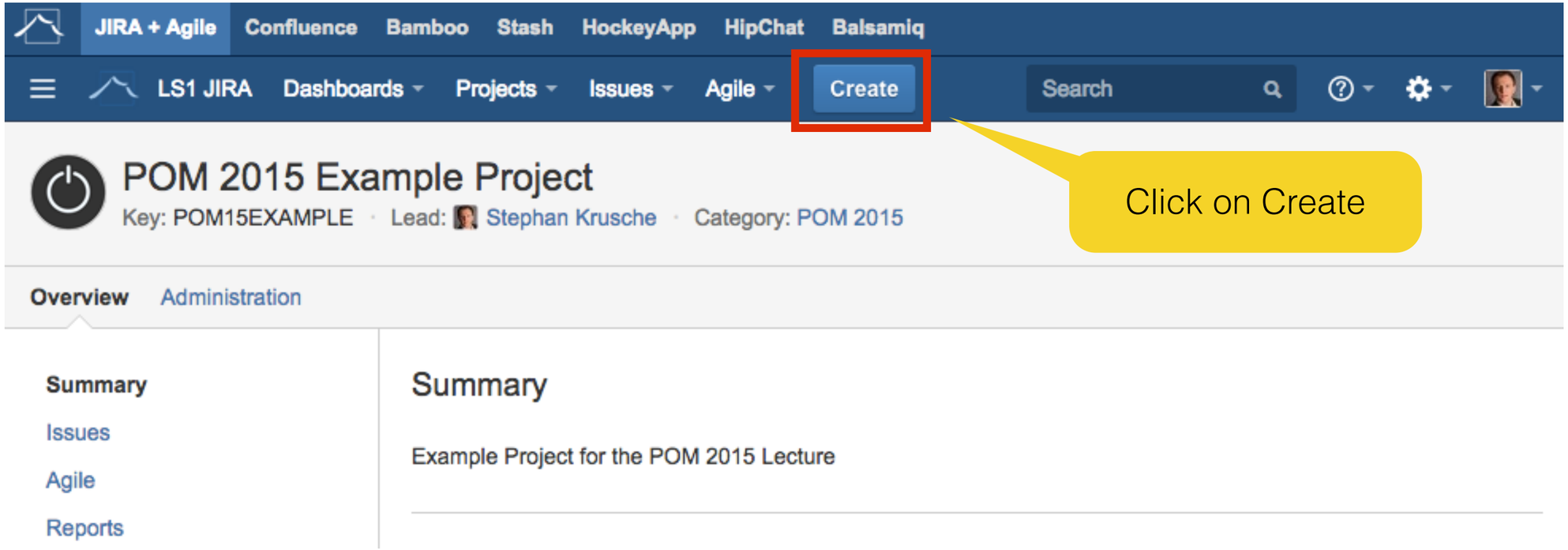


Task 4: Login into JIRA and open the POM 2015 Example Project

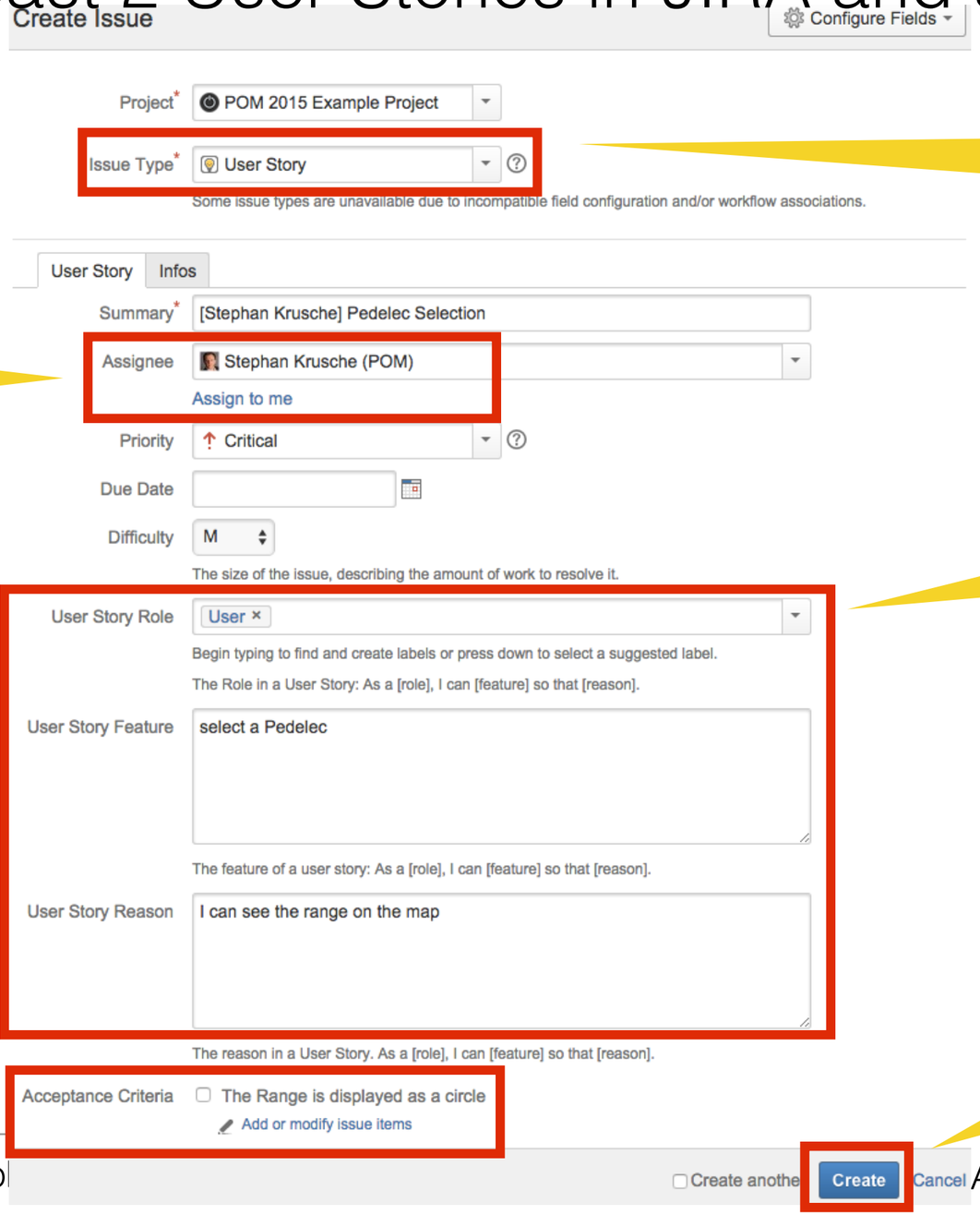


Task 5: Create at least 2 User Stories in JIRA and assign it to yourself

https://jirabruegge.in.tum.de/browse/POM15EXAMPLE



Task 5: Create at least 2 User Stories in JIRA and assign it to yourself



Choose User Story as issue type

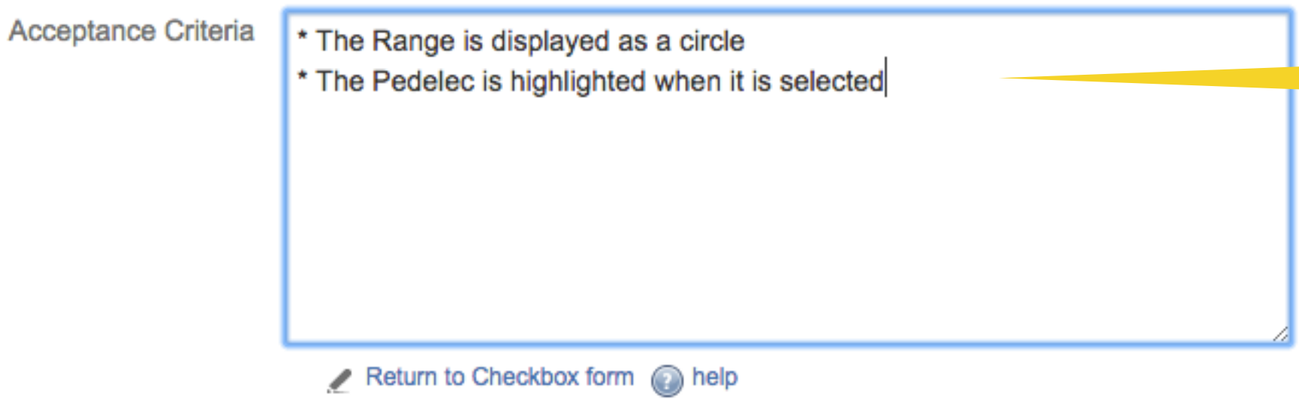
Assign to yourself

Fill out all fields to your needs (remember INVEST).

Create appropriate Acceptance Criteria

Finally, click on the 'Create' button.

Task 5: Create at least 2 User Stories in JIRA and assign it to yourself



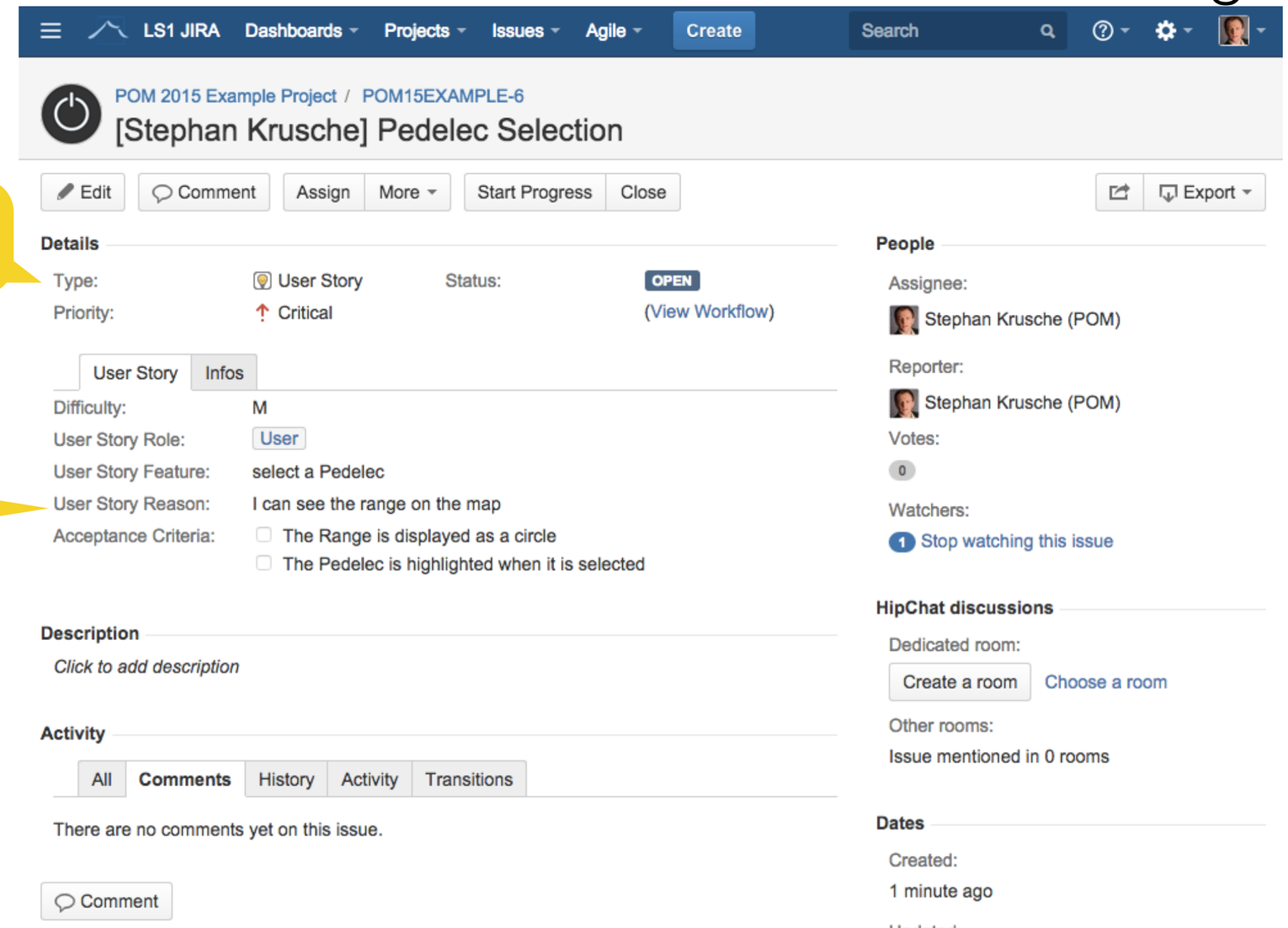
Add Acceptance Criteria as Bullet Points so that you can tick them off later

Task 5: Create at least 2 User Stories in JIRA and assign it to yourself

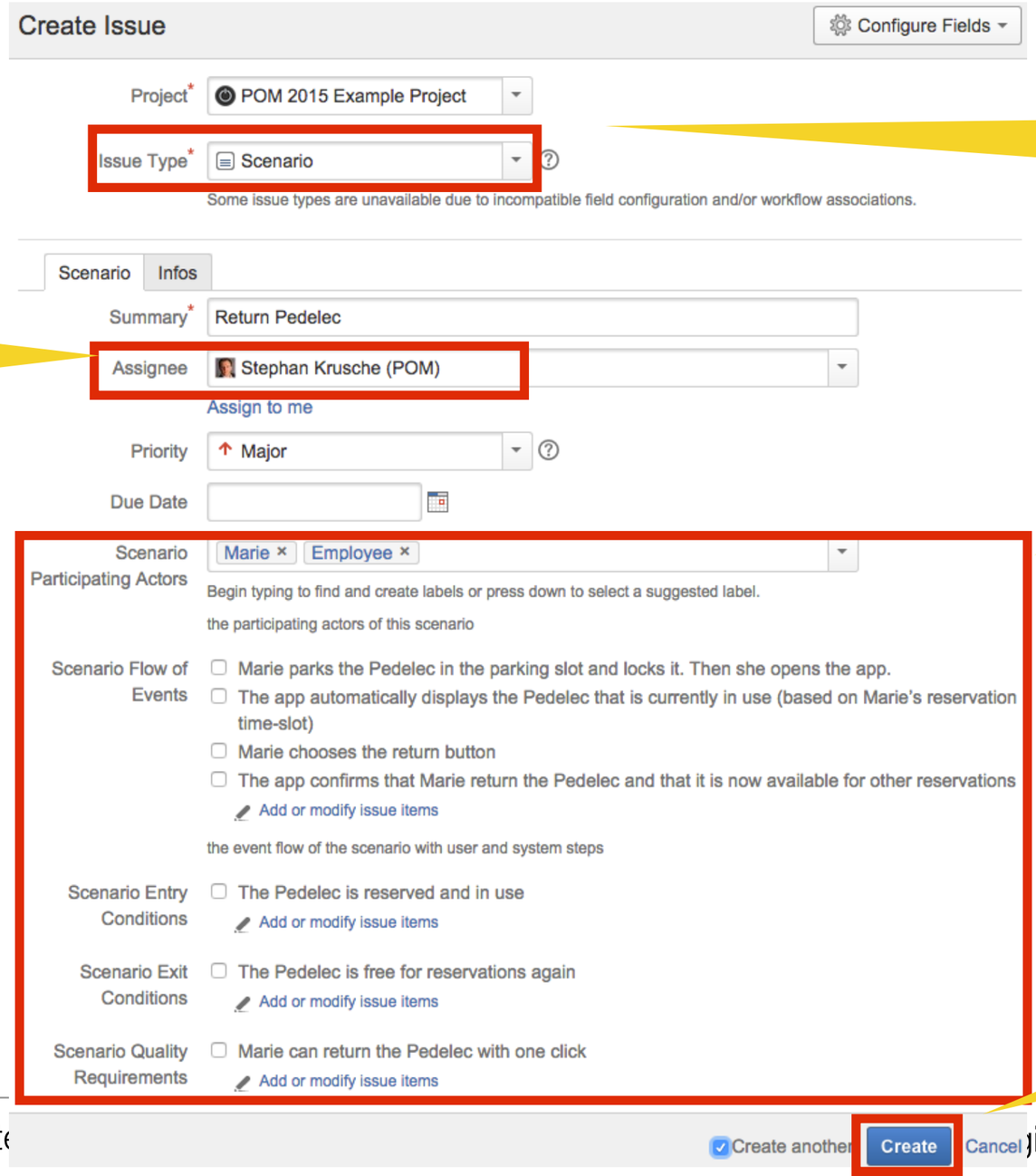
Task 5: Create at least 2 User Stories in JIRA and assign it to yourself

Issue Type: User Story

User Story Template and Acceptance Criteria



Task 7: Create at least 1 Scenario in JIRA and assign it to yourself



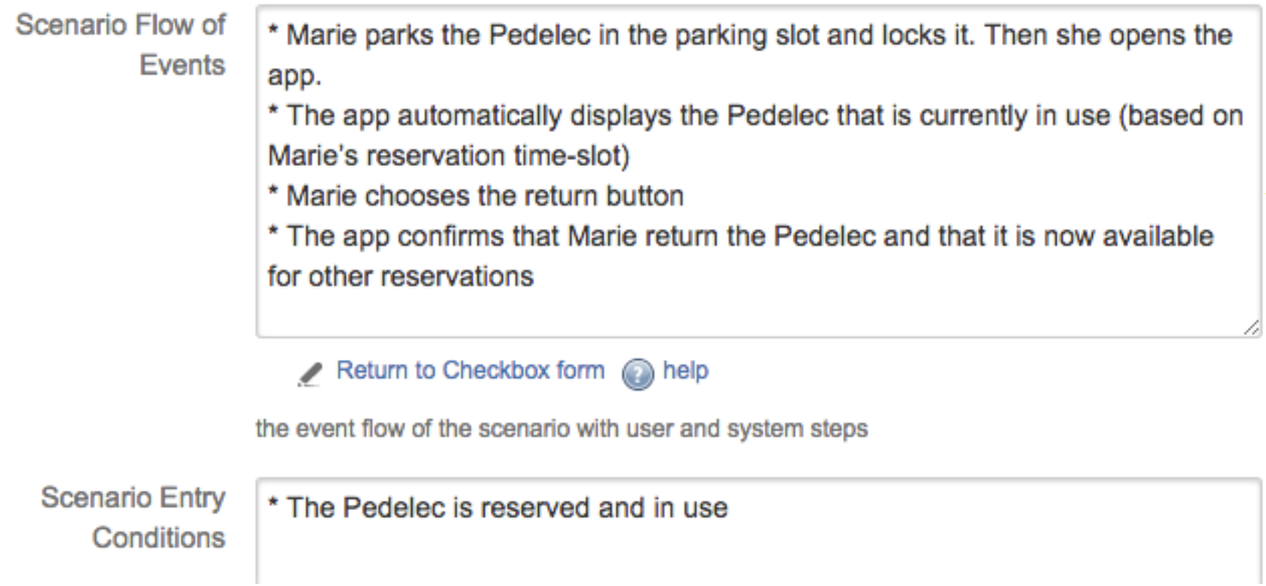
Choose Scenario as issue type

Assign to yourself

Fill out all fields of a scenario

Finally, click on the 'Create' button. Choose 'Create another' to save time when adding multiple issues

Task 7: Create at least 1 Scenario in JIRA and assign it to yourself

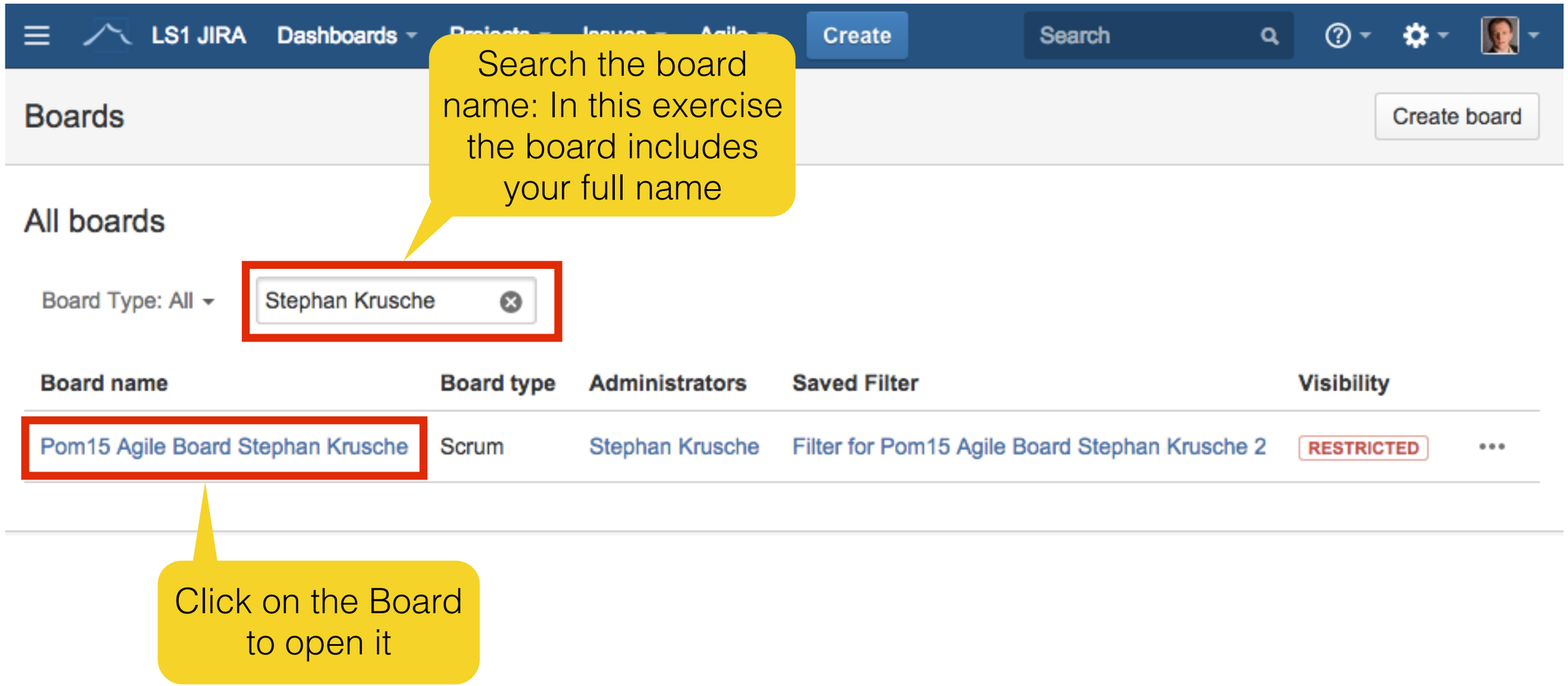


Use the Bullet Point notation (with stars) for Flow of Events, Entry Conditions, Exit Conditions and Quality Requirements

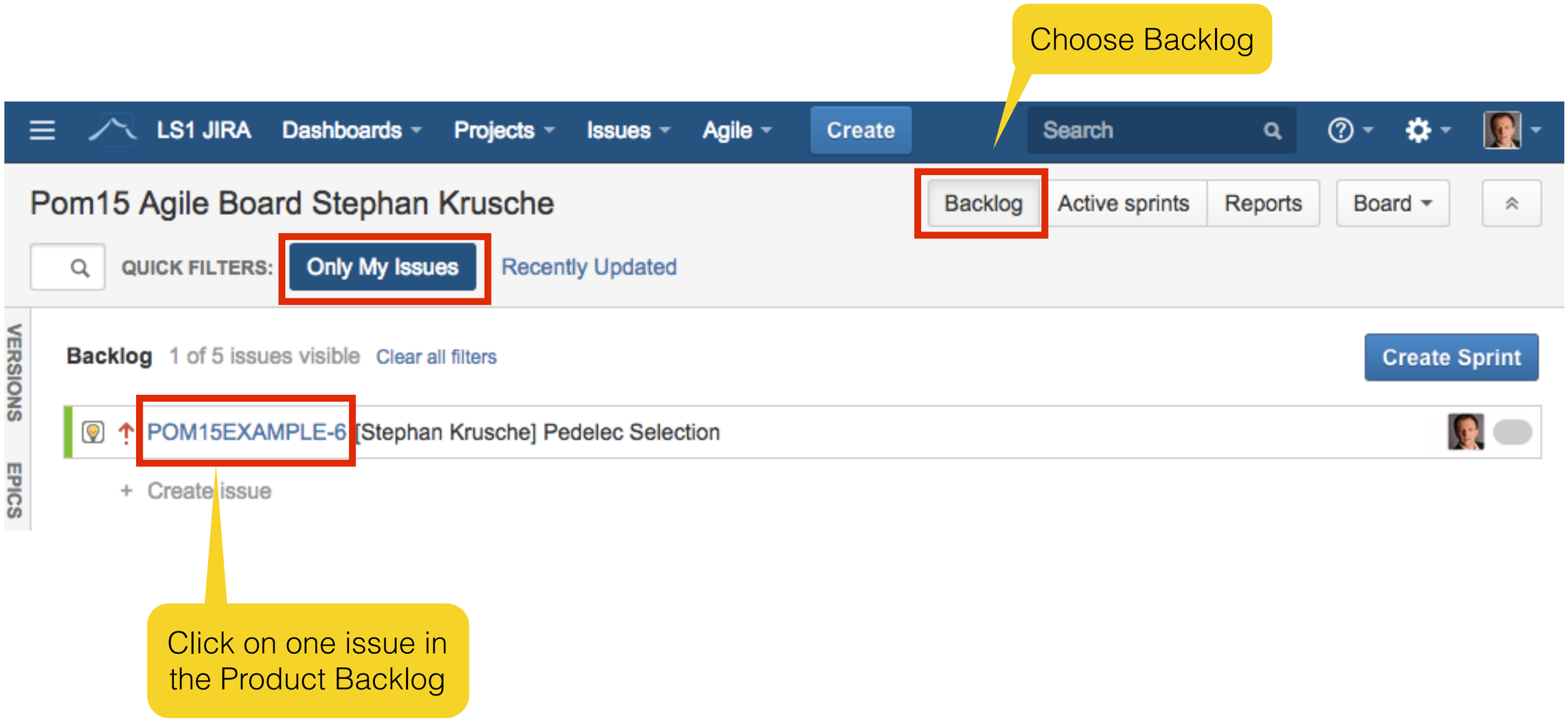
Note: JIRA does not support Scenarios natively and the current screen is a workaround to support it :-)

Task 8: Open and inspect the Product Backlog in JIRA Agile

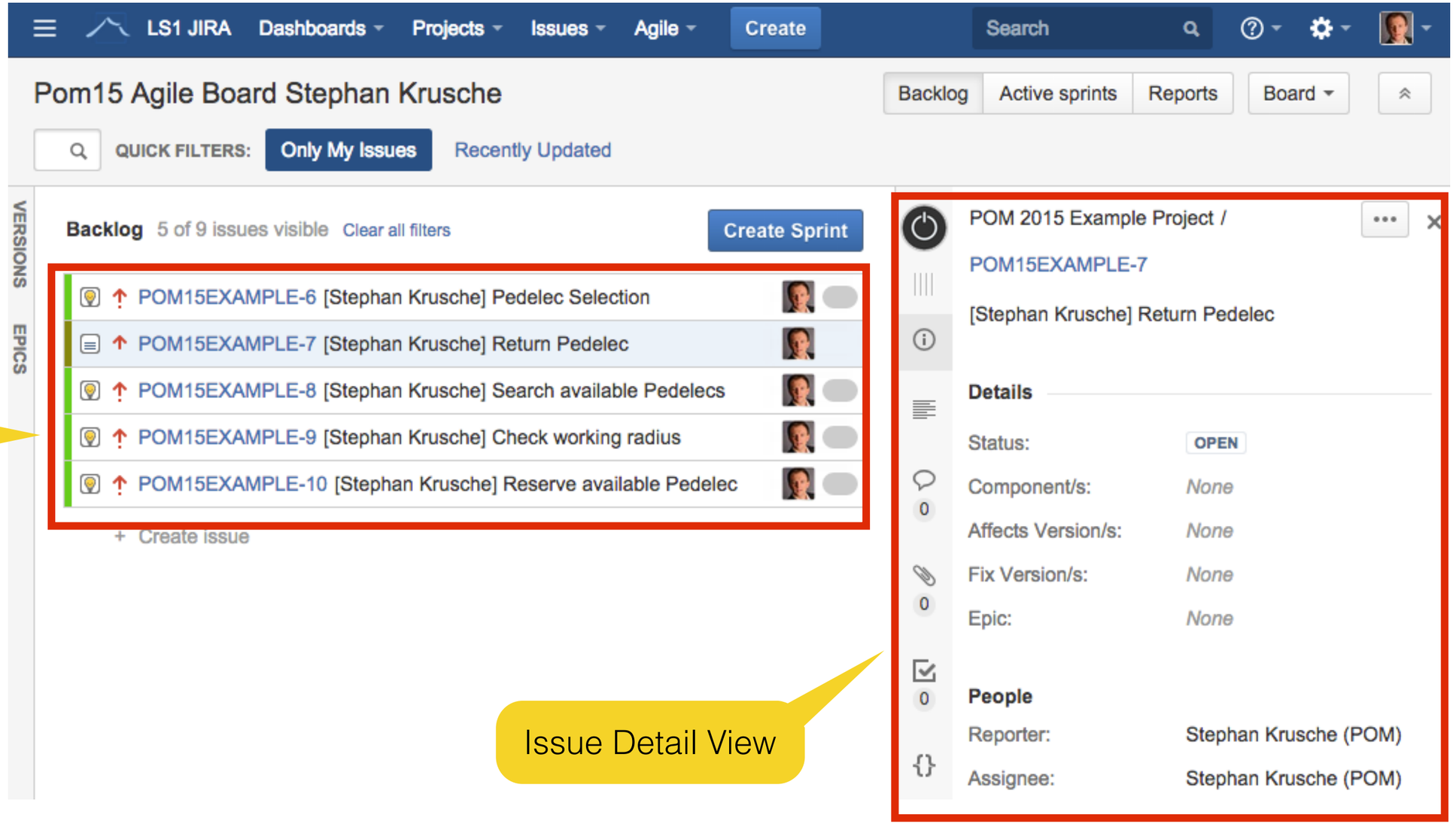
Task 8: Open and inspect the Product Backlog in JIRA Agile



Task 8: Open and inspect the Product Backlog in JIRA Agile



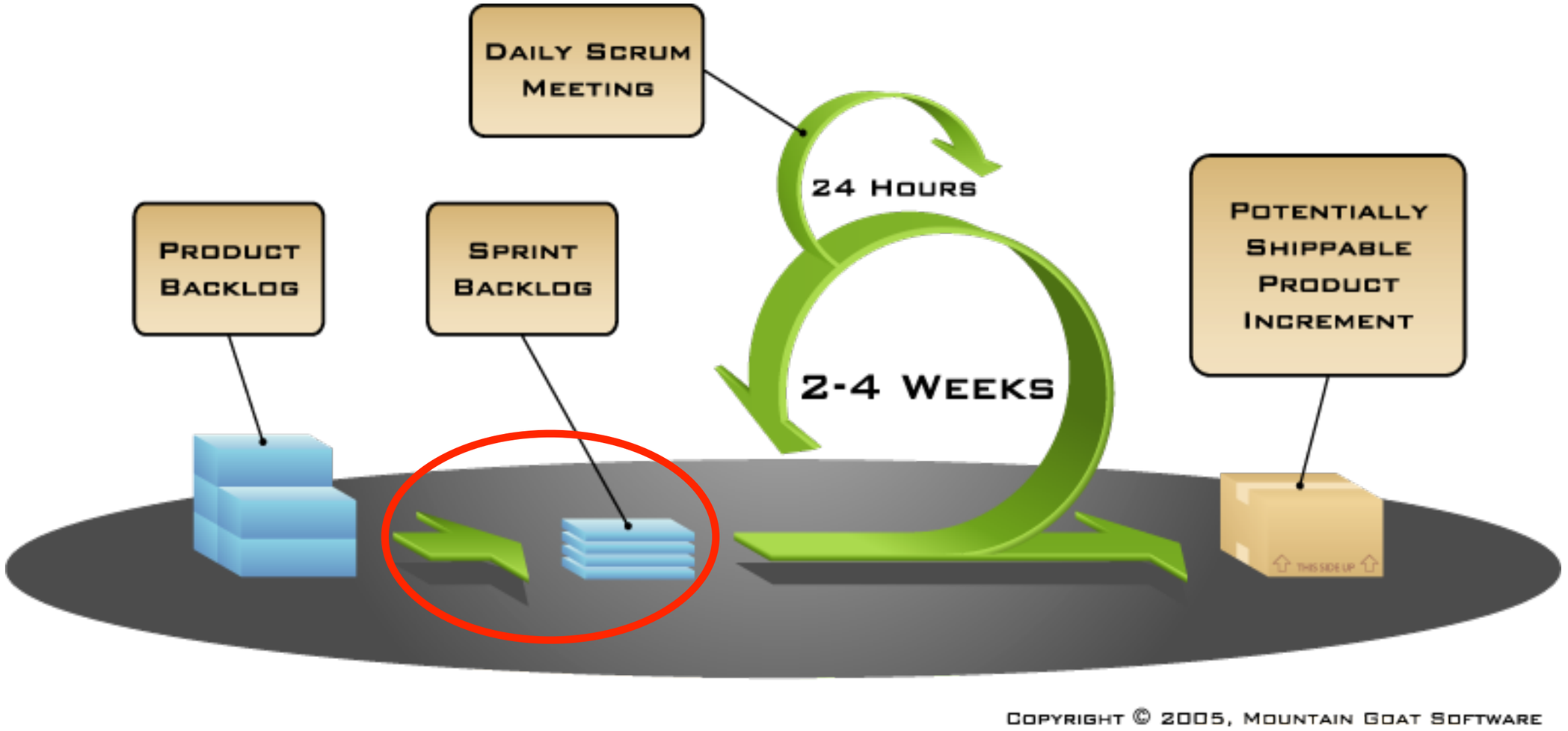
Task 8: Open and inspect the Product Backlog in JIRA Agile



Product Backlog

Issue Detail View

Exercise 3: Sprint Planning Part I - Creation of the Sprint Backlog in JIRA



Sprint Planning Meeting

- The Development Team estimates the difficulty for the most important 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

Prioritized and Estimated Product Backlog

ID	Name	Priority	Difficulty
1	Search available Pedelecs	Critical	Medium
2	Check working radius	Critical	Large
3	Reserve available Pedelec	Critical	Small
4	Return Pedelec	Major	Small
5	Contact colleague	Minor	Medium
6	Pass reservation	Minor	Medium
7	Report damage	Minor	Large
8	Unlock Pedelec	Major	Small

Sprint 1 Backlog

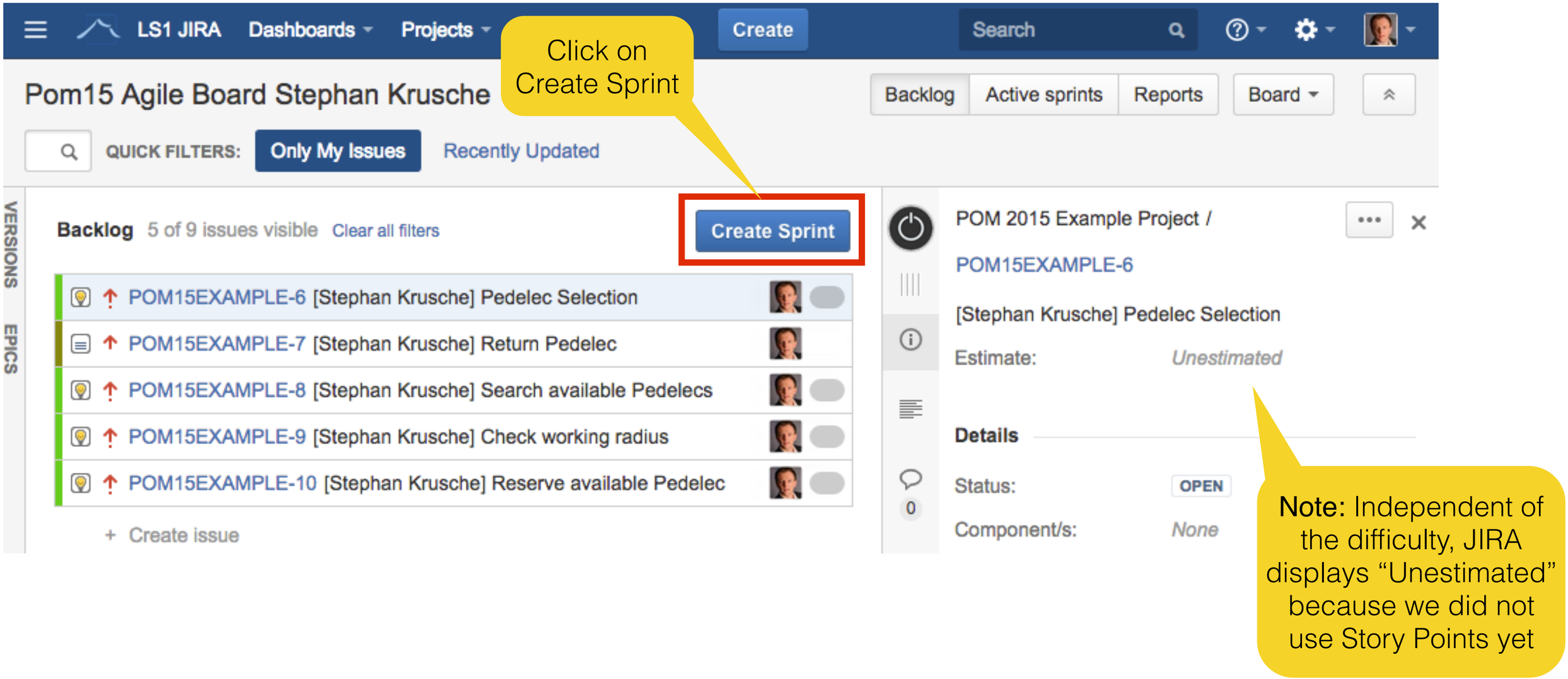
ID	Name	Difficulty
1	Search available Pedelecs	Medium
2	Check working radius	Large

Time:
3 min

Exercise 3: Sprint Planning Part I - Creation of the Sprint Backlog in JIRA

Task 9: Create a new Sprint
Task 10: Move 2 Backlog Items to the new Sprint Backlog
Task 11: Start the Sprint

Task 9: Create a new Sprint

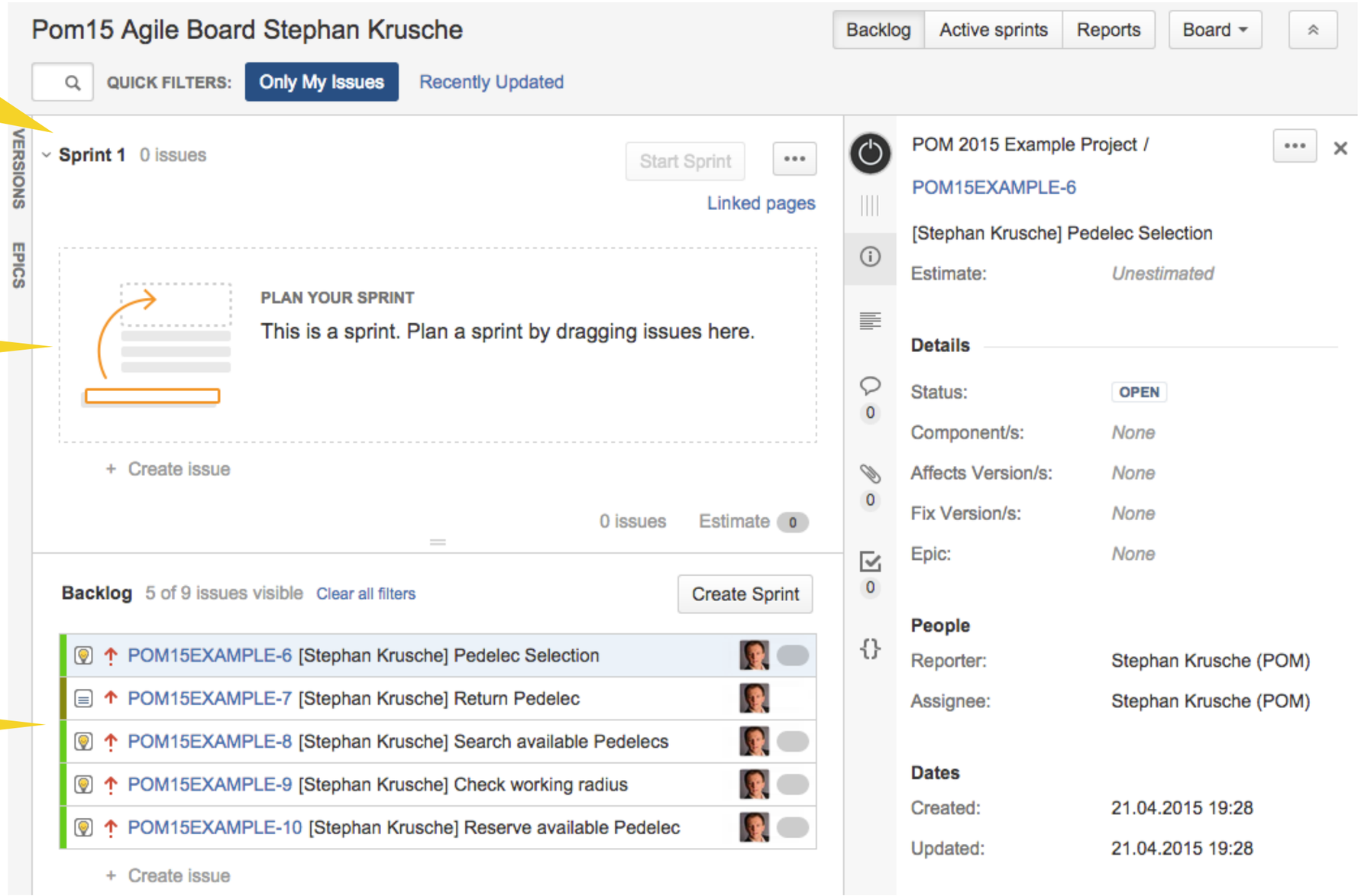


Task 9: Create a new Sprint

Note: You might need to rename the Sprint

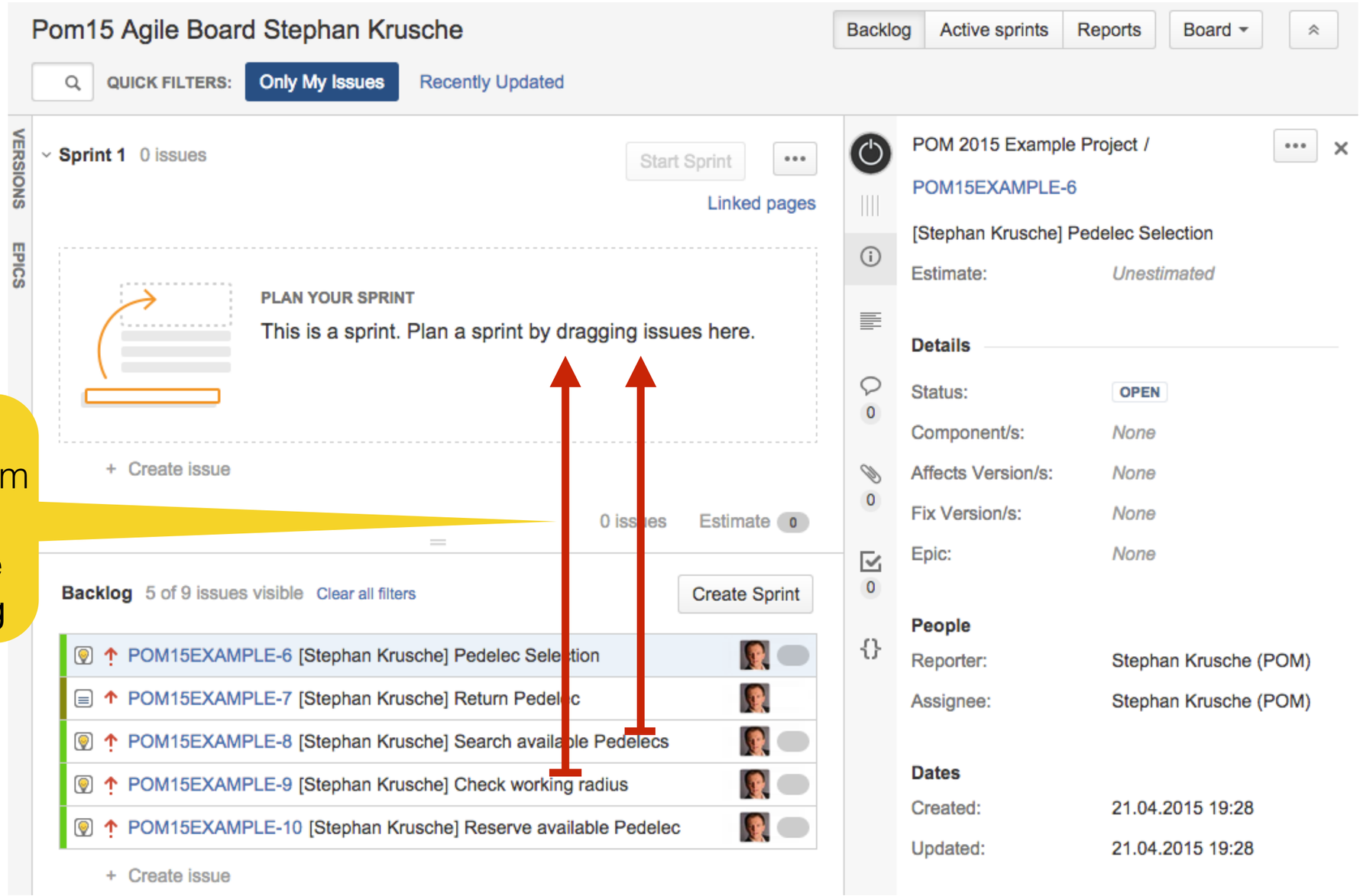
Sprint Backlog (empty)

Product Backlog

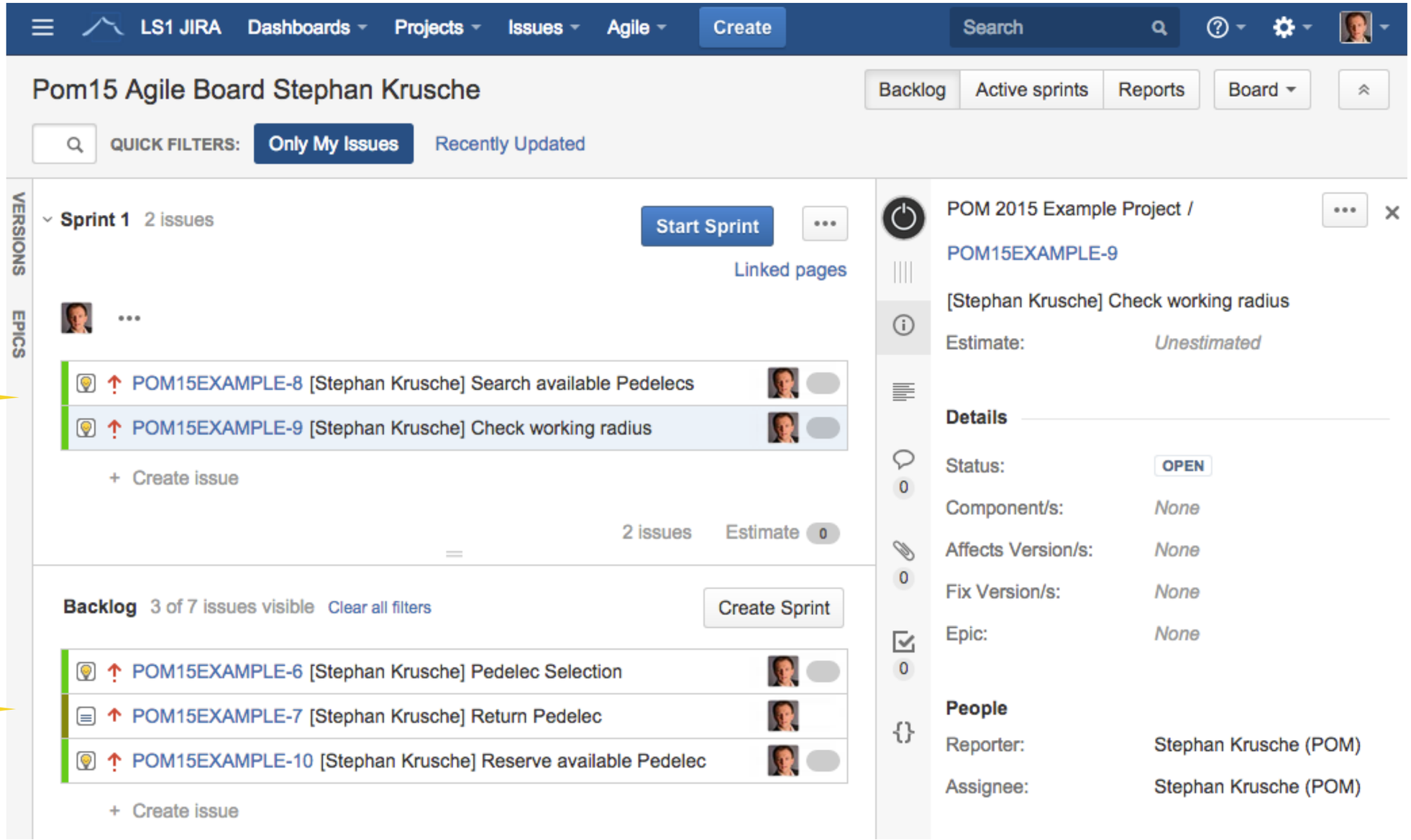


Task 10: Move 2 Backlog Items to the new Sprint Backlog

Move the backlog items from the **Product Backlog** into the **Sprint 1 Backlog**



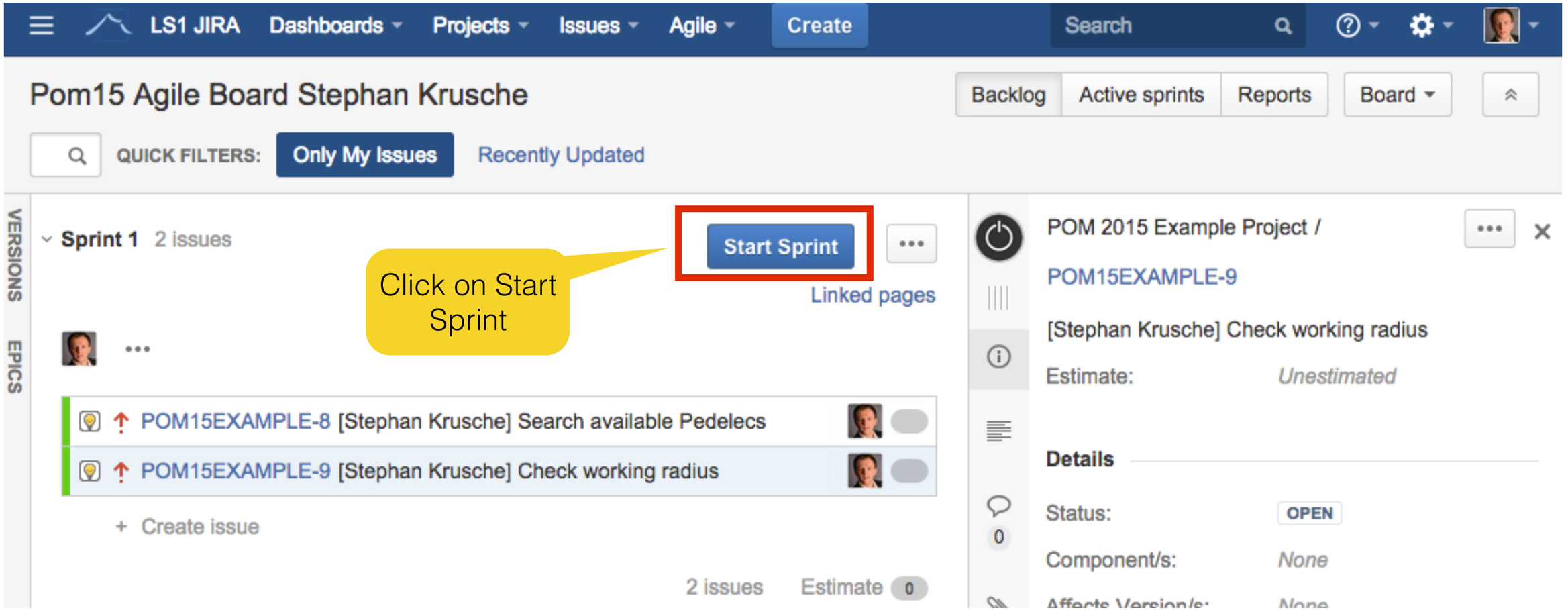
Task 10: Move 2 Backlog Items to the new Sprint Backlog



Sprint 1 Backlog

Product Backlog

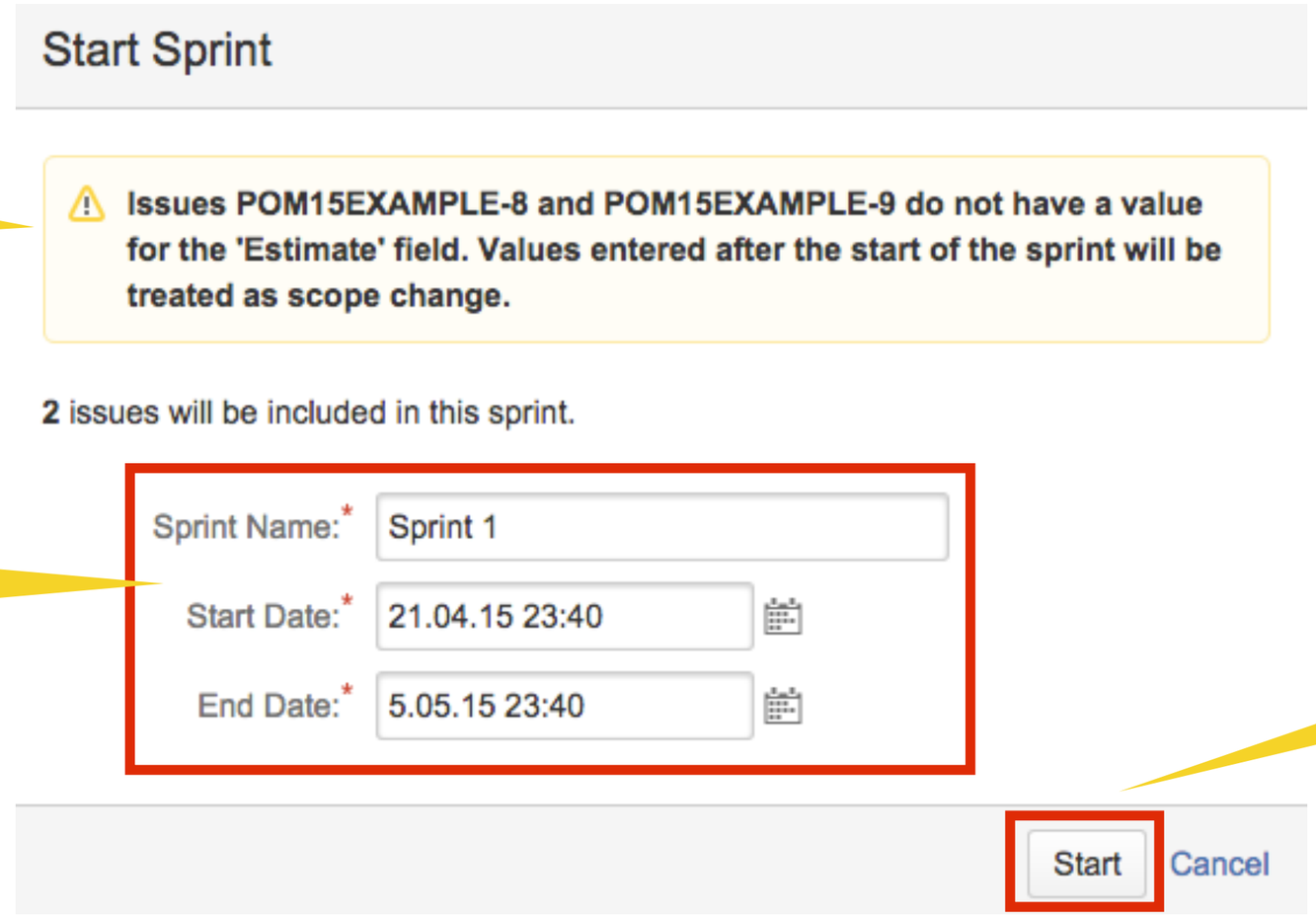
Task 11: Start the Sprint



Task 11: Start the Sprint

Ignore the warning because we do not yet use Story Points

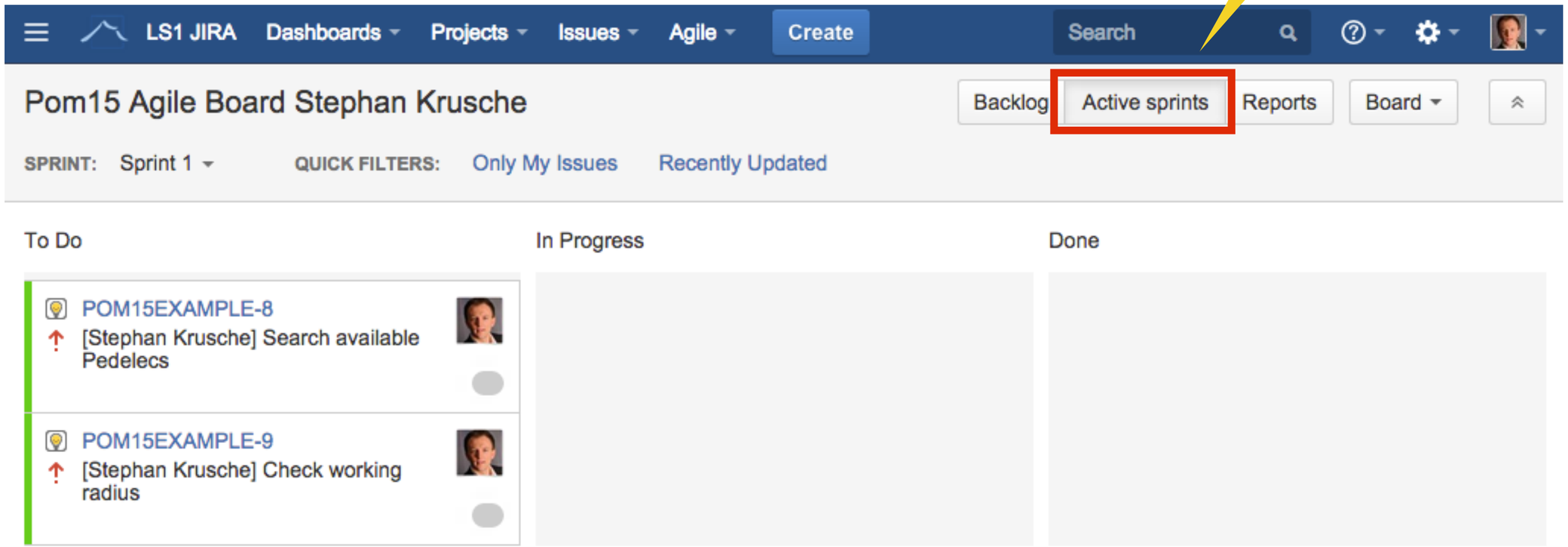
Make sure to use Sprint 1 as name. Leave the default duration of 2 weeks



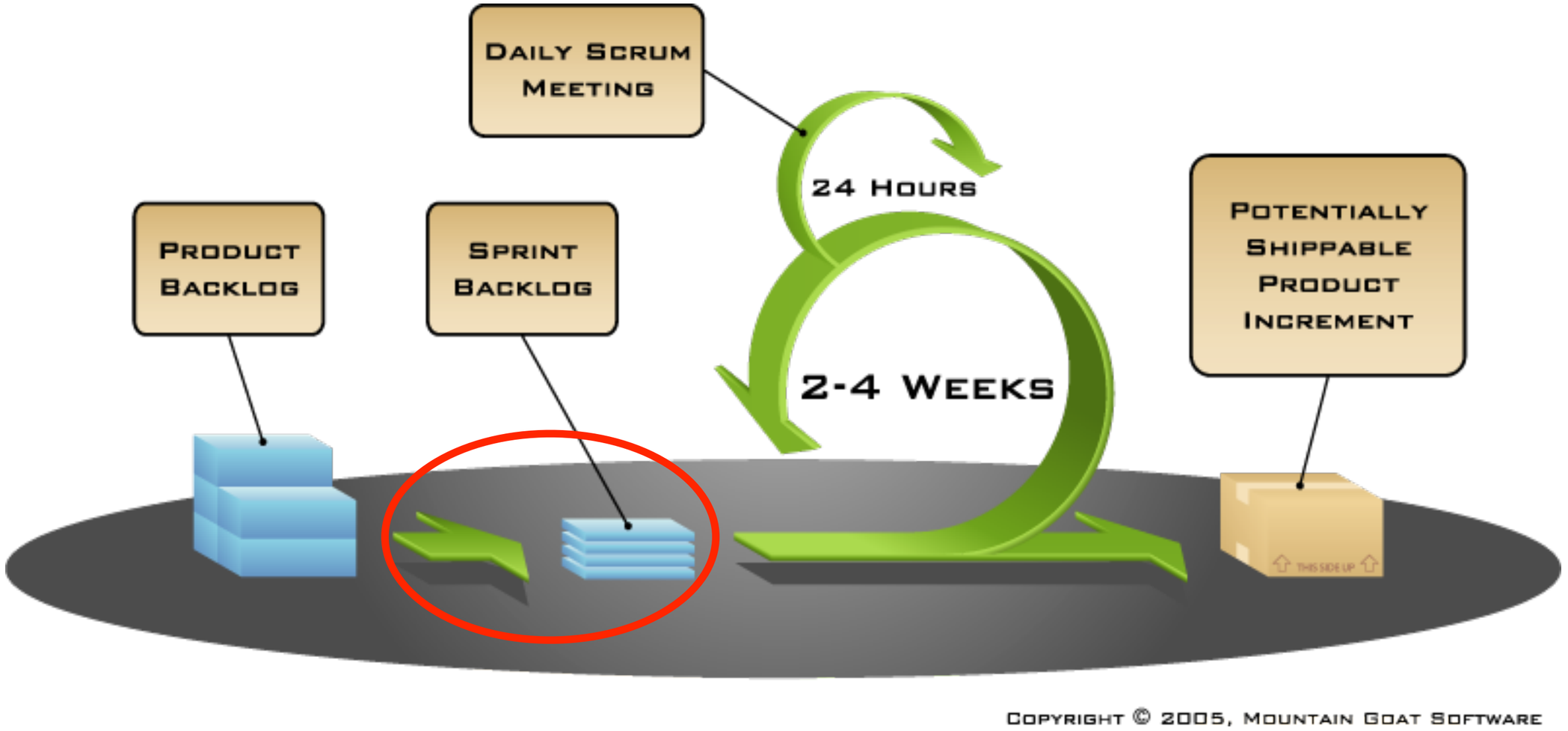
Click Start

Task 11: Start the Sprint

JIRA Agile automatically activates the page Active sprints

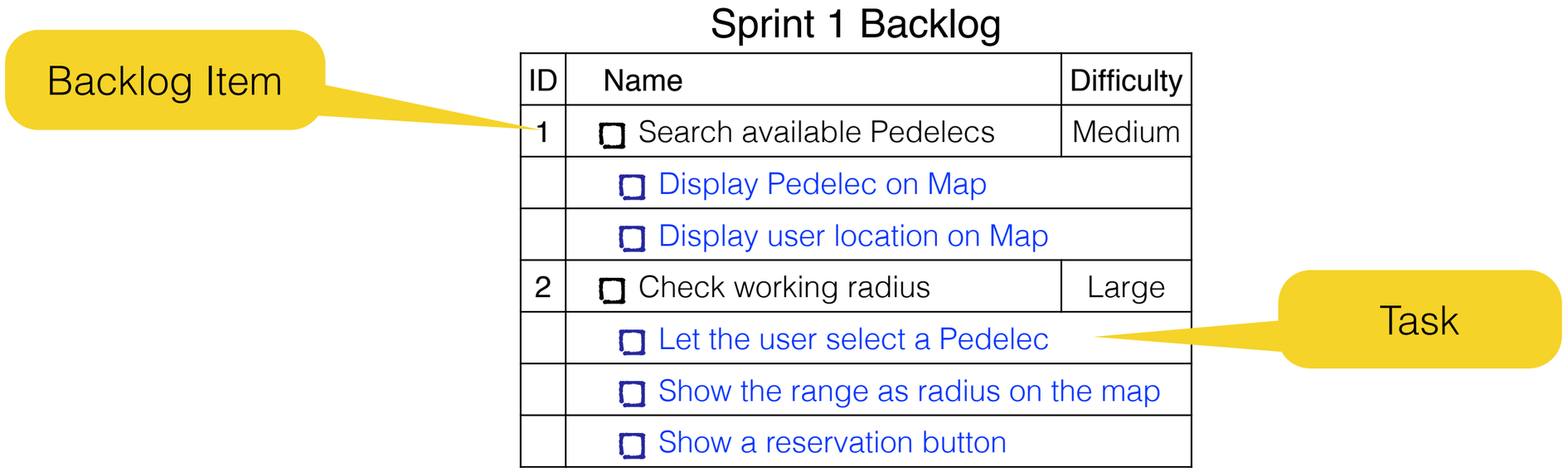


Exercise 4: Sprint Planning Part II - Refinement of the Sprint Backlog in JIRA



Refinement of the Sprint Backlog

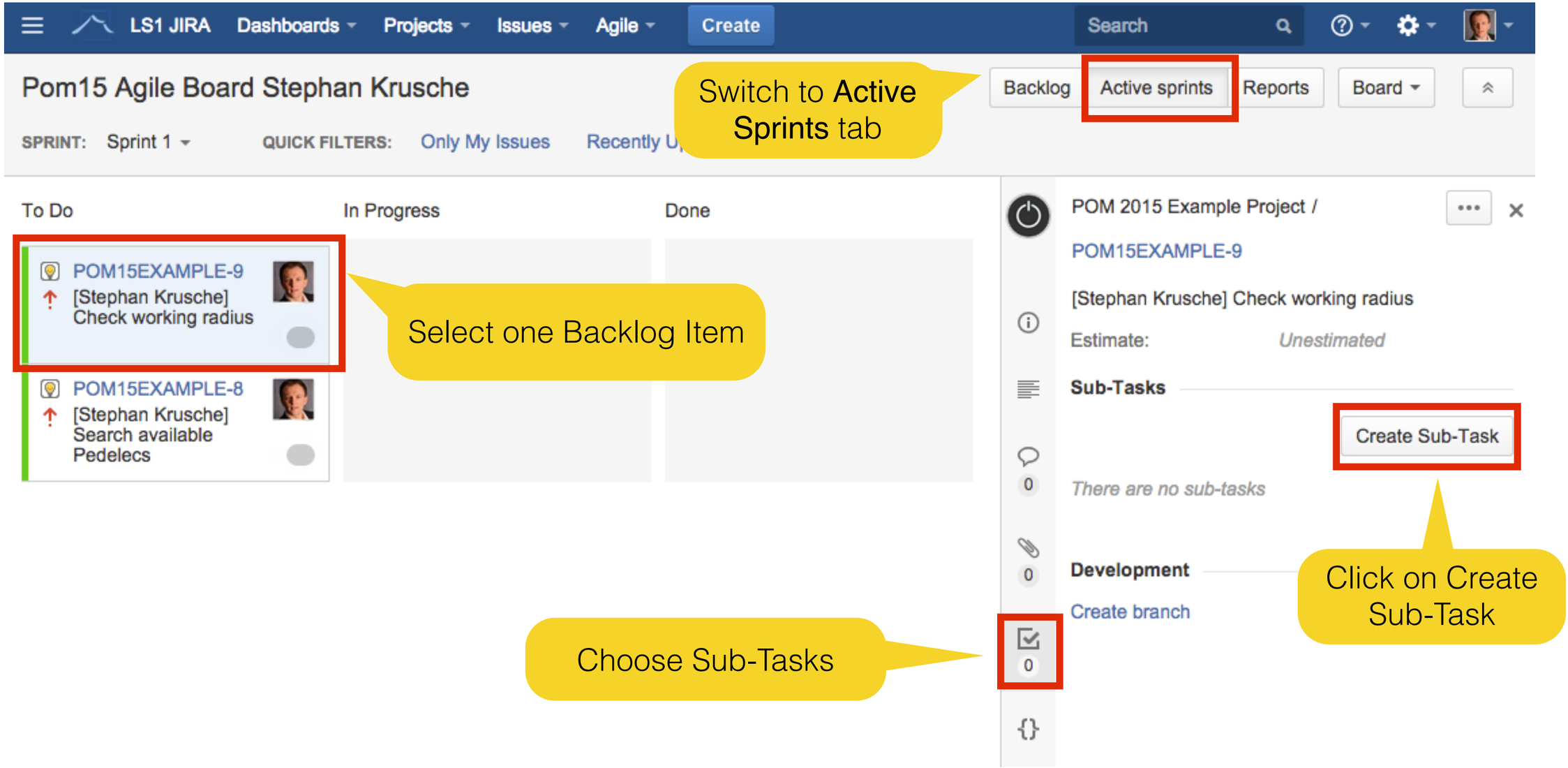
- The Development Team needs to discuss the concrete implementation to work on a backlog item
- The Development Team creates tasks to realize the backlog items (e.g. to match the acceptance criteria defined by the product owner)



Exercise 4: Sprint Planning Part II - Refinement of the Sprint Backlog in JIRA

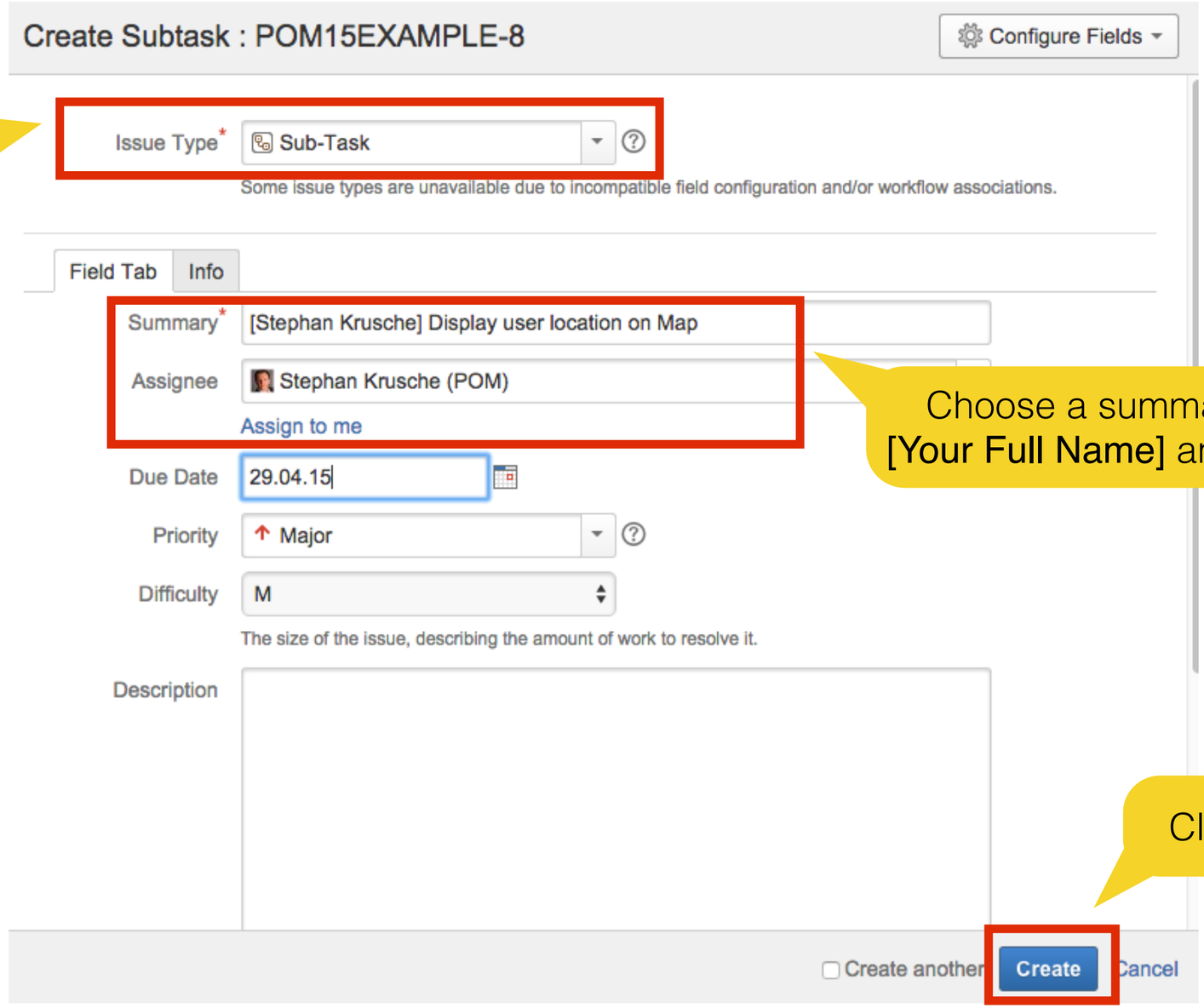
Task 12: Create at least 2 Sub-Tasks for the 2 Sprint Backlog Items and assign it to yourself

Task 12: Create Sub-Tasks for Sprint 1 Backlog Items and assign it to yourself



Task 12: Create Sub-Tasks for Sprint 1 Backlog Items and assign it to yourself

Issue Type Sub-Task is preselected



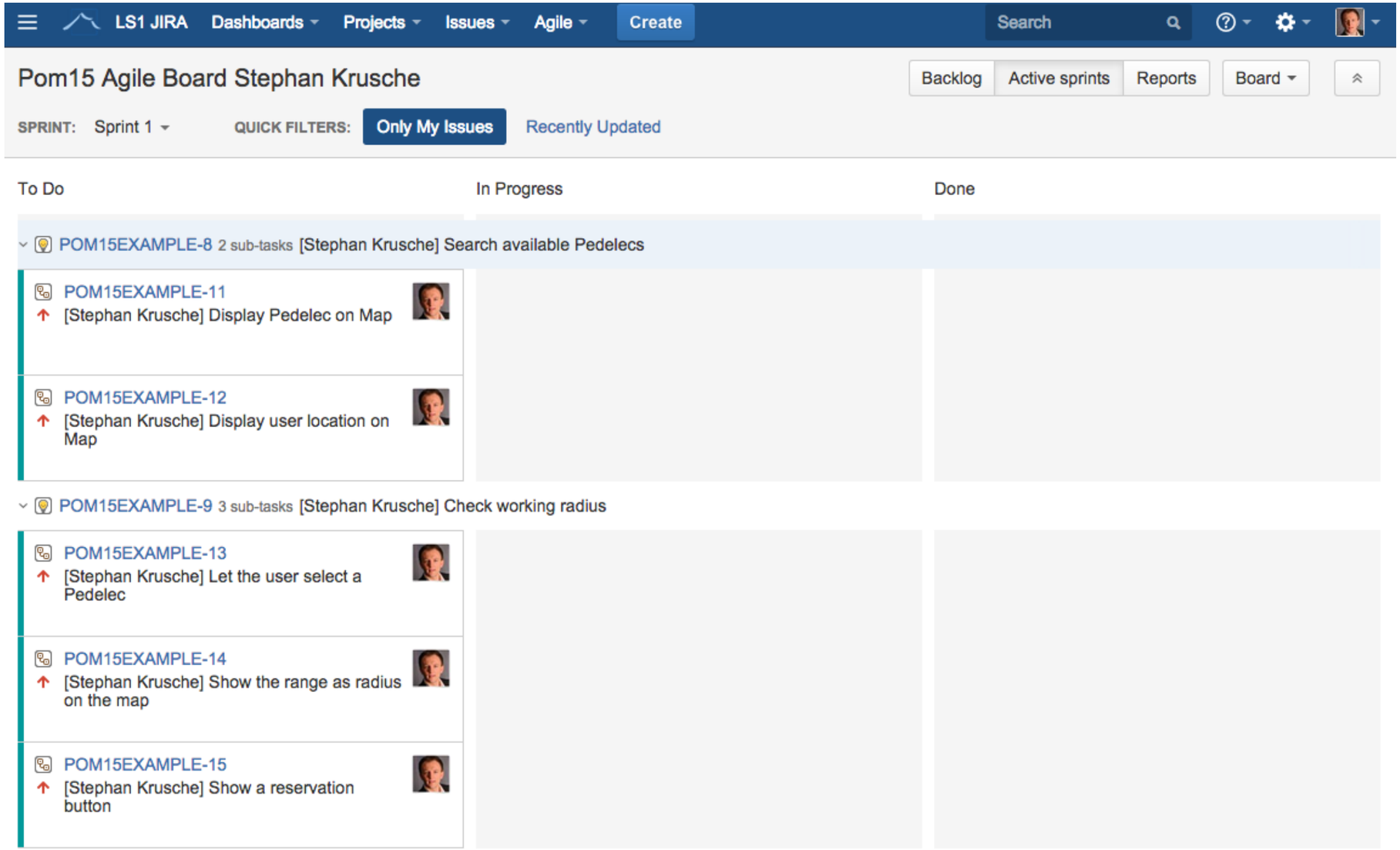
Choose a summary that starts with **[Your Full Name]** and assign to yourself

Click on Create

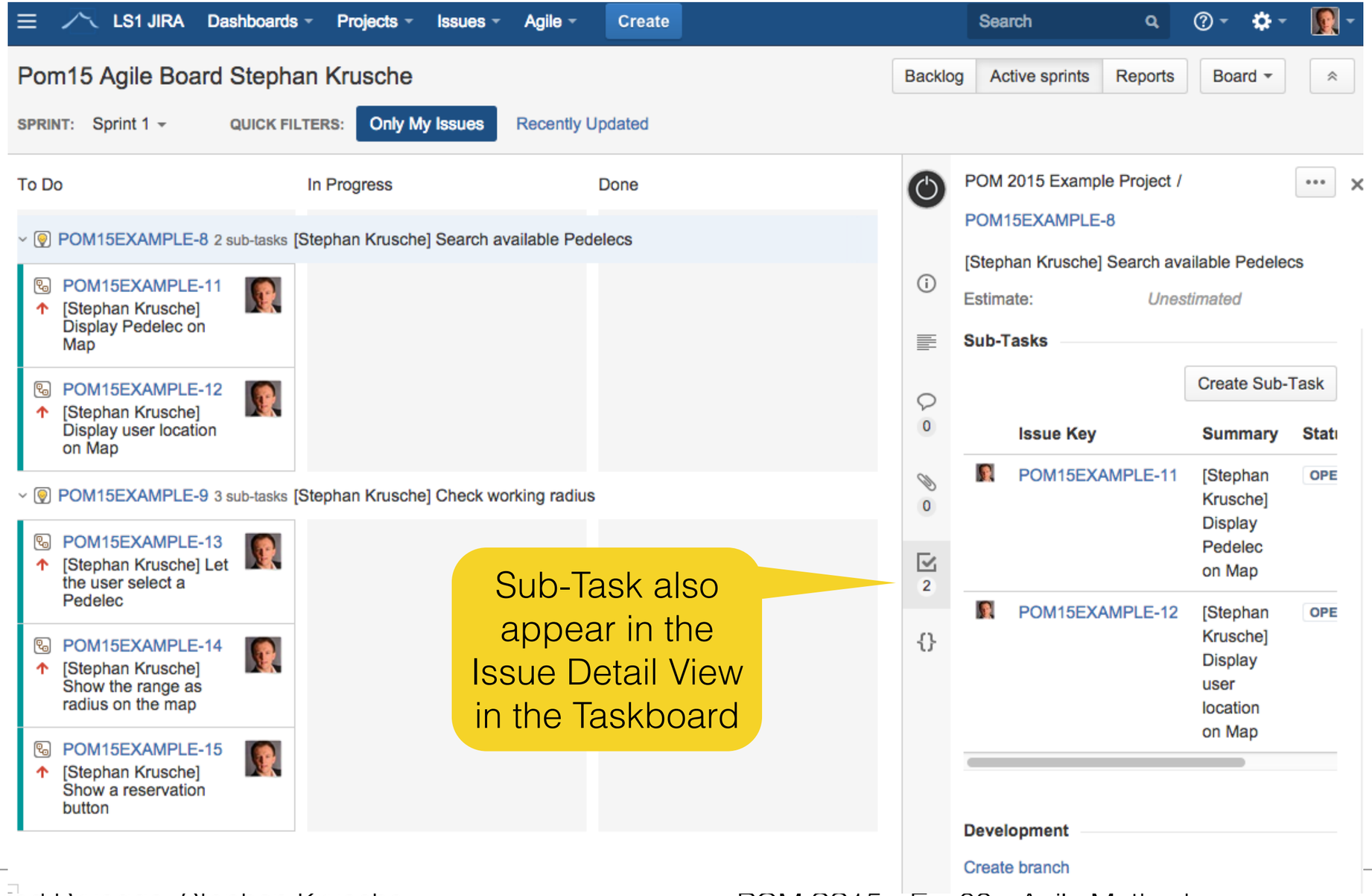
Task 12: Create Sub-Tasks for Sprint 1 Backlog Items and assign it to yourself

Backlog Item with 2 Sub-Tasks

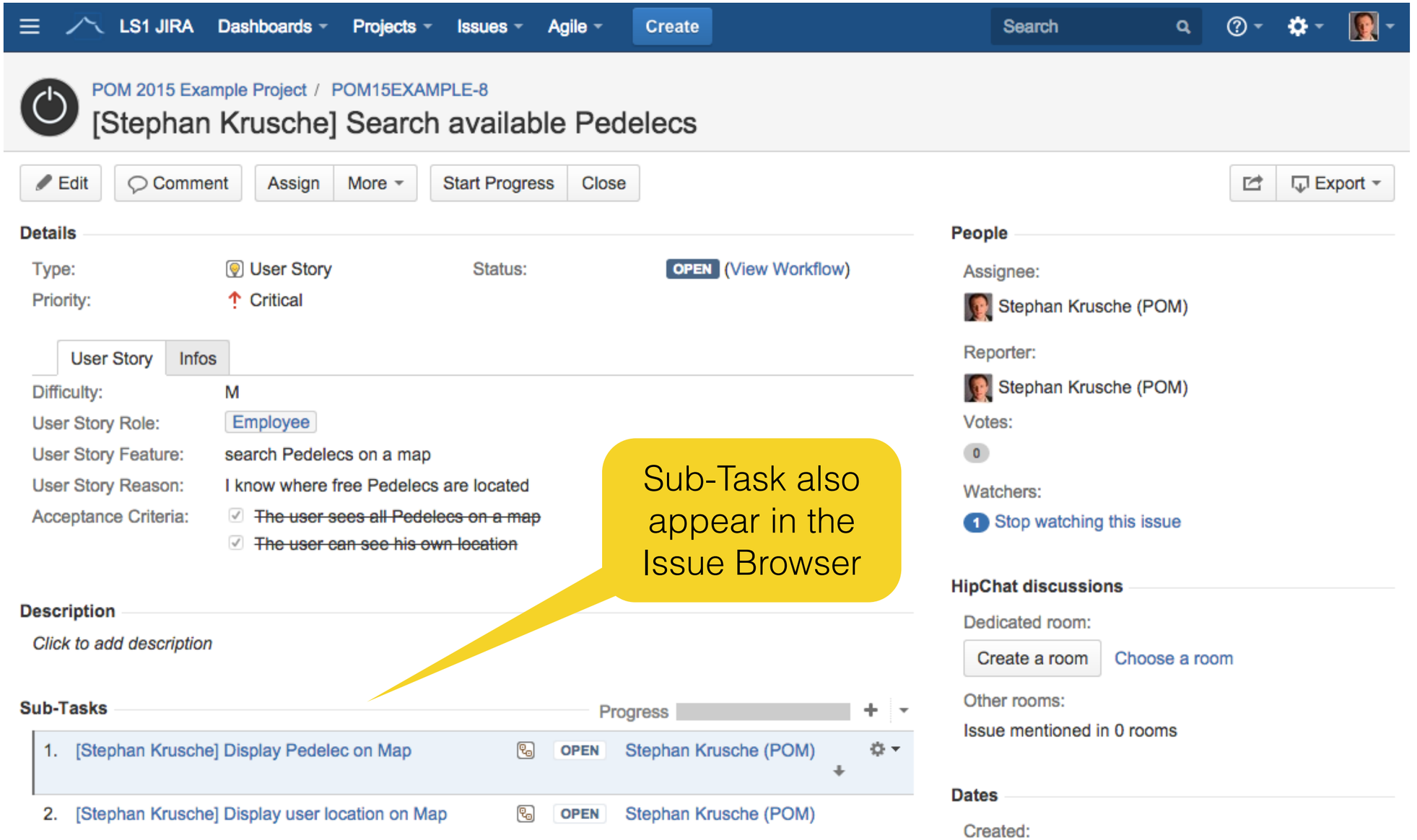
Backlog Item with 3 Sub-Tasks



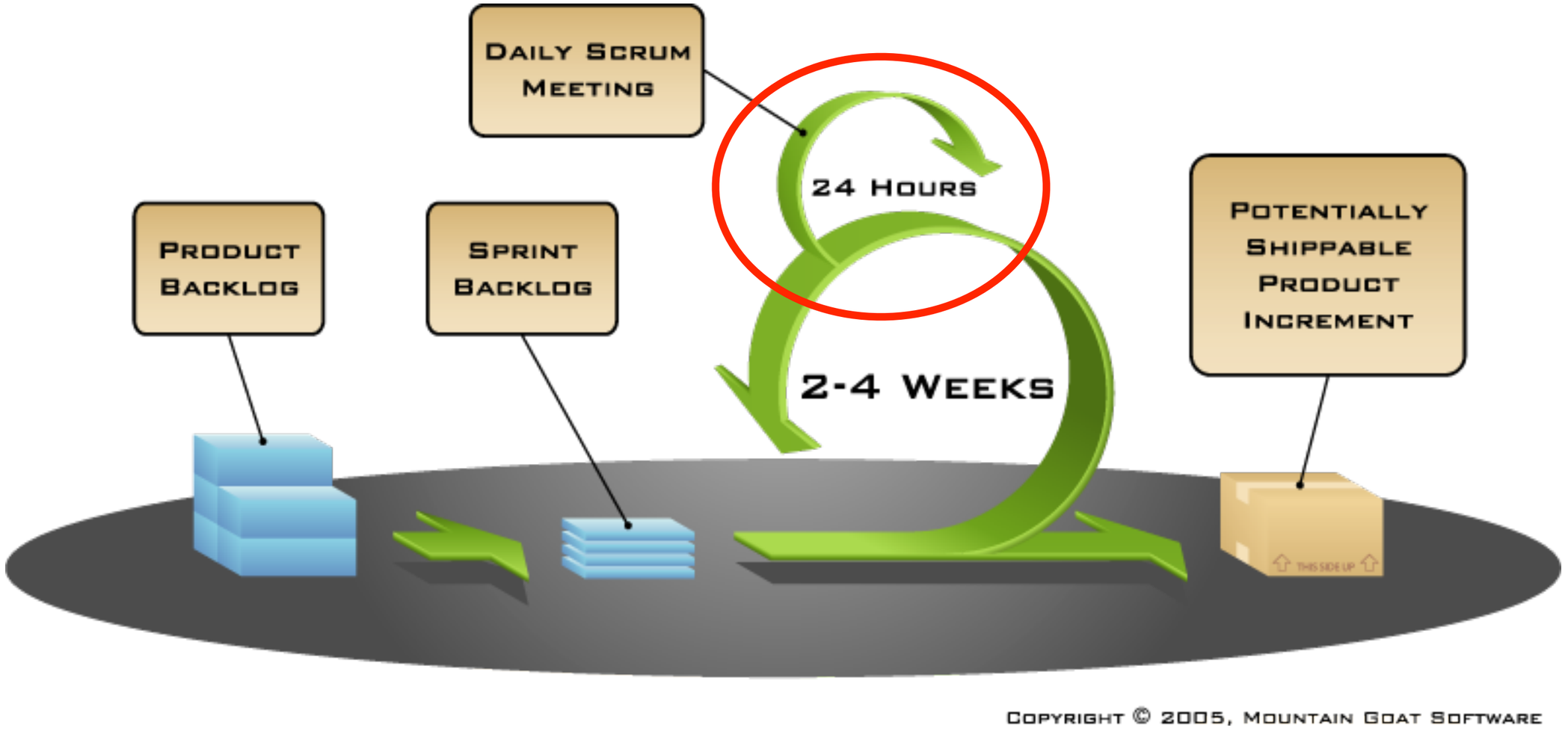
Task 12: Create Sub-Tasks for Sprint 1 Backlog Items and assign it to yourself



Task 12: Create Sub-Tasks for Sprint 1 Backlog Items and assign it to yourself



Exercise 5: Working on the Taskboard during the Sprint



Daily Scrum Meeting

- **Main purpose:** Risk reduction by early information sharing and discussion
- 15 min standup meeting every day
- Every developer answers the following three 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?

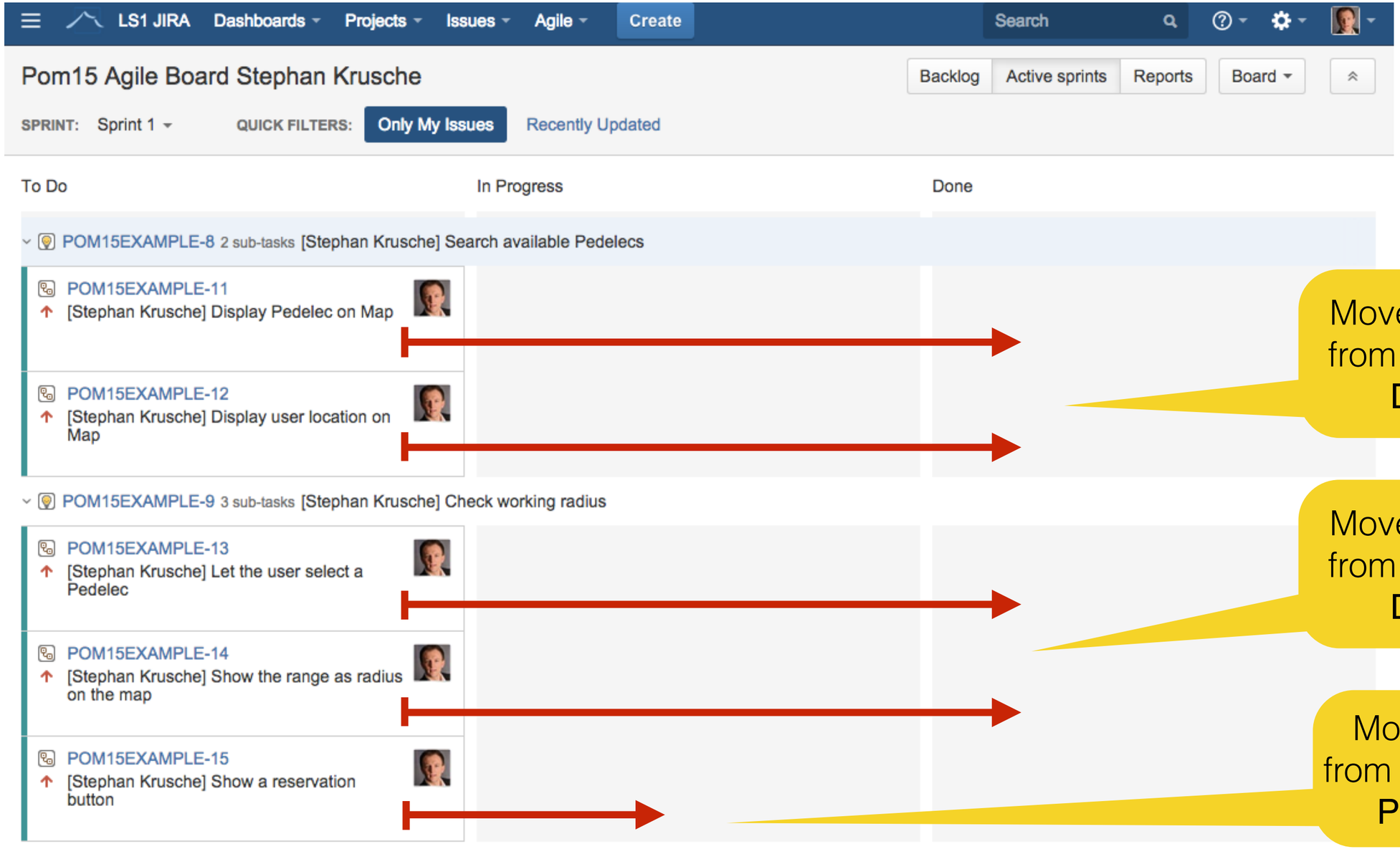
What would you do if you cannot meet daily?

Time:
5 min

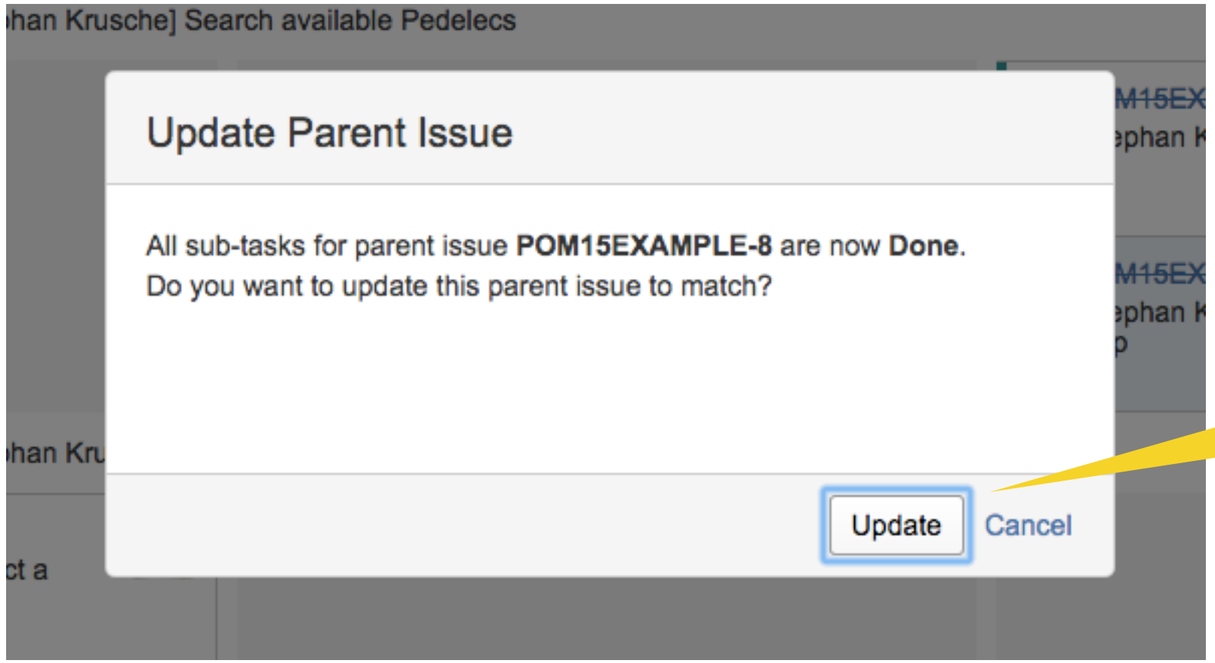
Exercise 5: Working on the Taskboard during the Sprint

Task 13: Open the Taskboard and start/resolve Sub-Tasks

Task 13: Open the Taskboard and start/resolve Sub-Tasks

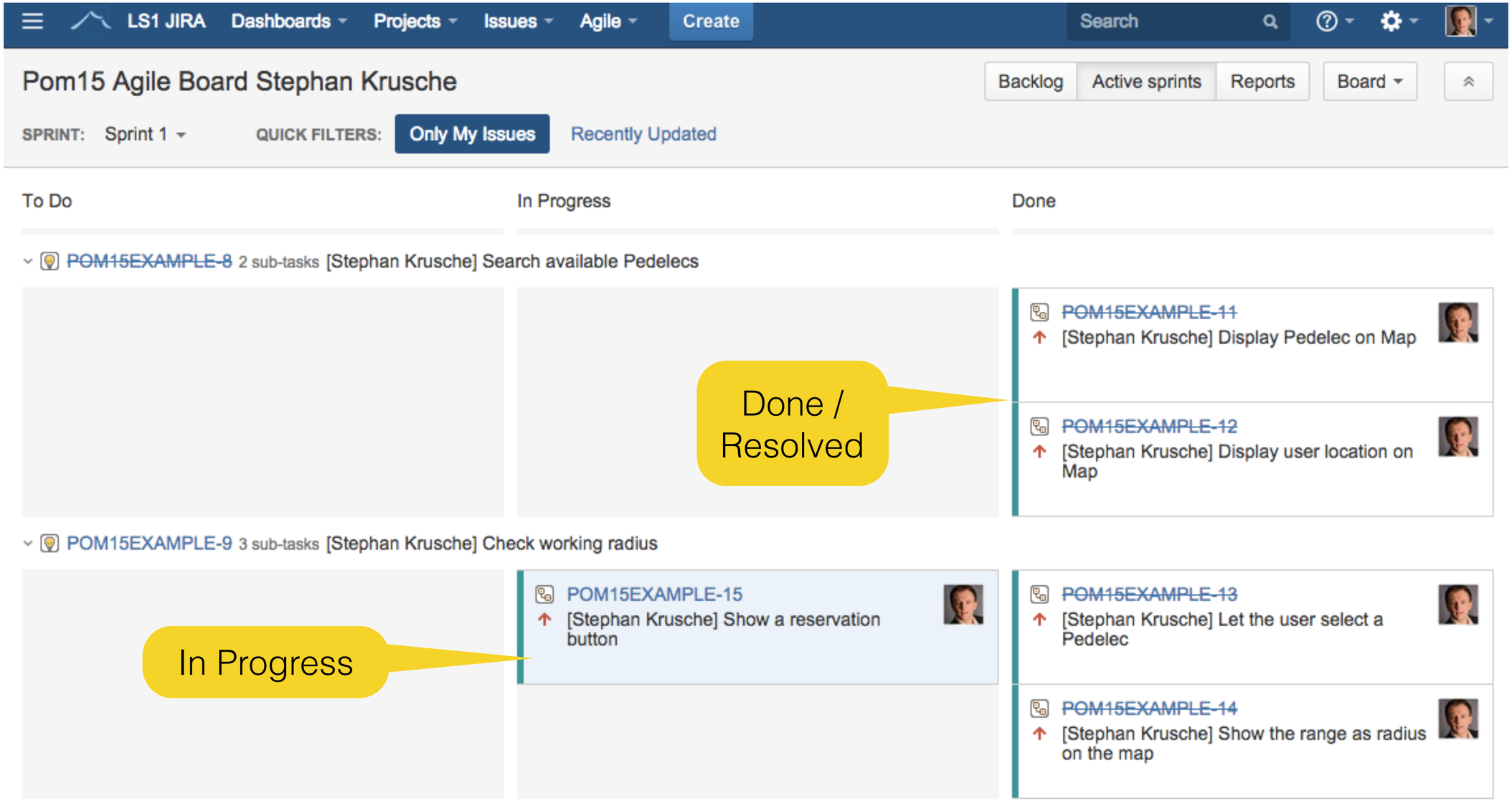


Task 13: Open the Taskboard and move subtasks

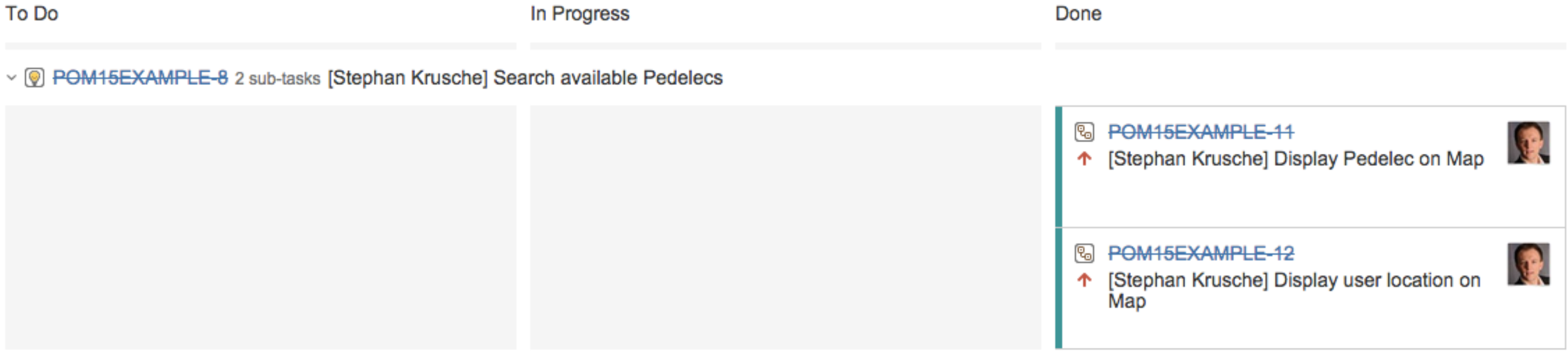


Choose update to also close the User Story

Task 13: Open the Taskboard and move subtasks

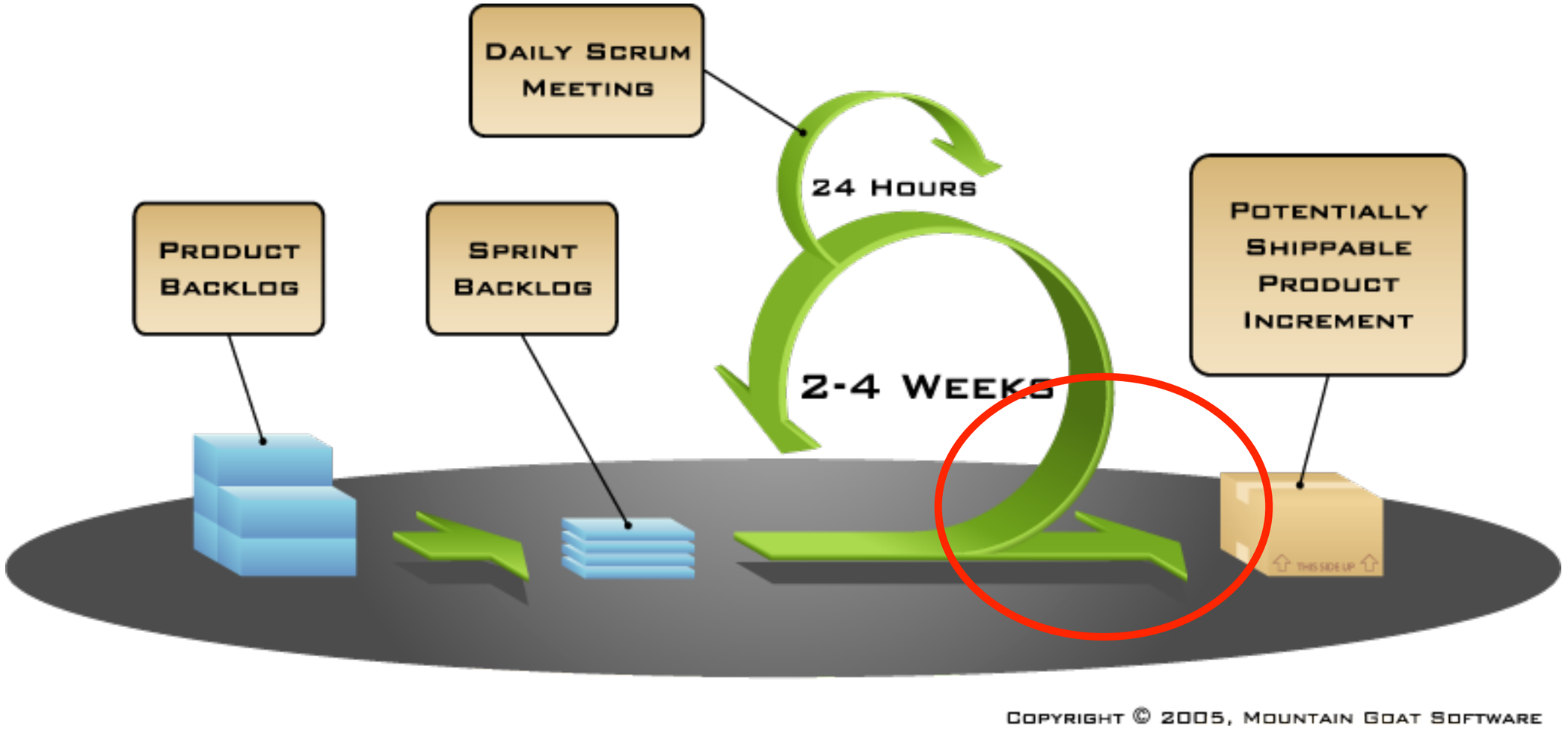


Issue Workflow in JIRA



- You can also setup JIRA Agile to show an additional column for the state **In Review**

Exercise 6: Sprint Review



Sprint Review Meeting

- The Development Team delivers a Product Increment including the realized items from the Sprint Backlog
  - Deliver the application 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) gives feedback and decides whether the items are realized completely
  - Realized items are ticked off in the Sprint Backlog

Sprint 1 Backlog

ID	Name	Difficulty
1	☑ Search available Pedelecs	Medium
	☑ Display Pedelec on Map	
	☑ Display user location on Map	
2	☐ Check working radius	Large
	☑ Let the user select a Pedelec	
	☑ Show the range as radius on the map	
	☐ Show a reservation button	

Sprint Review Meeting (continued)

- 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)

Sprint 1 Backlog

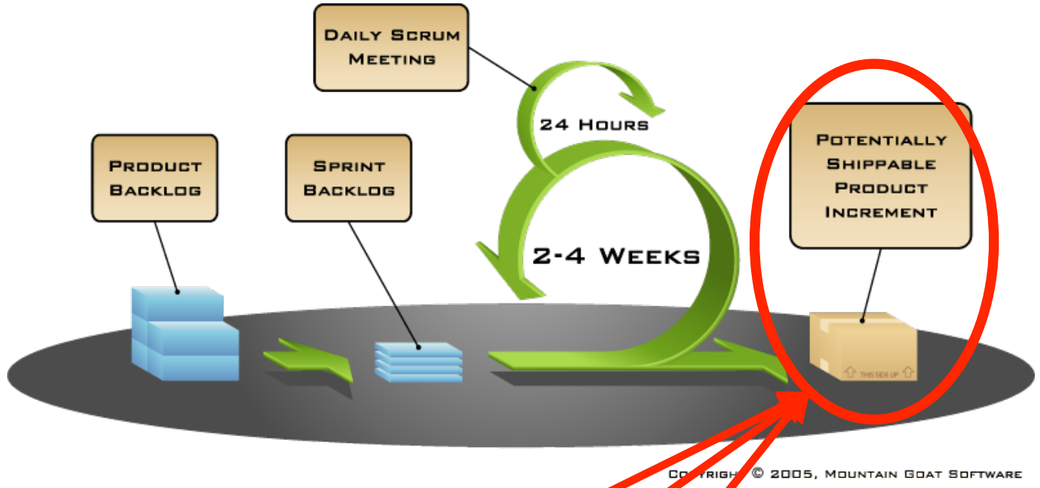
ID	Name	Difficulty
1	☑ Search available Pedelecs	Medium
	☑ Display Pedelec on Map	
	☑ Display user location on Map	
2	☐ Check working radius	Large
	☑ Let the user select a Pedelec	
	☑ Show the range as radius on the map	
	☐ Show a reservation button	

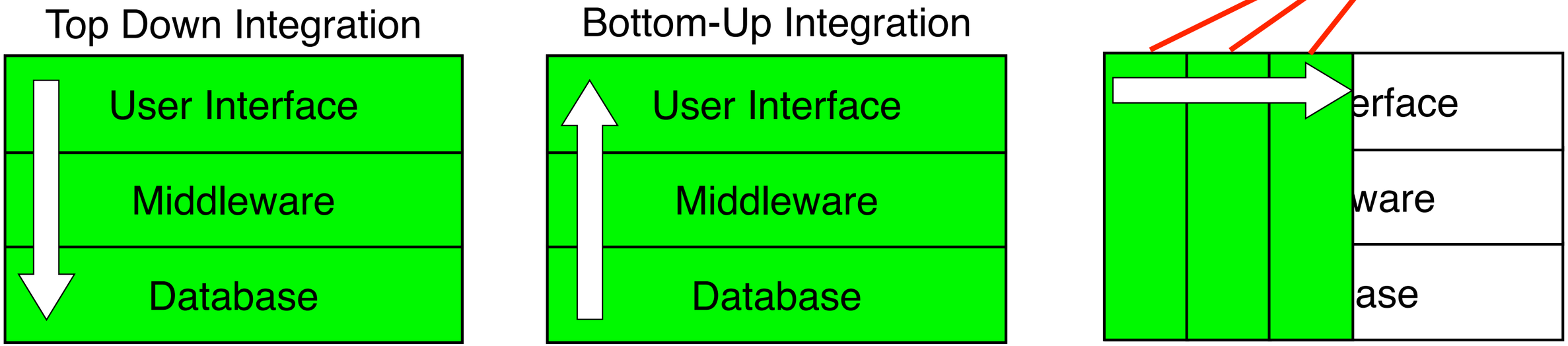
Prioritized and Estimated Product Backlog

ID	Name	Priority	Difficulty
1	☑ Search available Pedelecs	Critical	Medium
2	☐ Check working radius	Critical	Large
3	☐ Reserve available Pedelec	Critical	Small
4	☐ Return Pedelec	Major	Small
5	☐ Contact colleague	Minor	Medium
6	☐ Pass reservation	Minor	Medium
7	☐ Report damage	Minor	Large
8	☐ Unlock Pedelec	Major	Small
9	☐ Notification after Repair	Minor	?

Potentially Shippable Product Increment

- Each sprint focuses on the incremental creation of an working system
- Can be thrown away ("falsified") or delivered
- The product owner decides





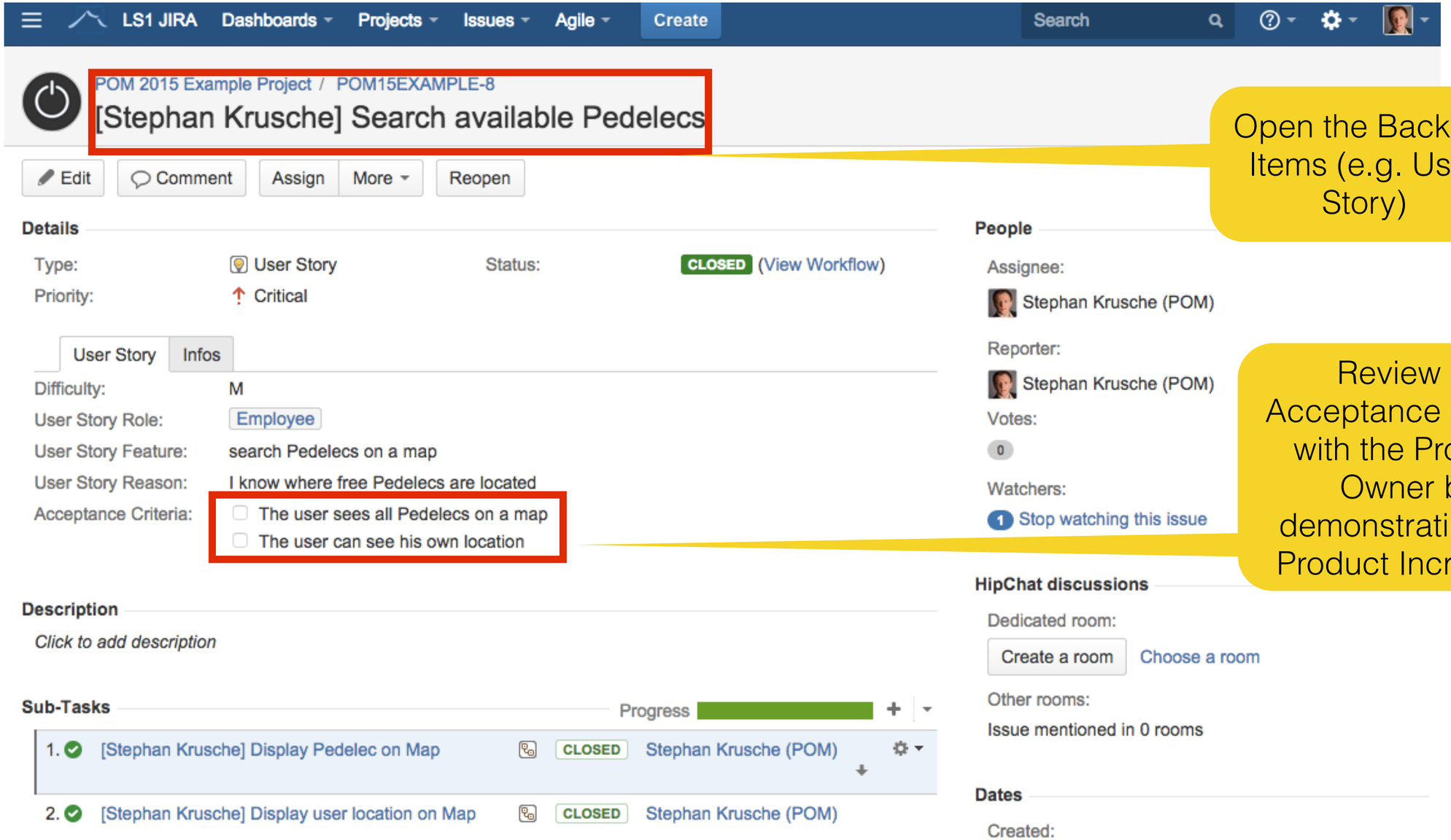
Horizontal Integration **Vertical Integration**

Time:
5 min

Exercise 6: Sprint Review

Task 13: Review Acceptance Criteria
Task 14: Finish the Sprint
Task 15: Inspect the Sprint Report

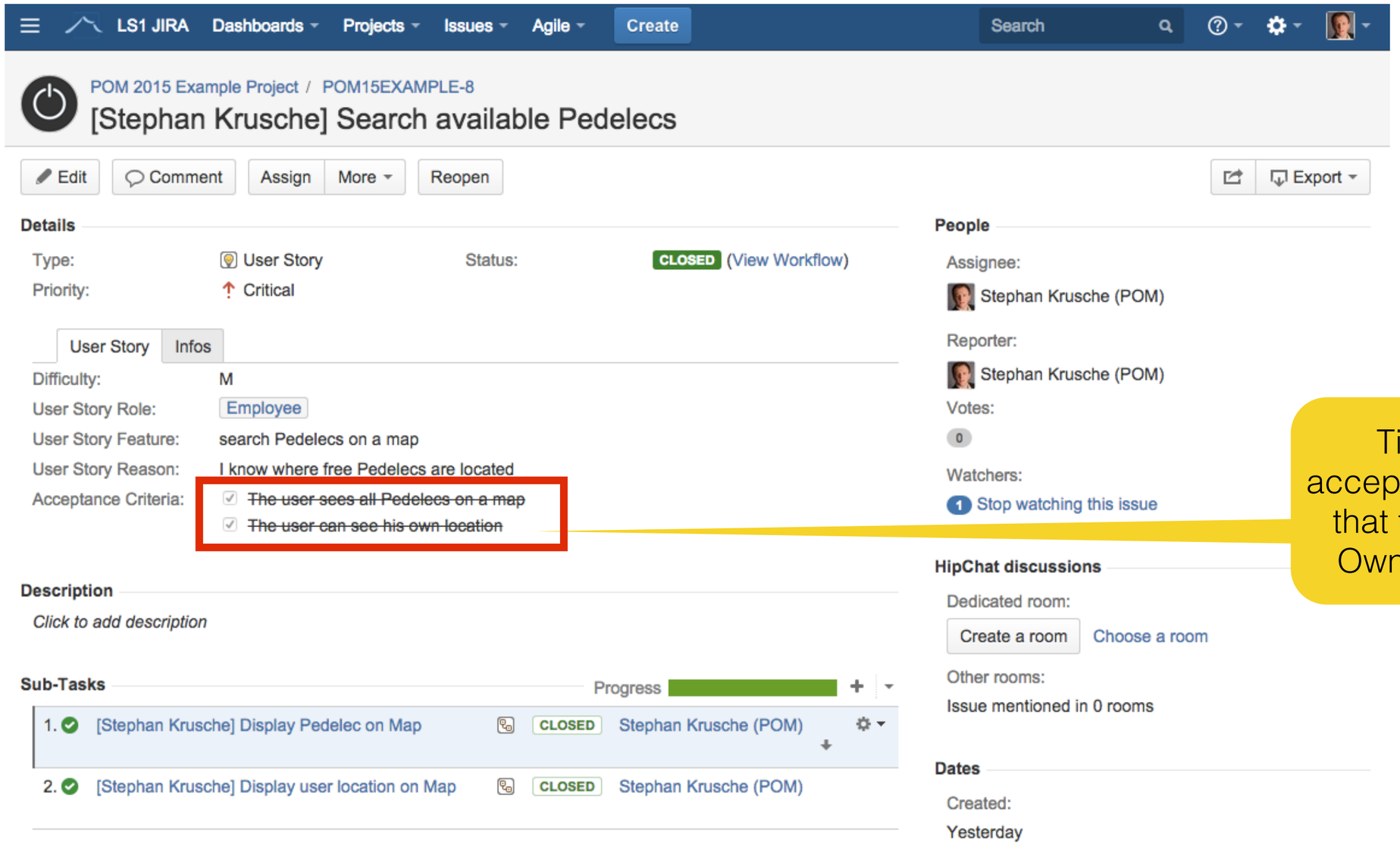
Task 13: Review Acceptance Criteria



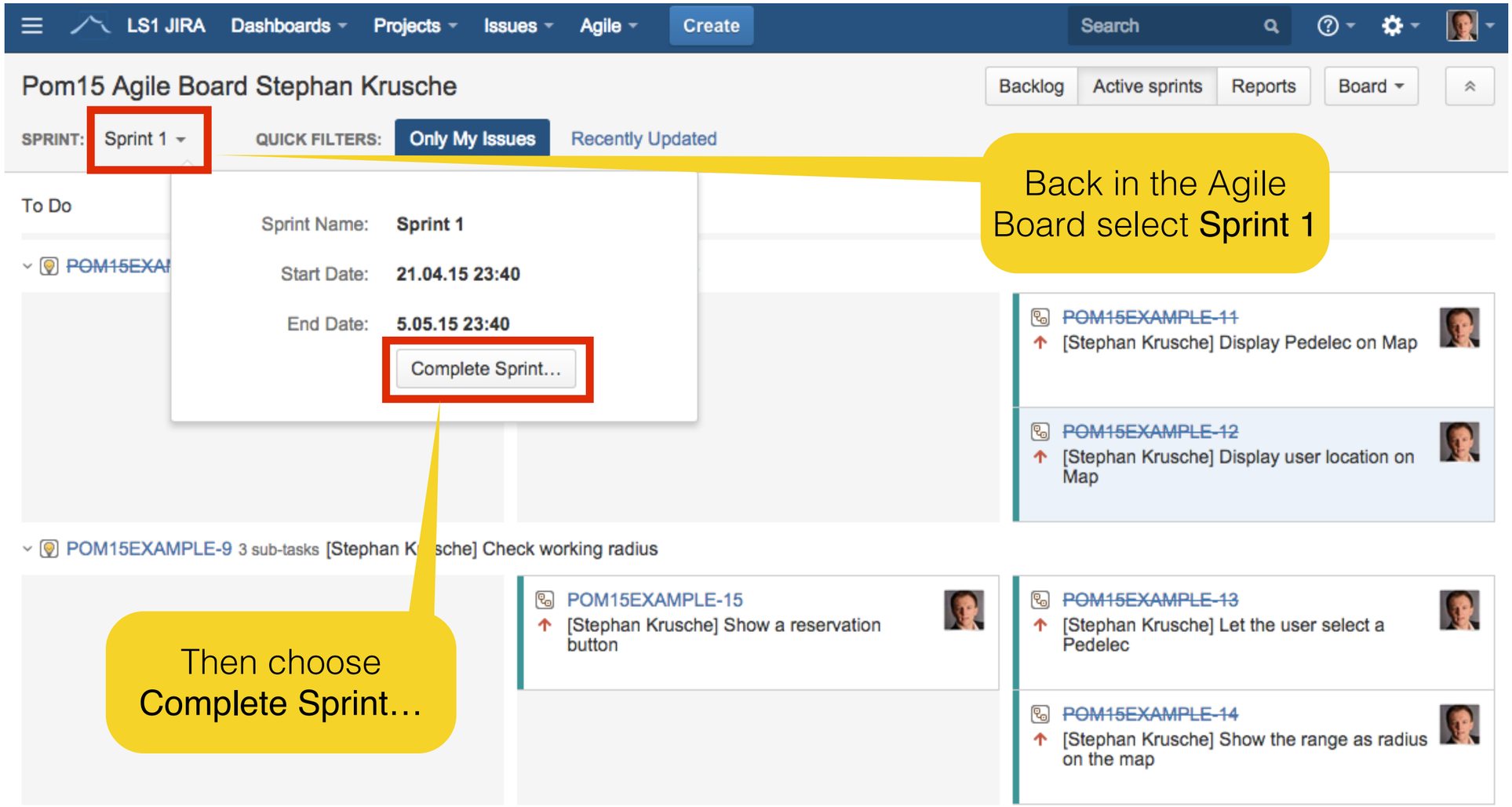
Open the Backlog Items (e.g. User Story)

Review all Acceptance Criteria with the Product Owner by demonstrating the Product Increment

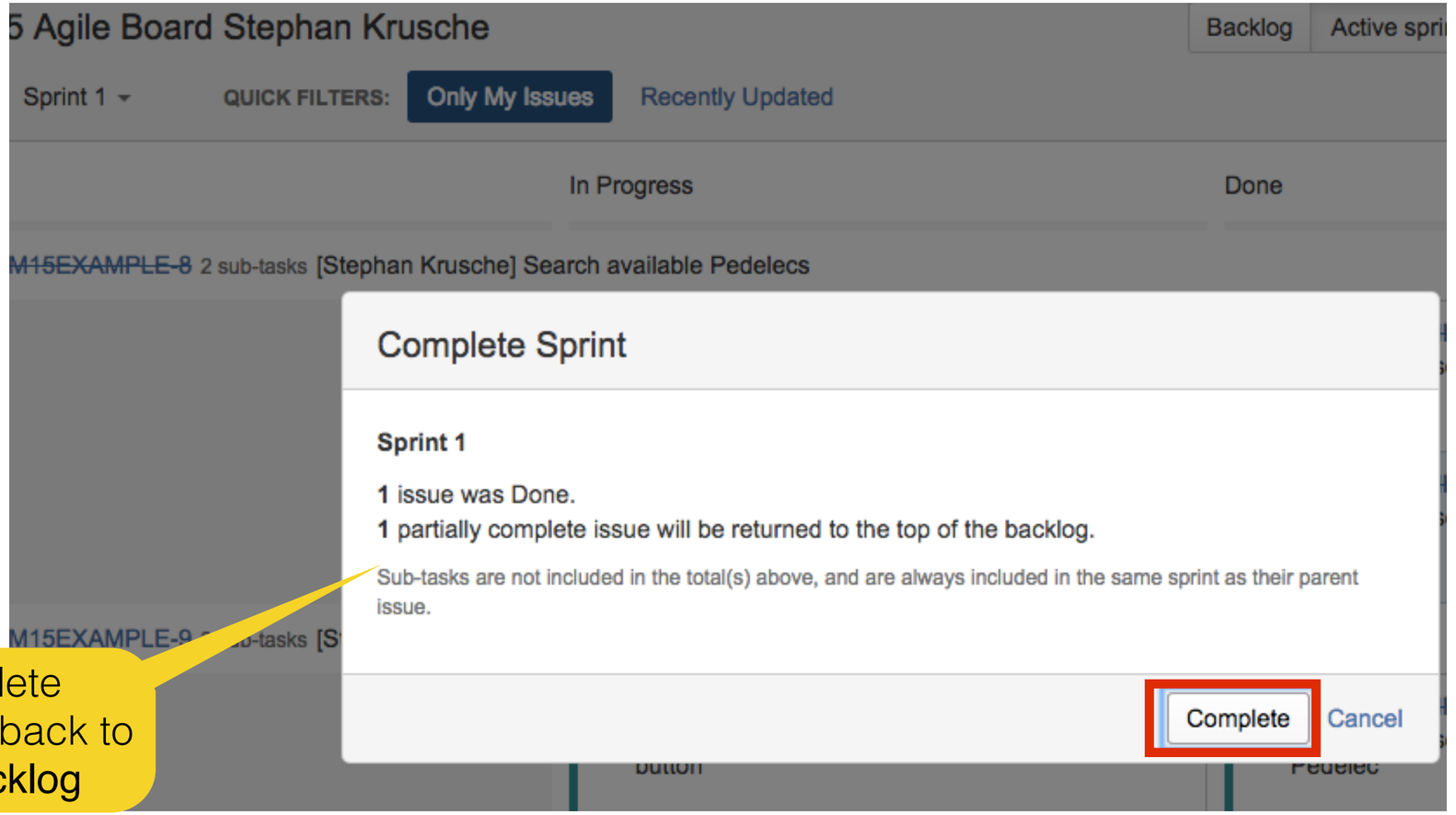
Task 13: Review Acceptance Criteria



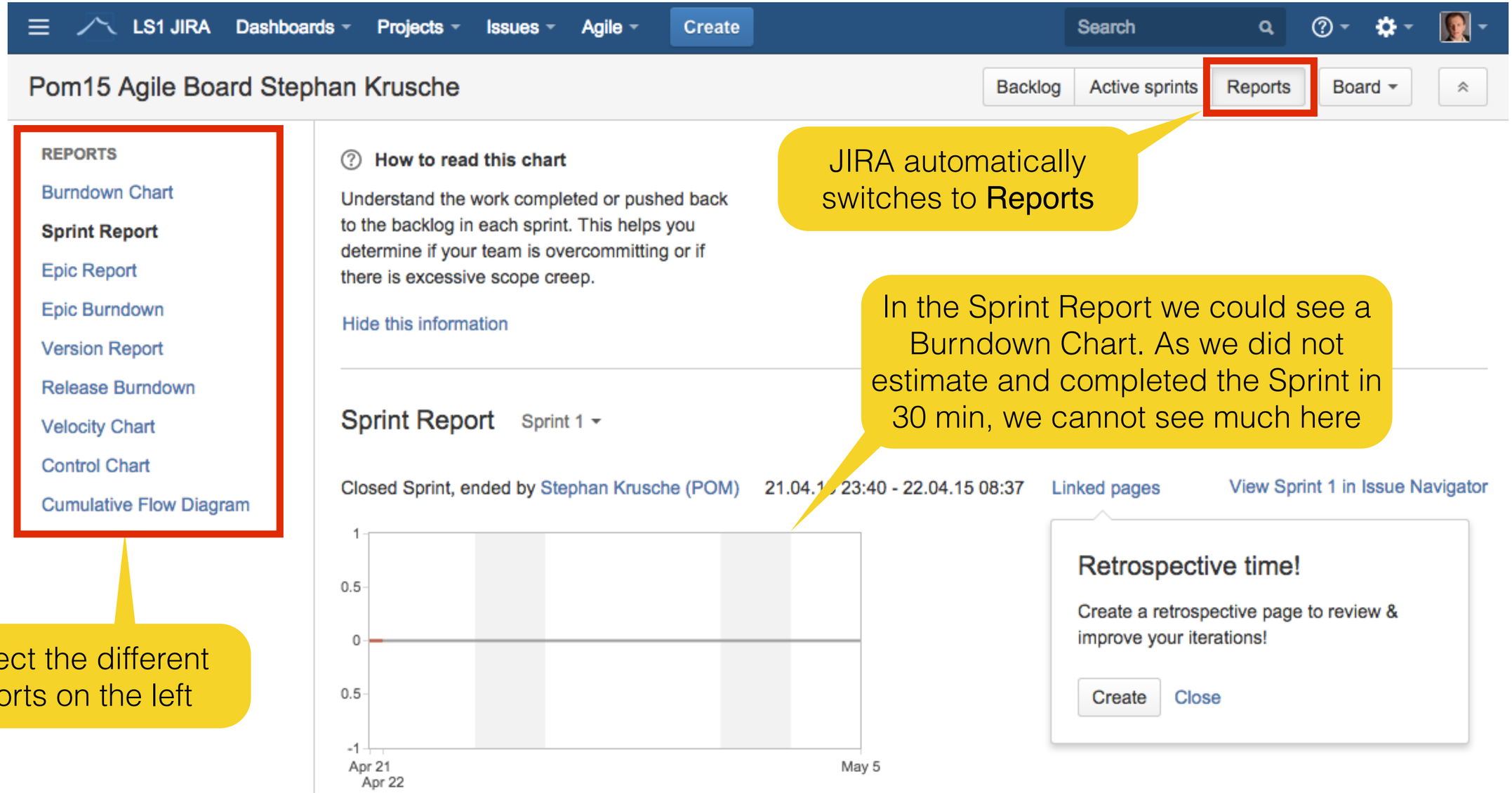
Task 14: Finish the Sprint



Task 14: Finish the Sprint



Task 14: Inspect the Sprint Report



Sprint Retrospective Meeting

- Scrum Master and Development Team meet to discuss how the previous Sprint worked out
- The Product Owner can also participate if necessary
- There are different Retrospective techniques
- Often it is most effective to brainstorm about things that worked well and things that did not work well
- Each team member is asked to identify specific things that the team should:
  - Start doing
  - Stop doing
  - Continue doing

Scrum as Methodology

- Involvement of the customer
  - Onsite customer (“co-located”)
- 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
  - **3 Artifacts:** Product Backlog, Sprint Backlog, Potentially Shippable Product Increment
  - **4 Meetings:** Project Kickoff, Sprint Planning, Daily Scrum, Sprint Review
  - **3 Roles:** Scrum Master, Product Owner, Developer
- JIRA Agile
  - 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

References

- Stephan Krusche, Lukas Alperowitz, Bernd Bruegge and Martin Wagner - Rugby: An Agile Process Model Based on Continuous Delivery, 1st International Workshop on Rapid Continuous Software Engineering (RCoSE'14), June 2014
- Stephan Krusche and Lukas Alperowitz - Introduction of Continuous Delivery in Multi-Customer Project Courses, 36th International Conference on Software Engineering (ICSE'14), Software Engineering Education and Training (SEET), June 2014
- Ken Schwaber and Mike Beedle - Agile Software Development with Scrum, Prentice Hall, October 2001
- Mike Cohn - User Stories Applied - For Agile Software Development, Addison-Wesley, March 2004
- Mike Cohn - Agile Estimating and Planning, Prentice Hall, November 2005
- Bernd Bruegge and Allen H. Dutoit - Object Oriented Software Engineering using UML, Design Patterns and Java, 2010
- Agiles Manifesto - http://agilemanifesto.org
- Mike Cohn, Mountain Goat Software - http://www.mountaingoatsoftware.com
- Agile42, Scrum Cheat Sheet - http://media.agile42.com/content/Scrum_Cheat_Sheet_09_09.pdf
- Bill Wake - INVEST in Good Stories, and SMART Tasks - http://xp123.com/articles/invest-in-good-stories-and-smart-tasks
- Meet the new gold standard in workflow automation - http://blogs.atlassian.com/2014/08/meet-new-gold-standard-workflow-automation
- 8 Steps to a Definition of Done in JIRA - http://blogs.atlassian.com/2013/10/8-steps-to-a-definition-of-done-in-jira
- Sprint Retrospective - http://www.mountaingoatsoftware.com/agile/scrum/sprint-retrospective