Agile Software Development Practices
Duration: 5 days (9 hours each day with working lunch)

Agile development is all about succeeding in our efforts with the help of short, meaningful feedback loops. If we're not succeeding, then we ought to change our practices. If we are succeeding, then we should not care what to call it. The goal is not to be agile, but to succeed. With that thought in mind, what set of practices, when done correctly, can help us improve the chances of success and lower the chances of failures? That's the question this course will answer. This course takes a highly practical and result driven approach to being agile. It will present the major impediments, things we have to deeply introspect and change. It will discuss why adaptive planning is highly essential and how to get the team to working in that mode. It has a good mixture of technical and non-technical details to help adapt, succeed and sustain agile development.

The course has a good balance of interactive lectures and hands-on exercises. The attendees are expected to pair-up and work on the lab exercises. The instructor will assist the attendees as they work on the labs. The objective of the course is for the attendees to gain an in depth practical knowledge of the concepts so they can put them to immediate use on their real projects.

The course content will be customized to meet your teams' specific needs. Please review this detailed outline and suggest changes (additions, deletions, modifications) as you feel fit.

Topics

Agile Development
* Agility
* What's Agile?
* Evolution of fields
* Rate of success in software development
* Software development vs. predictive manufacturing
* Innovative Gaming
* Process
* Iterative and incremental
* Project Schedule
* Pick Two:Time, Scope, Quality
* Agile Development Process
* The Agile Manifesto
* Agile Principles
* What's agile development, again?
* Exercise

Agile Processes
* Methodologies
* Why methodologies?
* eXtreme Programming (XP)
* Control Variables
* Set of Values and Principles
* Scrum
* Scrum values and Lifecycle
* Feedback cycles
* Automation and agile testing
* Tenets of testing
* Price of Automation, Price of not doing it
* Agile Team
* Kanban
* Exercise

Software Planning
* Planning
* Requirements: Why is it hard?
* Minimizing Risk
* Gathering Requirements
* User Stories
* Creating Stories
* Granularity, INVEST,
* Development Process
* Exploration and scoping
* Estimation
* Release Planning
* Iteration/Sprint Planning
* Task Planning, Progress vs. Planning
* Spiking, Halfway point, slippage
* Measuring Progress
* Exercise

Adaptive Planning and Execution
* Iteration vs. Increments
* Why adaptive planning?
* What's adaptive planning?
* Adapting scope or date
* Feedback and communication
* Who's responsible for planning?
* Who's responsible for delivering?
* What makes an agile team?
* Backlogs
* Release planning
* Sprint/iteration planning
* Sprint/Iteration length
* Task planning and estimation
* Demo at the end of Sprint

* Scheduling Sprint/Iterations
* Standup meetings
* Retrospections
* Exercises

Estimation and Planning
* Estimation Issues
* Estimates, targets, commitments
* Estimates vs. plans
* Prioritizing
* Good Estimates
* Team Responsibilities
* Right degree of estimation
* Realities of estimation
* Spiking
* Estimation techniques
* Story progression
* Pragmatic Scrum
* Story estimation
* Measuring velocity
* Splitting stories
* Exercises

Creating User Stories
* Good user stories
* Writing user stories
* Story sizes
* Discovering stories
* Story creation cycle
* Exercise

Design Considerations
* Risk in development
* Efforts to minimize Risk
* What's design?
* Up-front design vs. Agile Design
* Hacking vs. Coding
* How to validate design?
* Motivations for TDD
* Let's design a system
* Exercise

Evolutionary Design
* Architecture and design in Agile Development
* How to approach Evolutionary Design
* Engineering vs. Over-engineering
* Key steps towards Evolutionary Design
* Tracer Bullets
* Exercise

Test Driven Development
* How we typically create classes?
* TDD vs. TFD, Test First Coding
* Benefits
* Why TDD?
* Programming by Intention
* Simple code that works
* Types of Tests: Acceptance Testing, Unit Testing
* System functionality and relation to TDD
* Tenets of TDD, TDD Mantra, TDD Concepts
* Practices for Unit Testing
* When and where to write tests, and when not to write?
* Exercise

Testing with Dependencies
* Unit testing… easy and hard…
* Constraints of testing
* What are Mock Objects?
* How to create those?
* How to implement a Mock object?
* Advantages
* Using a Mock
* Mock object frameworks
* Which framework to use?
* Exercise

Refactoring your code
* What is refactoring?
* Why refactor?
* What's needed for refactoring?
* What to refactor?
* Reconcile differences
* Specific Refactoring recommendations
* …Method level, class level, data layer, abstraction
* Tools that help with refactoring
* Exercise

Continuous Integration and TDD
* Why unit test?
* What's hard about it?
* What tests to run and how often?
* Integrate early and frequently
* Continuous Integration
* Tools for Continuous Integration
* Extreme Feedback Devices
* Exercise

Beyond Unit Testing
* Functional and Integration Testing
* Fit and FitNesse
* Integration testing web applications
* Tools for integration testing
* Behavior Driven Design
* Tools for BDD
* Exercise

Agile Design
* Measuring quality of design
* Coupling, Cohesion
* Law of Demeter
* Characteristics of a Poor design
* Design Principles to create maintainable code
* Exercise

Agile Practices
* Agile Software Development
* Devil and the details
* Select Practices
* Beginning Agility
* Feeding Agility
* Delivering What Users Want
* Agile Feedback
* Agile Debugging
* Agile Collaboration
* Epilogue
* Exercise

About the Instructor

Dr. Venkat Subramaniam is an award-winning author, founder of Agile Developer, Inc., creator of agilelearner.com, and an instructional professor at the University of Houston.

He has trained and mentored thousands of software developers in the US, Canada, Europe, and Asia, and is a regularly-invited speaker at several international conferences. Venkat helps his clients effectively apply and succeed with sustainable agile practices on their software projects.

Venkat is a (co)author of multiple technical books, including the 2007 Jolt Productivity award winning book Practices of an Agile Developer. You can find a list of his books at agiledeveloper.com. You may read more about Venkat and Agile Developer, Inc. at http://agiledeveloper.com.