SECC Agile Foundation Certificate Examination Handbook

Versions 2.0

Software Engineering Competence Center

REVISION HISTORY

Version	Date	Remarks
1.0	12/4/2012	Initial version
2.0	3/8/2008	Updated knowledge areas Added questions examples Updated suggested readings section

TABLE OF CONTENTS

INTRODUCTION

The Agile Foundation Certificate (AFC) is a certification for agile practitioners developed by the Software Engineering Competence Center (SECC). The AFC is aimed at anyone developing projects using agile practices such as software developers, testers and technical leads. This certification is also appropriate for anyone who wants a basic understanding of agile philosophy and practices such as project managers and quality managers, software development managers and management consultants. Holders of the AFC will be able to demonstrate to their employer the level of professionalism and knowledge they possess in agile practices.

To obtain the AFC you have to pass an examination. The examination objective is to examine your familiarity with agile fundamental philosophy as well as a collection of best practices associated with mainstream agile frameworks and methodologies. The content of the examination covers the major knowledge areas key for an agile practitioner despite his/her role in the agile team. The examination has 40 multiple choice questions distributed among the following knowledge areas:

- **Fundamentals of Agile Thinking**: covering topics related to basics of agile thinking and philosophy
- **Agile Requirements Engineering**: covering topics related to requirements development and management using user-stories
- **Agile Planning & Tracking**: covering topics related to estimation using story points, release and iteration planning, and agile project tracking techniques
- **Agile Teams**: covering topics related to team roles, structure and dynamics
- **Technical Excellence**: covering techniques related to design, integration and testing of agile projects.

These knowledge areas are described in detail in the following section.

KNOWLEDGE AREAS

The knowledge areas documented in this section of the handbook forms the basis for the Agile Foundation Certification. The details covered in this section aims at helping candidates to understand required knowledge to pass the AFC exam. The description of every knowledge area consists of:

- **Concepts & Terms:** A list of terms and concepts the examinee must be able to recall and understand
- **Overview*:*** General description of the knowledge area
- **Tools & Techniques:** A list of tools and techniques applicable within the knowledge area
- **Detailed Topics:** A description of topics to learn within the knowledge area, with detailed list of tasks to perform in order to demonstrate understanding. Attached to each task (and/or topic) is the required *knowledge level* measured within the exam

Knowledge Levels

The examination measures the knowledge areas across three levels of knowledge:

- **Level 1 – Remember (K1):** The examinee can recognize and recall a concept or a term
- **Level 2 – Understand (K2):** The examinee can explain, compare, classify, or give examples for a concept or a technique
- **Level 3 – Apply (K3):** The examinee can apply a procedure and follow instructions for a technique; or identify the correct application of a concept in a given context

Fundamentals of Agile Thinking	
Concepts & Terms	Lean, waste, Agile manifesto
Overview	The objective of this section is to ensure profound understanding of the pillars of agile thinking. Topics in this section cover Lean concepts, categories of waste, and the agile values and principles.
Tools & Techniques	Value stream mapping
Detailed Topics	
Lean Concepts (K2)	1. Provide examples of how Lean principles applies to software development (K2)
	2. Recall the concept of waste in lean thinking (K1)
	3. Explain, giving examples, the categories of waste in software development (K2)
Agile Manifesto (K2)	1. Recall the four values of the Agile Manifesto (K1)
	2. Explain, giving examples, the difference between agile values and traditional approaches in software development (K2)
	3. Explain the 12 principles of the agile manifesto (K2)
Misconceptions about Agile (K1)	1. Recall the common misconceptions about agile (K1)

Agile Requirements Engineering	
Concepts & Terms	User stories, epics, themes, user proxies
Overview	This section covers the concept and application of incremental requirements development using user stories. Topics include identifying and writing user stories, user stories evolution, user role mapping, and customer collaboration techniques. Examinees should be able to write, detail, split, and prioritize user stories in preparation for planning project releases.
Tools & Techniques	Personas, INVEST, Slicing user stories
Detailed Topics	

User Stories (K3)	1. Recall what user stories are (Cards – Conversation – Confirmation) (K1)
	2. Explain what user roles are and why they are crucial to stories development (K2)
	3. Describe the difference between user stories, epics and themes (K2)
	4. Explain giving examples how non-functional requirements can be described using user stories (K3)
	5. Explain giving examples the INVEST model for developing user stories (K2)
	6. Understand how user stories evolve across the project lifecycle (K2)
	7. Write user stories according to guidelines of the INVEST model (K3)
	8. Translate user stories details into conditions of satisfaction (K3)
	9. Identify points of weaknesses of poorly defined user stories (K2)

Agile Planning & Tracking

Concepts & Terms	Story points, relative estimation, release, iteration, time-boxing, backlog, velocity, burn-chart, retrospective
Overview	This section covers feature-based and adaptive planning and tracking for agile projects. It covers relative size estimation, user stories prioritization, release planning and tracking techniques as well as iteration planning and tracking techniques. The examinee should be able to apply techniques for planning and tracking releases and iterations, like story estimation, velocity estimation, burn-charts interpretation, and velocity tracking, to make sound management decisions about project status and take corrective actions.
Tools & Techniques	Planning poker, release & iteration planning, burn-charts, daily standup meeting, iteration planning meeting, iteration review meeting, retrospectives
Detailed Topics	
Estimation (K3)	1. Recall what relative estimation is (K1)
	2. Recall what story points are (K1)
	3. Explain the attributes which impact the size of a user story (K2)
	4. Describe the Planning Poker estimation technique (K2)
	5. Estimate the relative size of user stories in story points using the Planning Poker technique (K3)

Agile Project Management (K3)	1. Recall the definitions of release and iteration/sprint (K1)
	2. Recall the definitions of Product Backlog and Iteration Backlog (K1)
	3. Explain the concept of time-boxing and why it is crucial to agile teams (K1)
	4. Recall the concept of Done definition for agile teams (K1)
	5. Understand what constitutes a good Done definition on user story, iteration and release level (K2)
	6. Recall the definition of velocity (K1)
	7. Understand how velocity is used in release planning (K2)
	8. Describe how release planning is conducted (K2)
	9. Explain different scenarios for velocity estimation (K2)
	10. Describe factors for prioritizing user stories (K2)
	11. Describe factors for selecting an appropriate iteration length for an agile team (K2)
	12. Apply velocity estimation methods to estimate velocity when historical data is available (K3)
	13. Determine Release delivery dates based on velocity estimation and inputs related to schedule and scope constraints (K3)
	14. Explain the difference between velocity-driven iteration planning and commitment-driven iteration planning (K2)
	15. Describe the steps of commitment driven iteration planning (K2)
	16. Apply the commitment driven iteration planning steps to estimate the iteration's velocity (K3)
Agile Project Tracking (K3)	1. Explain the importance of continuous feedback in agile project management (K2)
	2. Understand the Agile planning and tracking cycle on release and iteration level (K2)
	3. Recall the concept of burn charts as a tool for tracking agile projects (K1)
	4. Explain the difference between release and iteration burn charts (K2)
	5. Explain the difference between burn-up and burn-down charts (K2)
	6. Use burn-charts to draw conclusions about progress of releases and iterations (K3)
	7. Measure and analyze the team performance on iteration and release level and take adaptive actions accordingly (K3)

Agile Meetings (K2)	1. Recall various types of agile meetings (Iteration planning, daily standup, iteration review, retrospectives) (K1)
	2. Explain the scope, frequency, typical duration and required participants for each meeting type (K2)

Agile Teams

Concepts & Terms	Self-organizing, cross-functional, unified vision, Agile roles and responsibilities
Overview	This section covers topics related to team structure and dynamics.
Tools & Techniques	Co-located teams, team retrospectives
Detailed Topics	
Self-Organizing Teams (K2)	1. Explain the difference between self-organizing and micro-managed teams (K2)
	2. Explain, giving examples, rules of order in agile teams (K2)
	3. Recall the responsibilities of Scrum team roles (Scrum master, product owner, and team) (K1)
	4. Recognize the importance of regular retrospectives and reviews of team effectiveness and efficiency (K1)
	5. Explain the impact of motivation on team performance (K2)
	6. Explain the impact of support and trust on team motivation and morale (K2)
	7. Recognize that agile teams are cross-functional (K1)
	8. Explain, giving examples, the impact of being cross-functional on team performance (K2)
Collaboration Techniques (K1)	1. Recognize the importance of face-to-face communication over other communication means (K1)
	2. Recognize the impact of collocated teams on team performance (K1)

Technical Excellence

Key Concepts and Terms	Evolutionary design, up-front design, technical debt, software refactoring, spikes, continuous integration (CI), Test-Driven Development (TDD), pair programming

Overview	This section covers techniques of how to build high quality and robust software products in environments which keep changing requirements and priorities. The examinee should be able to apply evolutionary design techniques to balance early and continuous delivery on one side with potential architecture and design risks on another side. Also, it covers an overview of the most prominent technical practices of agile software development.
Tools & Techniques	Continuous integration, Pair programming, Test-Driven Development, Evolutionary design, Refactoring
Detailed Topics	
Evolutionary Design (K2)	1. Explain the spectrum of formality for different kinds of software projects (K1)
	2. Explain the idea of evolutionary design (K1)
	3. Differentiate between the upfront design compared to evolutionary design (K2)
Technical Debt & Refactoring (K2)	1. Explain the meaning of technical debt (K1)
	2. Give reasons why technical debt should be removed as soon as possible (K2)
	3. Recall what is meant by refactoring (K1)
	4. Explain how refactoring improves design and reduces technical debt (K2)
	5. Describe alternatives for planning refactoring effort (K1)
Technical Practices (K2)	1. Describe with examples what is meant by spikes (K1)
	2. Explain why spikes should be time-boxed (K2)
	3. Recall what is meant by continuous integration (K1)
	4. Explain the basic benefits of continuous integration (K2)
	5. Differentiate between continuous integration and big-bang integration (K2)
	6. Recall what is meant by test driven development (K1)
	7. Explain the basic flow of TDD (K2)
	8. Recall what is meant by pair programming (K1)

EXAM PHILOSOPHY AND RULES

- The exam has 40 questions, in multiple choice formats
- Each question in the exam has 4 possible answers, labeled (a), (b), (c), and (d). Only one answer is correct
- The examination time is 60 minutes total. No breaks are allowed
- The exam is offered in English
- A correct answer to a question in the exam scores at one point (up to a total of 40 points.) The published score is then scaled to a 100% (40 points = 100%)
- A score of 70% is required to pass the exam

Examination Topics Distribution

The following is a rough distribution of exam topics and levels of knowledge:

		K1	K2	K3	Total
		35%	45%	20%	**100%**
Fundamentals of Agile Thinking	12.5%	3	2	0	**5**
Agile Requirements Engineering	25%	2	4	4	**10**
Agile Planning & Tracking	35%	4	6	4	**14**
Agile Teams	12.5%	2	3	0	**5**
Technical Excellence	15%	3	3	0	**6**
Total	100%	14	18	8	**40**

SAMPLE QUESTIONS

1. A user story is finished when:
   a) The conditions of satisfaction are achieved
   b) Design, coding, unit testing and testing tasks of this story are finished
   c) The team Done definition criteria are satisfied
   d) The Product Owner reviews it in the iteration review meeting

2. The product owner is responsible of which of the following activities:
   1. Represent the Customer/users community
   2. Prioritize features according to business value
   3. Ensure that the team is fully functional and productive
   4. Make scope/schedule tradeoff decisions
   5. Protects the team from external conflicts and interferences

   a) 1, 2, 3, 4
   b) 1, 2, 4
   c) 2, 4, 5
   d) All the above

3. The implementation details of a user story are discovered:
   a) During the story writing workshops with the Product Owner
   b) During the iteration review meeting with Product Owner
   c) As the team progresses towards the end of the release
   d) During the iteration where this user story is developed

4. One of the basic benefits on using continuous integration is:
   a) The elimination of integration bugs
   b) Ensuring release is free of bugs
   c) Early detection of integration problems
   d) Ensuring baselines are correctly taken

5. The primary role of the Scrum Master in Planning Poker estimation meeting is to:
   a) Facilitate the discussions among the team members and make sure they are following the steps of planning poker properly
   b) Participate in the estimation and challenge the team estimates to make sure they are estimating correctly
   c) Make sure that everyone is estimating by himself/herself without being influenced by others
   d) Make sure that the Product Owner estimates don't influence the team estimates

6. In Agile software development, the best measure of progress is:
   a) Percentage of resolved bugs
   b) Working software
   c) Phase completion
   d) Release burn charts

7. The purpose of the daily standup meeting is:
   a) Track the remaining effort of the daily team tasks
   b) Analyze the daily issues as soon as they emerge
   c) Track the team's daily progress and highlight any pending issues
   d) Track the team's daily progress and solve any pending issues

8. An iteration is over when:
   a) The iteration tasks are completed
   b) The stories are completed according to the Done definition
   c) The timebox expires
   d) The iteration review meeting and retrospective are conducted

SUGGESTED READINGS

Books:

- M. Poppendieck and T. Poppendieck. *Implementing Lean Software Development from Concept to Cash*, Addison-Wesley Professional, 2006.
- M. Cohn. *User Stories Applied: For Agile Software Development.* Addison-Wesley Professional, 2006.
- M. Cohn. *Agile Estimating and Planning.* Prentice Hall, 2005.
- Henrik Kniberg. *Scrum and XP from the Trenches.* InfoQ, 2007.

Articles:

- K. Schwaber and J. Sutherland, "*The Scrum Guide*", Scrum.org, July 2011. (Internet: http://www.scrum.org/storage/scrumguides/Scrum Guide - 2011.pdf)
- M. Lacey. "*How Do We Know When We Are Done?*", Sept. 2008. (Internet: http://www.scrumalliance.org/articles/107-how-do-we-know-when-we-are-done)
- "*Agile Project Management*", ccpace, 2003. (Internet: http://www.ccpace.com/Resources/documents/AgileProjectManagement.pdf)
- M. Fowler. "*Continuous Integration*", May, 2006. (Internet: http://martinfowler.com/articles/continuousIntegration.html)
- M. Fowler. "*Is Design Dead?*", May, 2004. (Internet: http://www.martinfowler.com/articles/designDead.html)

Internet resources:

- *Manifesto for Agile Software Development*. (http://agilemanifesto.org/)
- R. Jeffries. "*XProgramming.com: An Agile Software Development Resource*" (http://www.xprogramming.com)
- Rally Software. " *Definition of Done Toolkit* " (http://www.rallydev.com/toolkits/definition-done-toolkit)
- Mountain Goat Software. (http://www.mountaingoatsoftware.com/)
  M. Fowler. "*Refactoring Home Page*" (http://www.refactoring.com/)

ANSWERS TO SAMPLE QUESTIONS

Answers:

Question	Answer
1	c
2	b
3	d
4	c
5	a
6	b
7	c
8	c