Professional Scrum Developer (Java)

The Professional Scrum Developer (PSD) course includes training, assessment, and certification. It consists of a a five-day course, an online assessment, and an industry-recognized certification.

Professional Scrum Developer course teaches students how to use modern software engineering practices to develop an increment of potentially shippable functionality using Java. Students learn to do so within the Scrum framework, working as part of a self-organizing, cross-functional team to do iterative, incremental development. Classes are exercise-driven, with students working in teams to develop "done" increments from product backlog items. Each class is five days long, and classes can be either public or private.

Professional Scrum Developer course covers three main topics:

1. **Scrum.** PSD course covers Scrum Fundamentals like Scrum roles, artifacts, and processes. The course simulates being part of a Scrum team to expose students to these concepts in action. Students learn how to work as part of a Scrum team, which requires them to understand techniques for self-organization. At the end of the course students develop skills in identifying and eliminate typical types of Scrum team dysfunction.
2. **Tools.** PSD courses teach students how to leverage different development tools to employ Scrum practices. PSD Java courses are taught in the context of either Eclipse + IBM Rational Jazz or Eclipse + Open Source Plug-Ins. Students learn how to map specific tool features and functions to the general Scrum practices they must use to be effective team members.

3. **Best Practices.** PSD courses cover all of the technical practices that team members need to successfully implement and ship functionality. These include coding practices like test-driven development, continuous integration, and refactoring; architecture practices such as emerging architecture and evolutionary database development; release management practices like planning, requirements definition, and shipment, and quality assurance practices from defining "done" to pair programming to version control to acceptance testing.

Course Objectives

Course attendees will learn about:

Scrum

- Scrum Overview
- Scrum Roles
- Scrum Artifacts (Product Backlog, Sprint Backlog, Release Burndown, Sprint Burndown)
- Sprint Planning Meeting
- Sprint Execution
- Daily Scrum
- Review
- Retrospective

Agile Development Practices

- User Stories
- Estimating with Story Points and Planning Poker
- Done Criteria
- Test Driven Development
- Code Smells and Refactoring
- Continuous Integration
- Pair Programming
- Verify that bugs are identified and eliminated

Theory and Foundation

- Self-Organization
- Pull versus Push principle
- Improving with Inspect and Adapt
- Acceptance Criteria

Each team is expected to self-organize and manage their own work during the sprint. Pairing is highly encouraged. The instructor/product owner will be available if there are questions or impediments, but will be hands-off by default. You should be prepared to communicate and work with your team members in order to achieve your sprint goal. If you have development-related questions or get stuck, your partner or team should be first level support.

Syllabus

The Modules 5-13 are embedded into the mini Sprints. Usually at the beginning of a virtual day after the Daily Scrum, the students will be taught some theory before they get back to their Sprints. It is intended, that this knowledge can be applied immediately or in one of the subsequent sprints.

Module 1: Introduction

This module provides a chance for attendees to get to know the instructors as well as each others. The Professional Scrum Developer program, as well as the day by day agenda, will be explained. Finally, the Scrum Teams will be selected and assembled so that the forming, storming, norming, and performing can begin.

- Trainer and student introductions
- Professional Scrum Developer program
- Agenda
- Logistics
- Team formation

Module 2: Introduction to Scrum

This module provides a level-setting understanding of the Scrum framework including the roles, timeboxes, and artifacts. The team will then experience Scrum firsthand by simulating a mini release with mini sprints of product development, including planning, review and retrospectives meetings.

- Scrum overview
- Scrum roles
- Scrum timeboxes (ceremonies)
- Scrum artifacts
- Motivation for Scrum
- Retrospective

It is required that you read the Scrum Guide in preparation of this module.

Module 3: Familiarizing with the 'brownfield' application

This module is intended to get the students hands dirty with the existing 'brownfield' application. They have to fix a couple of broken tests. Thereby, they have a chance to familiarize with the source code, the existing testing framework and the IDE.

- Project environment
- Source code structure
- Test framework
- Eclipse IDE

Module 4: Sprint Planning Meeting

This module explains what happens during the Sprint Planning Meeting. How the team is enabled to provide a realistic commitment for a set of Product Backlog items, the 'What'

in the Sprint Planning 1 and how it translates the ‘What’ into ‘How’ during the Sprint Planning 2.

- Agile planning
- Velocity
- Capacity
- Sprint goal
- Sprint planning 1
- Sprint planning 2
- Acceptance criteria

Module 5: Daily Scrum

This module describes in depth the idea behind the Daily Scrum, how it is facilitated and what to watch out for.

- Participants
- The three questions
- Examples and outcome
- Impediments

Module 6: Self-Organization (Pull vs. Push)

This module highlights the reason why self-organization is so important and how it can be facilitated. It looks in detail into the dynamics of a team and what environmental risks have to be mitigated in order to develop a strong self-organizing team.

- Reasons for self-organizing
- Team versus group
- Forming – Storming – Norming – Performing
- Environment
- Risks

Module 7: Pair Programming

This module looks into the history of pair programming, why it works and how to implement it right. Also, it shows reasons why it does make sense to work in pairs.

- Driving forces behind pair programming
- Typical setup for pair programming
- Good pairing behavior
- Pairing combinations
- Privacy concerns
- Reasons why it is beneficial to do pair programming

Module 8: Review and Retrospective

This module describes how to conduct a successful Review. It shows what is important and what the common pitfalls are. The Retrospective part addresses the importance of

inspect and adapt on the team level. How the outcome and action items of a retrospective are important for the self-organization of the team.

- Planning a review
- What to do and what is to be avoided
- How to handle feed back
- Inspect and adapt
- Plan – Do – Check – Act from the past into the future
- Techniques for effective retrospectives

Module 9: User Stories, Definition of Done, Planning Poker

This module explains why User Stories are a good for requirements elicitation, why they are no specification and how they can be validated during the sprint. How the Definition of Done relies on the Acceptance Criteria and why a company appropriate Definition of Done is so important. Finally, the art of Planning Poker is explained.

- Customer value, priority, effort
- Card – Conversation – Confirmation
- INVEST principle
- Acceptance criteria
- SMART principle
- Definition of done
- Planning Poker and why it works
- How to play poker

Module 10: Test Driven Development, Dependency Injection

This module looks into the history of Pair Programming, why it works and why it makes sense to do it. Testing is easy when a system is testable. Dependency Injection is one great technique to achieve this.

- The cycle of test, code and refactor
- The three laws of TDD
- 3A Pattern (Arrange, Act, Assert)
- Reasons why it is beneficial to do TDD
- Agile testing quadrants
- Dependency injection
- Mocking

Module 11: Code Smells, Refactoring, Design for Testability

This module describes the various kinds of code smells, how they can be eliminated by refactoring; this module includes a life example given by the instructor. Furthermore this module looks into code attributes which make testing easy.

- Types of code smells
- Law of Demeter

- Live refactoring example
- Design for testability

Module 12: Continuous Integration

This module looks into the history of continuous integration, why it is important to integrate often. Also, this module explains how continuous integration can be used for more then building and testing, but to enforce company standards and source metrics.

- Continuous integration setup
- Big visual display / eXtreme feedback devices
- Types of build processes
- How it can be rolled-out
- Build artifacts
- Team behavior

Module 13: Pair Programming, Test Driven Development and Tooling Revisited

This module describes in depth and detailed the mechanics of Pair Programming, Test Driven Development and tooling. Various tools are addressed and in which context they should be used.

- Agile tools
- Instructor-led examples

Module 14: Scrum Immersion

After the completion of the last sprint the remainder of the day is set for the immersion of Scrum. In a self-organizing way, small teams are forming and prepare presentations of the learned topics. After preparation, each team summarizes and presents to the whole group. The instructor moderates the lessons learned sessions and provides more information as required.

- Publicly speak about Scrum and describe how it works
- Specific and general Q&A

Expectations

This is a unique course in that it is technically-focused, team based, and employs timeboxes. It demands that the members of the team self-organize and self-manage their own work to collaboratively develop increments of software.

All students must commit to:

- Pay attention to all lectures and demonstrations
- Participate in team and group discussions
- Work collaboratively with other team members
- Obey the timebox for each activity
- Commit to work and do your best to deliver

All students should have:

- An understanding of Scrum

- Familiarity with Eclipse IDE for Java
- Java experience
- Understanding of relational databases is beneficial
- Software testing experience

People who should not take this course include:

- Students requiring command and control style instruction (there are no prescriptive/step-by-step labs in this course)
- Students who are unwilling to work within a timebox
- Students who are unwilling to collaboratively on a team
- Students who don't have any skill in any of the software development disciplines
- Students who are unable to commit fully to their team (not only will this diminish the student's learning experience, but it will also impact their team's learning experience)

Assessment and Certification

The Professional Scrum Developer assessment measures knowledge of how to develop software using Scrum on a specific technology platform. The Professional Scrum Developer assessment takes 60 minutes; a passing score of 90% is required for Professional Scrum Developer I certification.

You can find course schedule at http://www.accelright.com/courses