



Agile Software Development of Embedded Systems

Version: 1.0
Date: 2006.12.15
Pages: 33

Authors

Youri Metchev
Boyan Angelov

Status

Final

Confidentiality

Public

Agile Deliverable D.2.11

Combining Agile Methodologies and CMMI/ISO 9001:2000 Practices in Micro, Small and Middle-sized Enterprises (SME), Developing Software.

Abstract

This report presents the results of analysis of the applicability of Agile software development methodologies in companies working with CMMI/ISO 9001:2000 certified working processes. It presents the basics of several Agile methodologies and the requirements of CMMI/ISO standards and gives an overview of the applicability of the Agile practices in the process areas addressed by the standards.

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

Revision History

Date	Version	Description	Author
18.05.2006	0.1	First version of the technical report	Youri Metchev, Boyan Angelov
19.09.2006	0.2	Expand all TBE markings in the text	Youri Metchev
12.12.2006	0.3	Expand CMMI/AGILE mapping to Level 3 of CMMI	Youri Metchev
15.12.2006	1.0	Layout fixes	Boyan Angelov

Review List

Date	Version	Status / Comment	Author

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

1. Objectives

This document presents a guidance for implementing a combination of CMMI and Agile practices which will:

- streamline the process of software development in a typical SME
- help achieving official CMMI / Agile certification in the lightest possible way

2. Scope

A mapping is presented between specific and general practices of CMMI Level 2 and Level 3 and practices from the following Agile methodologies:

- Scrum
- XP
- Crystal (Crystal clear)

3. Audience

SME's, CEOs CTOs Development managers, Team leaders

4. Agile methodologies and practices

4.1. SCRUM

The Scrum process was first applied to software by **Ken Schwaber** and **Jeff Sutherland**. It has been most thoroughly documented in the book **Agile Software Development with Scrum** by **Ken Schwaber** and **Mike Beedle**.

4.1.1 Scrum overview

Scrum is an agile process for developing software. With Scrum, projects progress via a series of month-long iterations called **sprints**.

Scrum is ideally suited for projects with rapidly changing or highly emergent requirements. The work to be done on a Scrum project is listed in the **Product Backlog**, which is a list of all desired changes to the product. At the start of each Sprint a **Sprint Planning Meeting** is held during which the **Product Owner** prioritizes the Product Backlog and the **Scrum Team** selects the tasks they can complete during the coming Sprint. These tasks are then moved from the Product Backlog to the **Sprint Backlog**.

During the Sprint the team stays on track by holding **brief daily meetings**.

At the end of each Sprint the team demonstrates the completed functionality at a **Sprint Review Meeting**.

4.1.2 Scrum practices

The Scrum Master

The Scrum Master is responsible for making sure a Scrum team lives by the values and practices of Scrum. The Scrum Master protects the team by making sure they do not overcommit themselves to what they can achieve during a sprint.

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

The Scrum Master facilitates the **Daily Scrum Meeting** and becomes responsible for removing any obstacles that are brought up by the team during those meetings.

The Scrum Master role is typically filled by a Project Manager or a Technical Team Leader.

Product backlog

The Product Backlog is the master list of all functionality desired in the product. When a project is initiated there is no comprehensive, time-consuming effort to write down all foreseeable tasks. Typically, a project writes down everything obvious, which is almost always more than enough for a first sprint. The Product Backlog is then allowed to grow and change as more is learned about the product and its customers.

During the **Sprint Planning Meeting** the Product Owner prioritizes the items in the Product Backlog and describes them to the team. The team then determines which items they can complete during the coming Sprint. The team then moves items from the Product Backlog to the **Sprint Backlog**. In doing they expand each Product Backlog item into one or more **Sprint Backlog** tasks so they can more effectively share work during the Sprint. Conceptually, the team starts at the top of the prioritized Product Backlog list and draws a line after the lowest of the high priority tasks they feel they can complete. In practice it is not unusual to see a team select, for example, the top five items and then two items from lower on the list but that are associated with the initial five.

Backlog items can be technical tasks (“Refactor the Login class to throw an exception”) or more user-centric (“Allow undo on the setup screen”). A very interesting prospect is expressing Scrum backlog items in the form of Extreme Programming’s **User Stories**.

Scrum teams

A typical Scrum team is 6-10 people but it could be scaled to over 800 people. Scrum teams do not include any of the traditional software engineering roles such as Programmer, Designer, Tester, or Architect. Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint.

Daily Scrum meetings

During the month-long sprints the team holds daily meetings (“the daily Scrum”). Meetings are typically held in the same location and at the same time each day. Ideally the daily Scrums are held in the morning as they help set the context for the coming day’s work.

Each participant in the Daily Scrum is known as either a chicken or a pig, depending on his involvement in the project. The reference is to the old joke that in a ham-and-eggs restaurant the pig is committed while the chicken is only involved. Only pigs are allowed to talk or ask questions during a Daily Scrum meeting.

All pigs (those who have committed to completing the work of the sprint) are required to attend the Daily Scrum. Anyone else (for example, a departmental VP, a salesman, or a developer from another project) is allowed to attend but as a chicken he is there only to listen. This makes the Daily Scrums an excellent way for a Scrum team to

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

disseminate status information—if you're interested in hearing where things are at, attend that day's meeting.

The Daily Scrum is not used as a problem-solving or issue resolution meeting. Issues that are raised are taken offline and usually dealt with by the relevant sub-group immediately after the Daily Scrum. During the Daily Scrum each team member provides answers to the following three questions:

- What did you do yesterday?
- What will you do today?
- Are there any impediments in your way?

The Daily Scrum is not a status update meeting in which a boss is collecting information about who is behind schedule. Rather, it is a meeting in which team members make commitments to each other.

Any impediments that are raised become the Scrum Master's responsibility to resolve as quickly as possible.

In cases where the Scrum Master cannot remove these impediments directly himself (e.g., usually the more technical issues) he still takes responsibility for making sure someone on the team does quickly resolve the issue.

Sprint planning meeting

The Sprint Planning Meeting is attended by the **Product Owner**, **Scrum Master**, the entire **Scrum Team**, and any interested and appropriate management or customer representatives.

During the Sprint Planning Meeting the **Product Owner** describes the highest priority features to the team. The team asks enough questions during this meeting so that they can go off after the meeting and determine which tasks they will move from the **Product Backlog** to the **Sprint Backlog**.

Collectively the **Scrum Team** and the **Product Owner** define a "Sprint Goal," which is a short description of what the sprint will attempt to achieve. The success of the sprint will later be assessed during the **Sprint Review Meeting** against the Sprint Goal, rather than against each specific item selected from the **Product Backlog**.

After the Sprint Planning Meeting, the **Scrum Team** meets separately to discuss what they heard and decide how much they can commit to during the coming Sprint. In some cases there will be negotiation with the **Product Owner** but it will always be up to the team to determine how much they can commit to completing.

Sprint

Sprints last for thirty calendar days. Adjustments to the duration can be made after team members have more experience with Scrum.

Every Sprint has a defined Sprint Goal. The team has the authority to change the functionality of the Sprint as long as it meets its Sprint Goal.

The team has two mandatory accountabilities during a Sprint:

- Daily sprint meetings
- Sprint backlog

During the sprint all work that is performed is measured and empirically controlled.

The team is required to deliver a product increment at the end of the sprint

Sprint review

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

At the end of each sprint a Sprint Review Meeting is held. During this meeting the **Scrum Team** shows what they accomplished during the sprint. Typically this takes the form of a demo of the new features.

The Sprint Review Meeting is intentionally kept very informal, typically with rules forbidding the use of PowerPoint slides and allowing no more than two hours of preparation time for the meeting. A Sprint Review Meeting should not become a distraction or significant detour for the team; rather, it should be a natural result of the Sprint.

Participants in the Sprint Review Meeting typically include the **Product Owner**, management, customers, and engineers from other projects.

During the Sprint Review Meeting the project is assessed against the Sprint Goal determined during the **Sprint Planning Meeting**. Ideally the team has completed each task planned for the sprint but it is more important that they achieve the overall goal of the sprint.

4.2 Crystal

Crystal is a family of human-powered and adaptive, ultralight, “shrink-to-fit” software development methodologies. “Human-powered” means that the focus is on achieving project success through enhancing the work of the people involved (other methodologies might be process-centric, or architecture-centric, or tool-centric, but Crystal is people-centric). “Ultralight” means that for whatever the project size and priorities, a Crystal-family methodology for the project will work to reduce the paperwork, overhead and bureaucracy to the least that is practical for the parameters of that project. “Shrink-to-fit” means that you start with something possibly small enough, and work to make it smaller and better fitting. Crystal is non-jealous, meaning that a Crystal methodology permits substitution of similar elements from other methodologies.

Crystal collects together self-adapting family of “shrink-to-fit,” human-powered software development methodologies based on these understandings:

- Every project needs a slightly different set of policies and conventions, or methodology.
- The workings of the project are sensitive to people issues, and improve as the people issues improve, individuals get better, and their teamwork gets better.
- Better communications and frequent deliveries communication reduce the need for intermediate work products.

4.2.1 Crystal clear overview

Crystal methodologies are characterized along 2 dimensions:

- Size
- Criticality

Different Crystal methodologies are applicable to different kind of projects.

Crystal clear is methodology for collocated teams of 8 or fewer people.

4.2.2 Crystal clear Strategies

Exploratory 360⁰

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

At the start of the project (during chartering activity) the feasibility of the project is evaluated by considering the following:

- Business value (The scope of the system)
- Requirements (Low precision use cases)
- Domain model
- Technology plans
- Project plan (coarse grained project plan is created)
- Team makeup
- Methodology/working conventions

Early victory

The early victory is the the first piece of of visibly running tested code (walking skeleton). The preferred approach is “Easiest thing first, Hardest second”.

Walking skeleton

Tiny implementation of the system that performs a small end-to-end function.

Incremental rearchitecture

The system architecture evolves from the “Walking skeleton” and handles changes in technology and business requirements over time.

Information radiators

A display posted in a place where people can see it as they work or walk by. It shows status information such as:

- The current iteration’s use cases or stories
- Current work assignments
- Number of tests written/passed
- Number of use cases/stories delivered
- Status of key servers
- The core of domain model
- The results of the last reflection workshop

4.2.3 Crystal clear Techniques

Methodology shaping

Trough project interviews a small library of experiences within the organization (strengths, weaknesses) is build. Then during the workshops those experiences are examined and discussed and actions to take advantage of the strengths and how to avoid weaknesses are planned.

Reflection workshop

After each delivery the team pauses for about an hour to reflect on its working conventions. Typically the following is discussed:

- What should be kept
- What are the ongoing problems
- What the team wants to try in the next time period

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

Blitz planning

Planning session in which the executive sponsor of the project, expert user/s and developers take part and layout the project plan and schedule. During such meeting the Planning game (XP technique) can be used. It is called Blitz because it goes fast.

Delphi estimation

Project estimation session held after the initial requirements are collected and clarified. The Project Manager/Coordinator/Leader, expert user and most experienced developers estimate the size of the project, work time, releases, release size.

Daily standup meetings

See Scrum Daily meeting

Essential Interaction design

All activities within the project to define, design and test the system personalities and its interaction contexts. The Personalities of an software system are presented to its end users trough its help, information and speed -for example it can be fast and efficient or friendly and informative.

Process miniature

This technique is used to help introducing new practices, methodologies and processes in the teams by using test projects, small iterations, trials.

Side-by-side programming

Expansion to the Pair Programming practice – 2 people sit close enough to see each others screens but work on their own assignments. Each can ask the other at any moment to look at a piece of code, run a test ot similar.

Burn Charts

Burn charts are used to give visibility into the progress of an project. They resemble earned value charts. On the vertical axis relative work units or percent complete values are marked and on the horizontal axis - calendar time. The planned progress is represented by a line drawn between the origin and the completion point, by estimating the rate at which work can be accomplished or "drop dead" delivery date. After each iteration the relative work completed is marked.

4.3 XP

4.3.1 XP overview

Extreme Programming (XP) is a **software engineering methodology** for the development of software projects. It prescribes a set of day-to-day **practices** for developers and managers; the practices are meant to embody and encourage particular **values**. Proponents believe that the exercise of these practices, which are some software-engineering best practices taken to so-called “extreme” levels, leads to a development process with the qualities prized by the original signatories of the **Agile Manifesto**. This makes Extreme Programming the most prominent of several **agile software development** methodologies used to create software. Agile methodologies rank *adaptability* higher than *predictability*: the adaptability, to changing requirements, ranks higher than the project predictability valued by more traditional

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

methodologies. Some elements of the Extreme Programming methodology are debated.

XP became extremely popular in the late 1990s and early 2000s, seeing adoption in a number of environments radically different from its origins.

The high discipline required by the original practices often went by the wayside, causing certain practices to be deprecated or left undone on individual sites. Agile development practices have not stood still, and XP is still evolving, assimilating more lessons from experiences in the field. In the second edition of *Extreme Programming Explained*, Beck added more values and practices and differentiated between primary and corollary practices.

4.3.2 XP Primary Practices

Sit together

Development is done in an open space big enough for the whole team. This practice does not discourage multi-site development.

Whole team

Development team consists of people having all the skills, knowledge and experience needed for the project to succeed. (cross-functional team) The ideal size of a team is between 12 and 150 people. Many organizations split teams when they cross that threshold.

Informative workspace

The general idea of this practice is that anyone interested in the course of the project and its state should be able to walk into the team space and get a general idea of how the project is going in 15 seconds (see Information radiator pattern).

Energized work

During the day team members should work only as many hours as they can be productive and only as many hours as they can sustain.

Pair programming

Pair programming requires two **software engineers** to participate in a combined development effort at one workstation.

Each member performs the action the other is not currently doing: While one types in **unit tests** the other thinks about the class that will satisfy the test, for example.

The person that is doing the typing is known as the *driver* while the person that is guiding is known as the *navigator*. It is often suggested for the two partners to switch roles at least every half-hour.

Stories

Stories are units of customer-visible functionality. They are used as a basis of planning the software development projects. As soon as a story is defined, an estimation of the development effort necessary to implement it is done. Stories are given a short descriptive name and contain a prose or graphical description of the functionality required.

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

Weekly cycle

Actual work on a project is planned a week at a time. At the beginning of every week a meeting is held. During this meeting:

- progres to date is reviewed
- customers pick a week's worth of stories to be implemented
- Stories are broken into tasks

The work for the week starts with writing the authomated tests that will run when the stories are completed. The rest of the week is spend working on implementing the stories and getting the tests to pass. Preferably when stories are broken to tasks members of the team take responsibilities for separate tasks and estimate them.

Quarterly cycle

This practice describes planing project work a quarter at a time.

During quarterly planning:

- Bottlenecccks (especially those controlled outside the team) are identified
- Theme/themes (aggregation of stories) for the quarter are planned
- A quarter's worth of stories is picked up to address themes
- Focus is held on the big picture

Slack

During iteration planning some minor tasks that can be easily dropped or posponed are identified and included in the iteration plan.

Ten-minute build

In the ideal situation the authomated building the whole system and running all of the tests should take no more than 10 minutes.

Continuous integration

Continuous integration is a **software engineering** term describing a process that completely rebuilds and **tests** an application frequently. Although the underlying concepts existed earlier, **Continuous integration** typically refers to the **Extreme programming** practice.

As originally done in **Extreme programming**, **Continuous integration** is accomplished via a serialized build process. At the completion of a task, the developer (or development pair) takes a build token and runs the build process, including tests. If the tests pass, the task can be committed to the source code repository and release the token.

The more popular form (also known as Automated Continuous Integration) takes the form of a server process or **daemon** that monitors a **version control system** for changes and automatically runs the build process (e.g. a **make** script or **Ant**-style build script) and then runs test scripts (e.g. **JUnit** or **NUnit**). In many cases the build script not only compiles binaries but also generates documentation, website pages, statistics and distribution media (such as Windows **MSI** files or **RPM** files).

Scheduled builds, if done frequently enough, would fall under **Continuous integration**. However, nightly builds are too infrequent to qualify.

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

Continuous integration is frequently associated with **Extreme programming** and other **agile software development** practices, but can be, and has been, adopted by more conventional methodologies too.

The main advantages of continuous integration are:

- Integration problems are detected and fixed continuously - no last minute hiatus before release dates;
- Early warning of broken/incompatible code;
- Immediate unit testing of all changes;
- Constant availability of a "current" build for testing, demo, or release purposes;
- The immediate impact of checking-in incomplete or broken code acts as an incentive to developers to learn to work more incrementally with shorter feedback cycles.

Test-First programming

Test-first programming is a **computer programming** technique that involves writing **test cases** first and then implementing the code necessary to pass the tests. The goal of test-driven development is to achieve rapid feedback and implements the "illustrate the main line" approach to constructing a program. This technique is heavily emphasized in **Extreme Programming**.

Practitioners emphasize that test-driven development is primarily a **method of designing software**, not just a method of testing. The method is also used for removal of **software defects**.

For test-driven development to work, the system must be flexible enough to allow for **automated testing** of code, using test cases that return a simple true or false evaluation of correctness. These properties allow for rapid feedback of correctness and design. Unit testing frameworks (**xUnit**) such as **JUnit**, **NUnit** or **CppUnit** provide a mechanism for managing and running sets of automated test cases.

Test-first programming addresses the following problems:

- Scope creep – avoid putting in code “just in case”
- Coupling and cohesion – design problems are identified early
- Trust is established
- Rhythm is developed

Incremental design

The design of the system is made to be excellent fit for the today's needs of the system. Later when the understanding of the best possible design moves forward, the actual design of the system is GRADUALLY brought back into alignment with the understanding.

4.3.2 XP Corollary practices

Real customer involvement

Real customers of the system under development are made part/members of the team – see Agile customer pattern.

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

Incremental deployment

The system under development is gradually deployed based on agreed chunks of the overall functionality.

Team continuity

Effective teams are kept together across different projects.

Shrinking teams

As during the project course the project team grows in capability its workload is kept constant but its size is gradually reduced. This practice helps to free people to form more teams. On the other side if a team has too few members it is merged with another too-small team.

Root cause analysis

Every time when a defect is found it is not only the defect that is being eliminated but also its cause. The goal is not only that this kind of defect will never occur again, but that the team will never make the same kind of mistake again.

Shared code / (Collective code ownership)

When everyone on a project is pair programming, and pairs rotate frequently everybody gains a working knowledge of the entire codebase. Then anyone of the team can improve any part of the system at any time. This practice goes hand in hand with Continuous integration.

Code and tests

Only code and tests are maintained as permanent artifacts. Other documents are generated from the code and tests as necessary.

Single code base

There is only one code stream. Temporary branches are kept alive for no more than a few hours.

Only one code base is kept at anyone time. If multiple code bases exists for various reasons a plan is developed for reducing them gradually.

Daily deployment

New software is put in production every night. Prerequisites for this practice include:

- Defect rate is low
- Build environment is automated
- Deployment tools are automated
- Highly developed trust within the team and with customers

Negotiated scope contract

Contracts for software development fix time, costs and quality but have option for ongoing negotiation of the precise scope of the system.

Pay-per-use

Customers are charged for every time the system is used. Money are regarded as the "ultimate feedback"

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

5. CMMI

Capability Maturity Model[®] Integration (CMMI) is a process improvement approach, developed by SEI, which provides organizations with the essential elements of effective processes. It is used to guide process improvement across a project, a division, or an entire organization. CMMI helps integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes.

5.1 CMMI Overview

CMMI is a Capability maturity model. Capability maturity models (CMMs) focus on improving processes in an organization. They contain the essential elements of effective processes for one or more disciplines and describe an evolutionary improvement path from ad hoc, immature processes to disciplined, mature processes with improved quality and effectiveness.

CMMI covers several disciplines (bodies of knowledge):

- software engineering (CMMI-SW)
- systems + software engineering (CMMI-SE/SW)
- systems + software engineering + integrated product and process development (CMMI-SE/SW/IPPD)
- systems + software engineering + integrated product and process development + supplier sourcing (CMMI-SE/SW/IPPD/SS)
- systems + software engineering + supplier sourcing (CMMI-SE/SW/SS)

There are 2 representations of CMMI:

- Continuous
- Staged

The continuous representation offers a flexible approach to process improvement. An organization may choose to improve the performance of a single process-related trouble spot, or it can work on several areas that are closely aligned to the organization's business objectives. The continuous representation also allows an organization to improve different processes at different rates. There are some limitations on an organization's choices because of the dependencies among some process areas.

Capability levels are used to measure the improvement path through each process area from an unperformed process to an optimizing process.

The staged representation offers a systematic, structured way to approach process improvement one step at a time. Achieving each stage ensures that an adequate improvement has been laid as a foundation for the next stage.

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

Process areas are organized by maturity levels that take much of the guess work out of process improvement. The staged representation prescribes the order for implementing each process area according to maturity levels, which define the improvement path for an organization from the initial level to the optimizing level. Achieving each maturity level ensures that an adequate improvement foundation has been laid for the next maturity level and allows for lasting, incremental improvement.

The components of CMMI are:

- **Required Components** - describe what an organization must achieve to satisfy a process area. This achievement must be visibly implemented in an organization's processes. The required components in CMMI are the specific and generic goals. Goal satisfaction is used in appraisals as the basis for deciding if a process area has been achieved and satisfied.
- **Expected Components** - describe what an organization will typically implement to achieve a required component. Expected components guide those who implement improvements or perform appraisals. Expected components include the specific and generic practices. Before goals can be considered satisfied, either the practices as described or acceptable alternatives to them must be presented in the planned and implemented processes of the organization.
- **Informative Components** – Informative components provide details that help organizations get started in thinking about how to approach the required and expected components. Subpractices, typical work products, discipline amplifications, generic practice elaborations, goal and practice titles, goal and practice notes, and references are all informative model components.

5.2 CMMI Practices

5.2.1 Specific Practices

Specific practices are expected model components that are considered important guidance for achieving specific goals. Most specific practices are exactly the same in both representations. The exceptions are a few practices that are viewed differently when using one or the other model representation.

In the staged representation, all specific practices are viewed equally. Capability levels of specific practices are not recognized (they appear in the numbering of the specific practice but this information is considered irrelevant in the staged representation); therefore, base and advanced practices are not recognized concepts.

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

In the continuous representation, each specific practice is assigned a capability level. Practices with a capability level of one are called *base practices*. Those with capability levels greater than one are called *advanced practices*.

5.2.2 General Practices

Generic practices are the same in all process areas within a maturity level in Staged representation. When successfully implemented into the everyday activities of the organization they satisfy the generic goals which has the effect of institutionalising the specific practices.

6 Mapping SW-CMMI vS agile practices

For each CMMI process area, identify the agile practices that can be used to perform the activities of this area.

CMMI ML2 PA	Mapping level	Agile practices
Requirements Management	largely satisfies	Product backlog Product owner Sprint backlog Customer available Exploratory 360 Stories Negotiated scope Contract Bi-direction traceability is not covered in Agile
<i>SP 1.1 Obtain an understanding of requirements</i>		
<i>Sp 1.2 Obtain Commitment to requirements</i>		
<i>SP 1.3 Manage requirement changes</i>		
<i>SP 1.4 Maintain bidirectional traceability of requirements</i>		
<i>1.5 Identify inconsistencies between project work and requirements</i>		
Project planning	largely satisfies	Sprint planning meeting Exploratory 360 Blitz planning Delphi estimation Weekly planning Quarterly planning Methodology shaping

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

Project Monitoring and Control	largely satisfies	Sprint review meeting Daily Scrum Scrum master Reflection workshop Customer available Daily standup meetings Burn Charts
Supplier Agreement management	does not satisfy	
Measurement and analysis	largely satisfies	Sprint review meeting Daily Scrum Scrum master Reflection workshop Customer available Daily standup meetings Burn Charts
Process and Product Quality Assurance	does not satisfy	
Configuration management	Somewhat satisfies	Sprint planning Blitz planning Stories Weekly cycle Quarterly cycle Sprint review meeting Reflection workshop

7. ISO 9001:2000

The ISO 9000 family of international quality management standards and guidelines has earned a global reputation as the basis for establishing quality management systems.

ISO 9001:2000 specifies requirements for a quality management system for any organization that needs to demonstrate its ability to consistently provide product that meets customer and applicable regulatory requirements and aims to enhance customer satisfaction. The standard is used for certification/registration and contractual purposes by organizations seeking recognition of their quality management system.

There are five sections in the standard that specify activities that need to be considered when an organization implements a quality management system (QMS). The organization will describe the activities it uses to supply its products and may exclude the parts of the Product Realization section that are not applicable to its operations. The requirements in the other four sections - *Quality management system*, *Management responsibility*, *Resource management* and *Measurement, analysis and improvement* - apply to all organizations and the organization will demonstrate how it applies them in its quality manual or other documentation.

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

Together, the five sections of ISO 9001:2000 define what the organization should do consistently to provide products that meet customer and applicable statutory or regulatory requirements. In addition, it will seek to enhance customer satisfaction by improving its quality management system.

Additional standards exist, which can be used as guidance for implementation of the ISO 9001:2000 into the practice of the software development/IT intensive organizations:

- ISO 10006 for project management
- ISO 10007 for configuration management
- ISO 10012 for measurement systems
- ISO 10013 for quality documentation
- ISO/TR 10014 for managing the economics of quality
- ISO 10015 for training
- ISO/TS 16949 for automotive suppliers
- ISO 19011 for auditing

8. Mapping ISO 9001:2000 vs. Agile practices

ISO 9001 clause	Mapping level	Agile practices/patterns
Quality management system	Does not satisfy	
Management responsibility	Does not satisfy	
Resource management	Largely satisfies	Scrum master Scrum team Exploratory 360 ⁰ Methodology shaping Side by side programming Pair programming Sprint planning Daily meetings Sprint review meeting Process miniature Team continuity Shrinking teams

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

		<p>Root cause analysis Whole team Information radiators Sit together Informative workspace Ten-minute build</p>
<p>Product realization</p>	<p>Largely satisfies</p>	<p>Scrum plan – vision + product backlog Sprint planning meeting Sprint backlog Exploratory 360 Blitz planning Delphi estimation Weekly cycle Quarterly cycle Slack Product backlog Product owner prioritizing requirements Daily stand-up meetings Essential Interaction design Stories Real customer involvement Negotiated scope contract Sprint review meeting at the end of the Sprint Reflection workshop Weekly cycle Quarterly cycle Slack Incremental deployment Negotiated scope contract Pay-per-use Acceptance tests written by users Customer available Incremental re-architecture Methodology shaping Sprint review meeting Reflection workshop Acceptance testing Code and tests Continuous integration Test – first programming Incremental design Real customer involvement</p>

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

		Incremental deployment Daily deployment
Measurements Analysis and Improvement	Somewhat satisfies	Product Owner prioritizes the Product Backlog sprints Sprint review meeting Sprint Planning Meeting Early victory Walking skeleton Reflection workshop Weekly cycle Quarterly cycle Real customer involvement Incremental deployment Negotiated scope contract Pay-per-use Scrum master Daily scrum meeting Daily meeting Root cause analysis

9. Certification

9.1 CMMI Certification - SCAMPI

The Standard CMMI[®] Appraisal Method for Process Improvement (SCAMPISM) provides quality ratings using Capability Maturity Model[®] Integration (CMMI) models.

The SCAMPI family of appraisals includes Class A, B, and C appraisal methods.

SCAMPI A is the most rigorous method and the only method that can result in a rating.

SCAMPI B provides options in model scope, but the characterization of practices is fixed to one scale and is performed on implemented practices.

SCAMPI C provides a wide range of options, including characterization of planned approaches to process implementation according to a scale defined by the user.

Using SCAMPI B, every practice in the appraisal scope is characterized on a three point scale indicating the risk of CMMI goal satisfaction if the observed practices were deployed across the organizational unit. Model scope is not limited to the Process Areas but could include sets of related practices.

SCAMPI C can be scoped at any level of granularity and the scale can be tailored to the appraisal objectives, which might include the fidelity of observed practices to

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

model/goal achievement or the return on investment to the organization from implementing practices.

Reliability, rigor, and cost might go down from A to B to C, but risk might go up.

Characteristics of Appraisal Classes

Characteristic	Class A	Class B	Class C
Amount of objective evidence	High	Medium	Low
Ratings generated	Yes	No	No
Resource needs	High	Medium	Low
Team size	Large	Medium	Small

9.2 IT Mark

ESI Center Alliance has launched worldwide the I.T. Mark Certification, aimed at certifying the quality and maturity of the processes in SMEs that develop and maintain Information Technology (IT) Systems.

This new **certification scheme** was launched due to the general perception among IT SMEs that the Quality models existing in the Sector are designed for large organizations, and therefore are not properly adapted to SMEs, since the timelines, cost and implementation requirements are high. Therefore, there is a need for a specific model designed with and for Information Technology SMEs.

In general, SMEs need to improve their Software & Systems processes, but also other Management processes that the usual models normally do not take into account. This is why I.T. Mark assesses and certifies the quality of SMEs in three main areas: one related to overall **Business Management** (strategic, commercial, financial, marketing, etc); another on **Information Security Management**; and the third one, specifically related to the Maturity of their **Software & Systems Processes**. In matters relating to Business Management the reference used is the **10-Squared model**, which was developed to assess applications for Venture Capital. From the Information Security Management point of view, the reference model is the **ISO 17799**, whereas for Software & Systems a lightweight version of **CMMI®** is used, a standard globally acknowledged by the IT world with which ESI has broad experience in providing services to enterprises in all five continents.

I.T. Mark distinguishes three grades depending on the Maturity shown by the SME's processes. The Service also defines Improvement Actions, and seeks to launch SMEs into a process of Continuous Improvement that is fully compatible with the CMMI

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

model. The certification is valid for two years, in order to make sure of the maintenance of the accredited level for each company.

9.3 Agile certification

There are ongoing efforts to create some models for certification of Agile processes. These are still in draft phase.

5.3.1. IT Mark Agile

This is a variation of the IT Mark service in which Agile evaluation is added/replaces the CMMI based Software and systems processes maturity assessment.

9.4 ISO 9001:2000

Organization that has implemented an ISO 9001:2000 compliant QMS applies for independent certification of its QMS before third party certification body. The certification body on its turn is accredited by an accreditation body.

ISO 9001:2000 QMS certificates have a validity period of 3 years and the organization undergoes periodical surveillance audits done by the certification body. To renew the certificate after the validity period has expired a recertification audit is needed.

10. Conclusions

It is acknowledged that there are different priorities for software development companies of different sizes:

- SME (micro, small and medium enterprises):
 - Risk management
 - Task estimation (Project Planning)
 - Productivity, technology, rework
- Bigger companies
 - Consistency across teams
 - Team communication
 - Process adherence

CMMI and Agile are in fact clusters of useful practices that work.

CMMI is for PROCESS IMPROVEMENT (by presenting number of useful practices which organizations can implement in their everyday work) and not for CERTIFICATION. It is possible that as a result from a CMMI based process improvement initiative an organization will implement Practices from one or more Agile methodologies.

According to its own priorities each organization can choose some PAs from CMMI Continuous Representation and to start improving its processes within those PA as agile as it can/need choosing more management oriented Agile methods (Scrum, Crystal).

Technical report on Combining Agile Methodologies and CMMI/ISO 9001:2000 practices in Micro, Small and Middle-sized Enterprises (SME), developing software.

Author Youri Metchev, Boyan Angelov

Version 1.0

Date 15.12.2006

If (formal) certification is needed – organizations can choose Agile, CMMI (Scampi), IT Mark etc.

If while implementing Agile methodologies an ISO 9001 certification is needed, the typical roadmap to implementation of ISO 9001:2000 compliant QMS is:

- Identification of the organization's needs
- Identifications of what the external parties need from the organization
- Applying the ISO 9000 family of standards and more specifically ISO 9001:2000 in the organization's QMS
- Obtain guidance from additional standards as needed
- Determination of the gaps between the organization's QMS and ISO 9001
- Definition of organization's processes needed to supply products to customers
- Implementation of a plan to close the gaps and to implement the defined processes
- Undergoing periodical internal assessments
- Undergoing independent certification audit from a third-party certification body
- Continuous improvement of the organization's processes and QMS as per ISO 9004:2000 standard

As seen from the mapping between ISO 9001 clauses and Agile practices an ISO 9001 QMS can be build on the basis of Agile practices and methodologies without much overhead.

Process Areas, Specific Goals and Practices		Mapping level	Agile practices
CMMI ML2			
Requirements Management		largely satisfies	
SG 1	<i>SP 1.1 Obtain an understanding of requirements</i>	largely satisfies	Product backlog Product owner Sprint backlog Customer available Stories Negotiated scope Contract
	<i>Sp 1.2 Obtain Commitment to requirements</i>		
	<i>SP 1.3 Manage requirement changes</i>		
	<i>SP 1.4 Maintain bidirectional traceability of requirements</i>	does not satisfy	not covered in Agile
	<i>1.5 Identify inconsistencies between project work and requirements</i>		Product backlog Product owner Sprint backlog Customer available Negotiated scope Contract Sprint planning meeting
Project planning		largely satisfies	
SG 1	<i>SP 1.1 Estimate the scope of the project</i>	Largely satisfies, some additional artefacts might be needed	Sprint planning meeting Exploratory 360 Blitz planning Delphi estimation Weekly planning Quarterly planning Methodology shaping Product backlog Product owner Estimating backlog effort Scrum team Sprint planning meeting
	<i>SP 1.2 Establish estimates of workproduct and task attributes</i>		
	<i>SP 1.3 Define project lifecycle</i>		
	<i>SP 1.4 Determine estimates of effort and cost</i>		
SG 2	<i>SP 2.1 Establish the budget and schedule</i>	Largely satisfies, some additional artefacts might be needed	Daily scrum Sprint planning meeting Exploratory 360 Blitz planning Weekly planning Quarterly planning Product owner
	<i>SP 2.2 Identify project risks</i>		
	<i>SP 2.3 Plan for data management</i>		

	<i>SP 2.4 Plan for project resources</i>		
	<i>SP 2.5 Plan for needed knowledge and skills</i>		
	<i>SP 2.6 Plan stakeholder involvement</i>		
	<i>SP 2.7 Establish the project plan</i>		
SG 3	<i>SP 3.1 review plans that that affect the project</i>	Largely satisfies	Daily scrum Sprint planning meeting Blitz planning Weekly planning Quarterly planning Product owner Estimating backlog effort Scrum team
	<i>SP 3.2 Reconcile work and resource levels</i>		
	<i>SP 3.3 Obtain Plan commitment</i>		
Project Monitoring and Control		largely satisfies	
SG 1	<i>SP 1.1 Monitor project planning parameters</i>	Largely satisfies, some additional artefacts might be needed	Sprint review meeting Daily Scrum Scrum master Reflection workshop Customer available Daily stand-up meetings Burn Charts
	<i>SP 1.2 Monitor commitments</i>		
	<i>SP 1.3 Monitor project risks</i>		
	<i>SP 1.4 Monitor data management</i>		
	<i>SP 1.5 Monitor stakeholder involvement</i>		
	<i>SP 1.6 Conduct progress reviews</i>		
	<i>SP 1.7 Conduct milestone reviews</i>		
SG 2	<i>SP 2.1 Analyse Issues</i>		
	<i>SP 2.2 Take corrective actions</i>		
	<i>SP 2.3 Manage corrective actions</i>		
Supplier Agreement management		Does not satisfy	
SG 1	<i>SP 1.1 Determine Acquisition Type</i>	There are no Agile practices that can be mapped to the practices within this CMMI process area	
	<i>SP 1.2 Select Suppliers</i>		
	<i>SP 1.3 Establish Supplier Agreements</i>		
SG 2	<i>SP 2.1 Review COTS Products</i>		
	<i>SP 2.2 Execute the Supplier Agreement</i>		
	<i>SP 2.3 Accept the Acquired Product</i>		
	<i>SP 2.4 Transition Products</i>		
Measurement and analysis		Somehow satisfies	

SG 1	<i>SP 1.1 Establish measurement Objectives</i>	Some of the Agile practices can be used “as it is” (Burn charts support effort measurement) but additional activities and artefacts must be presented to satisfy the goals of this CMMI PA	Sprint review meeting Daily Scrum\ stand-up meetings Scrum master Reflection workshop Burn Charts Information radiators\Informative workspace
	<i>SP 1.2 Specify measures</i>		
	<i>SP 1.3 Specify data collection and Storage procedures</i>		
	<i>SP 1.4 Specify analysis procedures</i>		
SG 2	<i>SP 2.1 Collect measurement data</i>		
	<i>SP 2.2 Analyse measurement data</i>		
	<i>SP 2.3 Store data and results</i>		
	<i>SP 2.4 Communicate results</i>		
Process and Product Quality Assurance		does not satisfy	
GS 1	<i>SP 1.1 Objectively Evaluate Processes</i>	Some of the Agile practices can be used but a lot more of additional activities and artefacts must be presented to completely satisfy the goals of this CMMI	Methodology shaping Reflection workshop Sprint review
	<i>SP 1.2 Objectively Evaluate Work Products and Services</i>		
SG 2	<i>SP 2.1 Communicate and Ensure Resolution of Noncompliance Issues</i>		
	<i>SP 2.2 Establish Records</i>		
Configuration management		Somewhat satisfies	
SG 1	<i>SP 1.1 Identify configuration items</i>	Some of the Agile practices can be used but additional activities and artefacts must be presented to completely satisfy the goals of this CMMI PA	Sprint planning/Blitz planning Stories Weekly cycle/Quarterly cycle Sprint review meeting Technical environment with Automated tests, configuration management and frequent integration property(Crystal) Reflection workshop Ten minute build Continous integration\Shared code Incremental deployment Code and tests
	<i>SP 1.2 Establish a configuration management system</i>		
	<i>SP 1.3 Create or release baselines</i>		
SG 2	<i>SP 2.1 Track change requests</i>		
	<i>SP 2.2 Control configuration items</i>		
SG 3	<i>SP 3.1 Establish configuration Management records</i>		
	<i>SP 3.2 Perform configuration audits</i>		

CMMI ML3

Requirements Development		Somewhat satisfies	
SG 1	<i>SP 1.1 Elicit Needs</i>	Some of the Agile practices can be used but a lot of additional activities and artefacts must be presented to completely satisfy the goals of this CMMI PA	Stories Product Backlog Sprint backlog Product owner Customer onsite/available Exploratory 360 Essential Interaction Design
	<i>SP 1.2 Develop the Customer Requirements</i>		
SG 2	<i>SP 2.1 Establish Product and Product-Component Requirements</i>		
	<i>SP 2.2 Allocate Product-Component Requirements</i>		
	<i>SP 2.3 Identify Interface Requirements</i>		
SG 3	<i>SP 3.1 Establish Operational Concepts and Scenarios</i>		
	<i>SP 3.2 Establish a Definition of Required Functionality</i>		
	<i>SP 3.3 Analyze Requirements</i>		
	<i>SP 3.4 Analyze Requirements to Achieve Balance</i>		
	<i>SP 3.5 Validate Requirements with Comprehensive Methods</i>		
Technical solution		Somewhat satisfies	
SG 1	<i>SP 1.1 Develop Detailed Alternative Solutions and Selection Criteria</i>	Some of the Agile practices can be used but a lot of additional activities and artefacts must be presented to completely satisfy the goals of this CMMI PA	Exploratory 360 Walking skeleton Early victory Incremental Rearchitecture Techniques from Agile Modeling (out of the scope of this research) Essential Interaction Design Test Driven development Pair programming/Side-by side programming
	<i>SP 1.2 Evolve Operational Concepts and Scenarios</i>		
	<i>SP 1.3 Select Product-Component Solutions</i>		
SG 2	<i>SP 2.1 Design the Product or Product Component</i>		
	<i>SP 2.2 Establish a Technical Data Package</i>		
	<i>SP 2.3 Design Interfaces Using Criteria</i>		
	<i>SP 2.4 Perform Make, Buy, or Reuse Analyses</i>		
SG 3	<i>SP 3.1 Implement the Design</i>		
	<i>SP 3.2 Develop Product Support Documentation</i>		
Product Integration			
SG 1	<i>SP 1.1 Determine Integration Sequence</i>	Some of the Agile practices can be used but a lot of additional activities and artefacts must be presented to completely satisfy the goals of this CMMI PA	Technical environment with Automated tests, configuration management and frequent integration property(Crystal) Ten minute build Frequent integration
	<i>SP 1.2 Establish the Product Integration Environment</i>		
	<i>SP 1.3 Establish Product Integration Procedures and Criteria</i>		
SG 2	<i>SP 2.1 Review Interface Descriptions for Completeness</i>		
	<i>SP 2.2 Manage Interfaces</i>		
SG 3	<i>SP 3.1 Confirm Readiness of Product Components for Integration</i>		

	<i>SP 3.2 Assemble Product Components</i>		
	<i>SP 3.3 Evaluate Assembled Product Components</i>		
	<i>SP 3.4 Package and Deliver the Product or Product Component</i>		
Verification		Does not satisfy	
SG 1	<i>SP 1.1 Select Work Products for Verification</i>	Some Agile practices can be used but their implementation must be significantly extended in order to satisfy the Specific goals of this CMMI Process Area	User written Acceptance tests Test driven development Automated tests, configuration management and frequent integration property(Crystal) Ten minute build Frequent integration
	<i>SP 1.2 Establish the Verification Environment</i>		
	<i>SP 1.3 Establish Verification Procedures and Criteria</i>		
SG 2	<i>SP 2.1 Prepare for Peer Reviews</i>	Some Agile practices can be used but their implementation must be significantly extended in order to satisfy the Specific goals of this CMMI Process Area	User written Acceptance tests Test driven development Automated tests, configuration management and frequent integration property(Crystal) Ten minute build Frequent integration
	<i>SP 2.2 Conduct Peer Reviews</i>		
	<i>SP 2.3 Analyze Peer Review Data</i>		
SG 3	<i>SP 3.1 Perform Verification</i>	Some Agile practices can be used but their implementation must be significantly extended in order to satisfy the Specific goals of this CMMI Process Area	User written Acceptance tests Test driven development Automated tests, configuration management and frequent integration property(Crystal) Ten minute build Frequent integration
	<i>SP 3.2 Analyze Verification Results and Identify Corrective Action</i>		
Validation		Does not satisfy	
SG 1	<i>SP 1.1 Select Products for Validation</i>	Some Agile practices can be used but their implementation must be significantly extended in order to satisfy the Specific goals of this CMMI Process Area	User written Acceptance tests Test driven development Automated tests, configuration management and frequent integration property(Crystal) Ten minute build Frequent integration
	<i>SP 1.2 Establish the Validation Environment</i>		
	<i>SP 1.3 Establish Validation Procedures and Criteria</i>		
SG 2	<i>SP 2.1 Perform Validation</i>	Some Agile practices can be used but their implementation must be significantly extended in order to satisfy the Specific goals of this CMMI Process Area	User written Acceptance tests Test driven development Automated tests, configuration management and frequent integration property(Crystal) Ten minute build Frequent integration
	<i>SP 2.2 Analyze Validation Results</i>		
Organizational Process Focus		Does not satisfy	
SG 1	<i>SP 1.1 Establish Organizational Process Needs</i>	Some Agile practices can be used but their implementation will not satisfy the Specific goals of this CMMI Process Area	Reflection workshop Methodology shaping Sprint review meeting
	<i>SP 1.2 Appraise the Organization's Processes</i>		
	<i>SP 1.3 Identify the Organization's Process Improvements</i>		
SG 2	<i>SP 2.1 Establish Process Action Plans</i>	Some Agile practices can be used but their implementation will not satisfy the Specific goals of this CMMI Process Area	Reflection workshop Methodology shaping Sprint review meeting
	<i>SP 2.2 Implement Process Action Plans</i>		
	<i>SP 2.3 Deploy Organizational Process Assets</i>		
	<i>SP 2.4 Incorporate Process-Related Experiences into the Organizational Process Assets</i>		
Organizational Process Definition		Does not satisfy	
SG 1	<i>SP 1.1 Establish Standard Processes</i>	There are no Agile practices that can be mapped to the practices within this CMMI process area	
	<i>SP 1.2 Establish Life-Cycle Model Descriptions</i>		
	<i>SP 1.3 Establish Tailoring Criteria and Guidelines</i>		
	<i>SP 1.4 Establish the Organization's Measurement Repository</i>		
	<i>SP 1.5 Establish the Organization's Process Asset Library</i>		

Organizational Training		Does not satisfy	
SG 1	<i>SP 1.1 Establish the Strategic Training Needs</i>	There are no Agile practices that can be mapped to the practices within this CMMI process area	
	<i>SP 1.2 Determine Which Training Needs Are the Responsibility of the Organization</i>		
	<i>SP 1.3 Establish an Organizational Training Tactical Plan</i>		
	<i>SP 1.4 Establish Training Capability</i>		
SG 2	<i>SP 2.1 Deliver Training</i>		
	<i>SP 2.2 Establish Training Records</i>		
	<i>SP 2.3 Assess Training Effectiveness</i>		
Integrated Project Management		Does not satisfy	
SG 1	<i>SP 1.1 Establish the Project's Defined Process</i>	Some Agile practices exist that can satisfy only the SG 2.	Daily Scrum\Stand-up Whole team Customer on site/available Sprint Planning\Review meetings Reflection workshops Scrum teams Scrum of Scrum Team continuity\Shrinking teams Shared code\Collective code ownership
	<i>SP 1.2 Use Organizational Process Assets for Planning Project Activities</i>		
	<i>SP 1.3 Integrate Plans</i>		
	<i>SP 1.4 Manage the Project Using the Integrated Plans</i>		
	<i>SP 1.5 Contribute to the Organizational Process Assets</i>		
SG 2	<i>SP 2.1 Manage Stakeholder Involvement</i>		
	<i>SP 2.2 Manage Dependencies</i>		
	<i>SP 2.3 Resolve Coordination Issues</i>		
Risk Management		Somewhat satisfies	
SG1	<i>SP 1.1 Determine Risk Sources and Categories</i>	Some of the Agile practices can be used but a lot more of additional activities and artefacts must be presented to completely satisfy the goals of this CMMI PA – existing practices are more appropriate for Risk Identification (PP) and Risk monitoring (PMC)	Crystal – Coordinator/risk list Information radiator Sprint planning Sprint review Daily scrum\stand-up
	<i>SP 1.2 Define Risk Parameters</i>		
	<i>SP 1.3 Establish a Risk Management Strategy</i>		
SG2	<i>SP 2.1 Identify Risks</i>		
	<i>SP 2.2 Evaluate, Categorize, and Prioritize Risks</i>		
SG3	<i>SP 3.1 Develop Risk Mitigation Plans</i>		
	<i>SP 3.2 Implement Risk Mitigation Plans</i>		
Decision Analysis and resolution			
	<i>SP 1.1 Establish Guidelines for Decision Analysis</i>	There are no Agile practices that can be mapped to the practices within this CMMI	
	<i>SP 1.2 Establish Evaluation Criteria</i>		
	<i>SP 1.3 Identify Alternative Solutions</i>		

	<i>SP 1.4 Select Evaluation Methods</i>		
	<i>SP 1.5 Evaluate Alternatives</i>		
	<i>SP 1.6 Select Solutions</i>		

4. Quality management system	No
5. Management responsibility	No
6. Resource management	
6.1 Provision of resources	No
6.2 Human resources	
6.2.1 General	Scrum master Scrum team Exploratory 360 ⁰ Methodology shaping Side by side programming Pair programming
6.2.2 Competence, awareness and training	Sprint planning Daily meetings Scrum master Sprint review meeting Process miniature Team continuity, Shrinking teams, Root cause analysis, Whole team Keeping Quality Records is not covered
6.3 Infrastructure	Information radiators Side-by-side programming Sit together Informative workspace Pair programming Ten-minute build
6.4 Work environment	No
7 Product realization	
7.1 Planning of product realization	Scrum plan – vision + product backlog Sprint planning meeting Sprint backlog Exploratory 3600 Blitz planning Delphi estimation Weekly cycle Quarterly cycle Slack
7.2 Customer related processes	

<p>7.2.1 Determination of product requirements</p> <ul style="list-style-type: none"> • Customer requirements • Requirements specified not by customer but necessary for use • Statutory and regulatory requirements • Additional requirements determined by the organization 	<p>Product backlog Product owner prioritizing requirements Sprint backlog Exploratory 3600 Blitz planning Delphi estimation Daily stand-up meetings Essential Interaction design Stories Slack Real customer involvement? Negotiated scope contract?</p>
<p>7.2.2 Review of product requirements</p>	<p>The practices the same as the above Keeping Quality Records is not covered</p>
<p>7.2.3 Customer communication</p>	<p>Sprint a Sprint Planning? Product Owner prioritizes the Product Backlog Sprint review meeting at the end of the Sprint Reflection workshop Burn Charts? Stories Weekly cycle Quarterly cycle Slack Real customer involvement Incremental deployment Negotiated scope contract Pay-per-use Acceptance tests written by users Customer available</p>
<p>7.3 Design and development</p>	
<p>7.3.1 Design and development planning</p>	<p>Sprint planning Product backlog</p>

7.3.2 Design and development inputs	Sprint backlog Exploratory 360 Incremental re-architecture Methodology shaping Blitz planning Delphi estimation Stories Weekly cycle Quarterly cycle Slack Incremental design
7.3.3 Design and development outputs	Sprint review meeting
7.3.4 Design and development review	Reflection workshop Acceptance testing
7.3.5 Design and development verification	Code and tests
7.3.6 Design and development validation	Incremental re-architecture
7.3.7 Control of design and development changes	Continuous integration Test – first programming Incremental design Real customer involvement Incremental deployment Daily deployment Negotiated scope contract
7.4 Purchasing	No direct mapping
7.5 Product and service provision	
7.5.1 Control of production and service provision	Sprint review meeting Acceptance testing Daily deployment Incremental deployment Daily deployment
7.5.2 Validation of processes for production and service provision	No direct mapping
7.5.3 Identification and traceability	Code and tests but no direct mapping
7.6 Control of monitoring and measuring devices	No mapping
Measurements Analysis and Improvement	
8.2 Monitoring and measurement	

8.2.1 Customer satisfaction	Product Owner prioritizes the Product Backlog sprints Sprint review meeting Sprint Planning Meeting Early victory Walking skeleton Reflection workshop Weekly cycle Quarterly cycle Real customer involvement Incremental deployment Negotiated scope contract Pay-per-use
8.2.2 Internal audit	No
8.2.3 Mon & measurement of process	Scrum master Daily scrum meeting Sprint review meeting Daily meeting Reflection workshop
8.2.4 Mon & measurement of product	No
8.3 Control of non-conforming product	No
8.4 Analysis of data	No
8.5 Improvement	Root cause analysis