A Guide to Scrum and CMMI®: Improving Agile Performance with CMMI

CMMI® Institute

Contents

Introduction

Companies are increasingly turning to CMMI to improve performance of agile initiatives

Capability Maturity Model Integration (CMMI)® is being used increasingly around the world to build scalable, resilient, high performance organizations and empower those organizations to deliver on the promises of agile approaches. Organizations leverage the CMMI to scale and strengthen agile implementations and address business problems outside the scope of agile approaches. Globally, organizations are discovering that CMMI is an effective platform for achieving agile performance. CMMI is lifecycle agnostic and is useful with any software development methodology. With widespread, growing adoption in government and commercial sectors, CMMI has become a de facto model for performance improvement for software and systems engineering organizations throughout the world.

SUCCESSFUL ORGANIZATIONS NEED BOTH AGILITY AND STABILITY

In *Agility: It Rhymes with Stability*[1], Wouter Aghina, Aaron De Smet, and Kirsten Weerda argue that truly agile organizations must build dynamic capability on a stable backbone of process, governance, and organizational structure. To achieve this necessary combination of stability and responsiveness, organizations must "design structures, governance arrangements, and processes with a relatively unchanging set of core elements—a fixed backbone. At the same time, they must also create looser, more dynamic elements that can be adapted quickly to new challenges and opportunities."

Organizations are embracing the combination of CMMI and agile approaches to achieve this seemingly paradoxical combination that creates true organizational agility. The discipline, organizational learning, and consistency provided by the adoption of CMMI supports organizations in making their agile approaches even stronger and more effective.

The CMMI provides a map of "what" a high performance organization must do. Agile approaches prescribe "how" to do it. As methods and techniques are adapted and evolve, the CMMI provides the foundation upon which organizations can iterate or tailor their techniques in a way that is appropriate to the dynamics of their business environment. For software engineers, a simple analogy would be to think of the CMMI as the "requirements" or "story points" for their organization and various agile ceremonies or techniques as a particular instantiation of those requirements.

1 ***http://www.mckinsey.com/business-functions/organization/our-insights/agility-it-rhymes-with-stability***

ORGANIZATIONS USE CMMI TO IMPROVE AGILE PERFORMANCE

Agile organizations struggling with issues of performance are increasingly turning to the CMMI for proven results. The CMMI provides a model of best practices to look beyond team performance to apply lean principles at the system level. For example, Minacs IT Services experienced a 30 to 40 percent increase in attaining sprint commitments, a 30% increase in the number of user stories delivered in each sprint, and a 40% increase in on-time delivery after applying CMMI to existing agile processes. Minacs also transformed its internal work culture from silo-heavy to unified and aligned to a single common vision.

Organizations use CMMI to identify performance gaps in their processes and operations, and to provide a baseline for continuous improvement based on industry best practices. By addressing these gaps, organizations build the stability with CMMI to be more agile in their projects and programs and cut costs, improve quality, and improve on-time delivery.

CMMI HELPS TO SCALE AND SUSTAIN AGILE ACROSS THE ORGANIZATION

Organizations leverage CMMI as a platform to scale, align, and unify operations across the geographically distributed operations of large multinationals. For example, Cognizant has sustained a CMMI maturity level 5 rating and uses the CMMI with agile approaches to encourage process improvement across its globally distributed organization to meet customer-centric business objectives.

The discipline, organizational learning, and consistency provided by the adoption of CMMI practices allows organizations to use CMMI to make their agile approaches even stronger and more effective. In fact, Honeywell India used CMMI and agile across their enterprise with 7,000 engineers to improve problem-solving skills and resolve issues earlier in the development process. Results included a 12-15% decrease in functional defects, a 15% improvement in implementation of Kaizen strategy, and a shortened learning curve for employees.

CMMI IS RAPIDLY GROWING IN GLOBAL ADOPTION WITH FIRMS USING AGILE APPROACHES

In 2015 alone, CMMI adoption grew 17% globally with 28% growth in the US. During that year, more than 1,900 high-performing organizations earned a CMMI maturity level rating. With adoption in over 100 countries and a world-class Net Promoter Score of 41 as a platform for elevating organizational performance, companies deploying CMMI are very pleased with the results they are achieving.

Adoption of CMMI in organizations implementing agile is steadily increasing. In 2015, approximately 70% of appraised organizations reported using one or more agile practice. [Sourced from CMMI Institute appraisal records]. Multinational companies with technology centers in the United States, China, India and Latin America are using CMMI to scale agile practices and unify their performance platform globally across geographically distributed operations. CMMI and agile approaches are used harmoniously

at Perficient Chennai, where the organization used CMMI to scale agile across its North America, India, and China locations. They were able to reduce defects on projects by 70%. Nearly 85% of the organization's project teams have adopted CMMI maturity level 4 and 5 practices along with agile approaches for predicting a project's performance and velocity.

A PLATFORM FOR GOVERNMENT AND PRIVATE-SECTOR FIRMS BOTH LARGE AND SMALL

While CMMI continues to have a strong footprint in the aerospace and defense industries, with users such as GE Aviation, Boeing, Lockheed, Northrup Grumman, BAE Systems, and Raytheon, 90% of CMMI adoption is found in commercial sectors including mobile, finance, telecom, and IT services, at firms such as Honeywell, Samsung, Ericsson, and Fujitsu.

While CMMI is relied on heavily by large-scale multinational operations, the highest adoption is among small, high performance business units. In fact, 68% of organizations that implement CMMI have fewer than 100 employees and 22% of CMMI-appraised organizations have fewer than 25 employees.

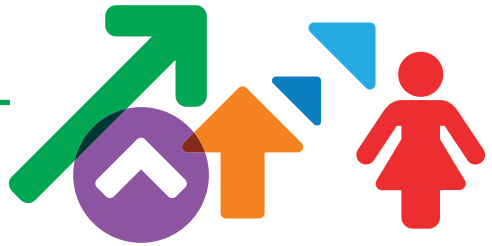
CMMI INSTITUTE ADVANCES RESEARCH TO IMPROVE ORGANIZATIONAL PERFORMANCE AROUND THE WORLD

In 2012, after decades of increasing commercial and government adoption across the globe, the CMMI Institute spun out of Carnegie-Mellon University's Software Engineering Institute. In 2016 the Institute became a part of the family of ISACA, the global non-profit professional association focused on IT governance, risk and cybersecurity. This change in structure leaves the CMMI Institute better able to execute its larger mission: advancing research in operational best practices and elevating organizational performance for the global community.

Since the transition, the CMMI Institute has greatly expanded the industries and global perspectives that contribute to its research, model development, and strategic direction. The Institute is actively collaborating with leading organizations around the world to advance the state-of-the-practice and help deliver on the promise of the *Agile Manifesto* to cultivate genuinely dynamic and adaptive high-performance organizations.

Learn more at ***http://cmmiinstitute.com/cmmi-and-agile***

Background



At the dawn of the new century, two independent events occurred that would forever change the face of software and systems engineering.

The first event was the release of the Capability Maturity Model Integration (CMMI), a maturation of earlier work at Carnegie Mellon University on the Software CMM. It introduced a broader, more comprehensive model that took a process-centric approach to aligning operations to organizational goals, improving organizational performance and the quality of products.

The initial release of the CMMI was intended for multiple engineering disciplines, but its roots in software were clearly apparent and dominated the early releases of the product.

Today we have the CMMI Institute's *CMMI for Development*, *CMMI for Services*, *CMMI for Acquisition*, the *People CMM*, and the *Data Management Maturity Model*. Holistically they provide a platform for elevating enterprise performance across the entire value chain and closely align with the needs of the software and systems engineering community.

The second event was the release of the *Agile Manifesto*. In contrast with the CMMI's empirical research-based approach, the signatories of the Agile Manifesto preferred to find common ground through collaboration while agreeing on a core set of values that were people-focused in support of the nascent methods and techniques they were developing independent of one another.

Advocates were present for many approaches to agile development within the original group, but today the most commonly adopted approaches are Scrum and Extreme Programming (XP).

The Early Adopters

Early on, the differing cultural context of early adopters led to confusion around the relationship between the CMMI and agile within the US. The CMMI is methodologically agnostic and can be applied to improve the performance of an organization, whether it is using traditional work management methods like "waterfall" or other approaches like Scrum, Kanban, Critical Chain, or Spiral.

Before spreading globally and being leveraged as a platform for organizations large and small in many industries,

"These early adopters would choose to appropriately apply the CMMI, a model for improving existing behaviors, to the way they were already doing business."

the early adopters of the CMMI were in the U.S. defense and aerospace industries. This was followed by large-scale organizations in the commercial sector, including automotive and manufacturers of products with embedded systems. These early adopters applied the CMMI to improve their existing traditional "waterfall" operations to great effect. This led to confusion, even within organizations using the CMMI, that the CMMI prescribed their existing practices. Common wisdom began to misunderstand the CMMI as a prescriptive process that was an alternative to other methods, rather than a *flexible model* that aligned any method to an organization's goals, filled gaps in the value chain, and installed a culture of continuous improvement to elevate the strategic alignment and performance of the organization regardless of the particular methods in play.

Elsewhere in software development, the agile movement was beginning to flourish. Envisioned as a way to improve collaboration and capitalize on the strength of individuals and personal accountability ("high-trust"), agile adoption grew exponentially throughout the decade, largely as a reaction to the "low-trust" characteristics of traditional approaches favored by most organizations developing software at the time.

Agile approaches eschew many of the traditional events and behaviors favored by "waterfall" organizations, choosing to forgo project audits and traditional measures, for example, in lieu of personal productivity and highly skilled teams collaborating in small groups that leverage structured behaviors.

Currently, the most popular agile approach is Scrum, a collaborative approach to managing work that emphasizes short, fixed, time-boxed iterations ("sprints"), small, consistent teams that are collocated, short-term planning, and strong peer accountability.

As such, it follows that many of the early adopters of Scrum would be either small organizations, or small teams within larger organizations. Another popular agile approach, Extreme Programming ("XP"), originated within a company that would later become one of the largest CMMI adopters, The Chrysler Corporation.

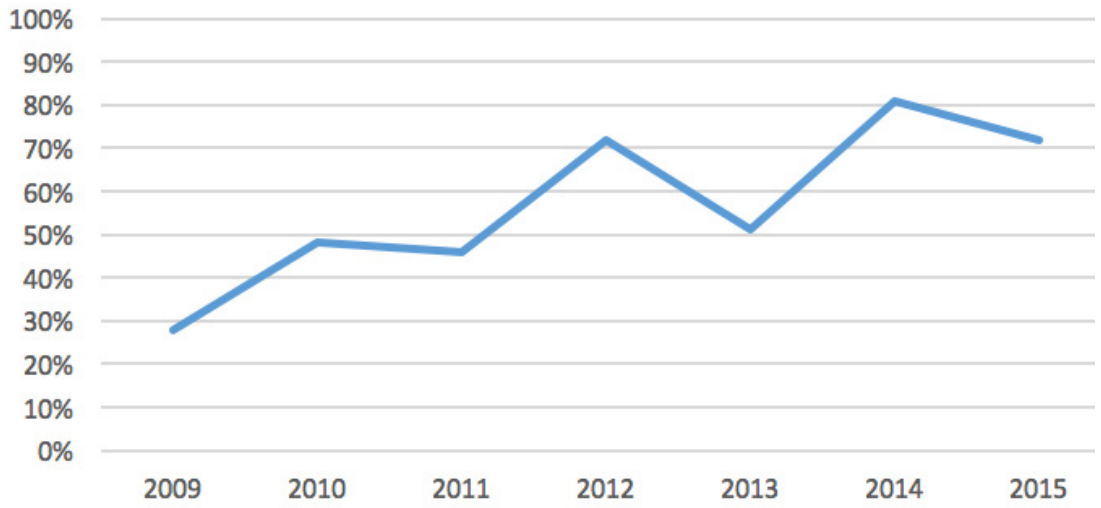
Fortunately, as agile approaches have become more widespread, organizations who were not hobbled by the cultural presuppositions of the legacies of the CMMI and agile began to use the CMMI to overcome challenges of scale, consistency, transparency, resilience, and performance in agile installations. Further, organizations that were already using the CMMI used it to approach their agile transformations in a disciplined manner that provides both the stability and dynamic capability they sought.

This document is intended to help provide clarity for practitioners on how they too can deliver on the promise of agile values by leveraging the platform of the CMMI.

"Large, early adopters drive market perception, and the perceptions of CMMI and agile have been driven by both."

"Part of this perception about CMMI is derived from the sheer size and reach of the early adopters."

PERCENTAGE OF CMMI APPRAISALS WITH AN AGILE COMPONENT



Source: 2015 research project conducted by CMMI Institute personnel analyzed over one thousand SCAMPI Appraisals from 2015 in PARS that referenced the use of Scrum, Extreme Programming, agile, and other keywords in the description of the Basic Units and Support Functions.

A Guide to Scrum & CMMI: Improving Agile Performance with CMMI

The document that follows is a roadmap for applying CMMI in agile environments. While agile frameworks can successfully be used to manage almost any type of work, they are most widely adopted in the software and systems development industry, so this document adopts CMMI-DEV, Scrum, and Extreme Programming as a baseline.

Audience

The document is intended to help three discrete constituent groups apply the CMMI to improve performance. They are represented by the following user stories:

As an agile organization

We want to understand how to use CMMI to improve performance

So that we can make agile projects rock

As an agile organization

We want to understand how to best adopt CMMI

So that we can improve while also achieving a CMMI rating

As a CMMI lead appraiser

I want to understand how to apply CMMI in an agile environment

So that I can effectively conduct an appraisal of an agile organization

To satisfy the needs of all three groups, the document has the following sections:

I. Aligning and strengthening agile ceremonies and techniques with the CMMI
II. Aligning and strengthening agile values with the CMMI
III. Performance improvement practices in the CMMI not addressed by agile
IV. References, white papers, case studies, and examples

Our recommendation is that you begin your journey toward understanding how CMMI can strengthen agile by selecting a particular ceremony or technique, jumping to that section, and then applying some of the practices and recommendations that you find there. Use this reference as guide, not as a rule book. Consider the implications of each suggestion, and decide for yourself how much value they can bring to you, your team, and your organization as a whole. Calibrate the intensity of the improvement to the value you are trying to achieve, and then move on to the next one. Don't worry about maturity levels yet, just iteratively and incrementally learn about how the CMMI can help you to improve team performance, reduce chaos, and build great products.

Don't "Do CMMI." Do agile, better.

What is CMMI?

The Capability Maturity Model Integration, or CMMI, is a process model that clearly defines what an organization should do to define, understand, and promote behaviors that lead to improved performance.

With five "maturity levels" and three "capability levels," the CMMI for Development defines the most important practices that need to be demonstrated to build great products and services, and wraps them all up in a comprehensive model.

The CMMI also helps companies to identify and achieve measurable business goals, build better products, keep customers happier, and ensure that we are working as efficiently as possible. The Capability Maturity Model Integration (CMMI)® is a capability improvement model that provides guidance for organizations to elevate performance. CMMI has five maturity levels and each level builds on the previous level for continuous improvement. Organizations can be "rated" at a Capability or Maturity Level based on over 300 practices. Organizations that work toward higher maturity levels can advance capabilities and promote more effective processes. The power of CMMI is that it helps organizations identify gaps where improving capability will make the most impact. Organizations using CMMI to build capability have increased quality, better customer satisfaction, employee retention, and improved profitability.

“CMMI is an excellent choice for strengthening agile so that it can scale and succeed for projects of any size or complexity”

Understanding CMMI

CMMI FOR DEVELOPMENT

CMMI for Development is a reference model that covers activities for developing both products and services. Organizations from many industries, including aerospace, banking, computer hardware, software, defense, automobile manufacturing, and telecommunications, use CMMI for Development.

CMMI for Development contains practices that cover project management, process management, systems engineering, hardware engineering, software engineering, and other supporting processes used in development and maintenance.

Use professional judgment and common sense to interpret the model for your organization. That is, although the process areas described in this model depict behaviors considered best practices for most users, process areas and practices should be interpreted using an in-depth knowledge of CMMI-DEV, your organizational constraints, and your business environment.

PROCESS AREAS

A process area is a cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making improvement in that area.

GENERIC GOALS

Generic goals are called "generic" because the same goal statement applies to multiple process areas. A generic goal describes the characteristics that must be present to institutionalize processes that implement a process area. A generic goal is a required model component and is used in appraisals to determine whether a process area is satisfied.

An example of a generic goal is "The process is institutionalized as a defined process."

GENERIC PRACTICES

Generic practices are called "generic" because the same practice applies to multiple process areas. The generic practices associated with a generic goal describe the activities that are considered important in achieving the generic goal and contribute to the institutionalization of the processes associated with a process area. A generic practice is an expected model component.

What is Agile?

Agile is a set of values and a related collection of methods and techniques for managing and delivering projects. Agile emphasizes iterative and incremental development, close collaboration with the customer, and cross-functional teams to build working software. Based on a set of values and principles outlined in the Agile Manifesto, agile development includes a number of frameworks such as Scrum and Extreme Programming (XP).

The agile movement evolved in response to the failures experienced by organizations employing a more traditional waterfall approach commonly used in large software development projects. Many viewed the waterfall process as document-heavy and expensive. Waterfall is based on significant up-front planning and in completing each phase of a project in a sequence of steps before moving on to the next phases. The linear approach used in waterfall contrasts sharply with the incremental and iterative empirical approach used in agile development.

Agile Values

Agile evolved with a mindset significantly different from the waterfall approach. The core values that shape agile are outlined in the Agile Manifesto, and emphasize the following:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile efforts are also shaped by the following principles, also outlined in the Agile Manifesto:

- Customer satisfaction by rapid delivery of useful software
- Welcome changing requirements, even late in development
- Working software is delivered frequently (weeks rather than months)
- Close, daily cooperation between business people and developers
- Projects are built around motivated individuals, who should be trusted
- Face-to-face conversation is the best form of communication (colocation)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Continuous attention to technical excellence & good design
- Simplicity—the art of maximizing the amount of work not done—is essential
- Self-organizing teams
- Regular adaptation to changing circumstances

These values and principles guide the agile movement and the various methods and tools that have evolved, including Scrum, Extreme Programming, Crystal Clear, and Dynamic Systems Development Method (DSDM).

Agile Issues

With its collaborative and iterative approach, agile has become widely accepted as a set of values, ceremonies, and techniques. However, several issues have emerged. For organizations seeking to adopt agile, it is often a challenge to quantitatively understand the true value and return on investment for implementing an agile approach. While there is great value to an incremental and iterative approach, the measures of success are usually anecdotal and qualitative. This presents challenges to the management approach and structure of many organizations and can drive behaviors that conflict with agile values as management burdens teams with more traditional measures.

In addition, large adopters are driving changes that threaten the agile approach by imposing requirements that conflict with agile values, ceremonies, and techniques. Organizations committed to agile values are seeking ways to instill greater resiliency in their agile implementations to counteract the negative effects of new, large-scale adopters.

How Can Agile & CMMI Work Together to Help Produce Better Software?

The table below maps some common business problems where agile ceremonies/techniques and the CMMI will provide guidance for improvement. Identify the problem(s) you need to solve, and turn to the agile ceremonies /techniques that you need to solve it. Use the CMMI Process Areas to strengthen the agile ceremonies/techniques to ensure scalability and resilience.

Problems	Agile Ceremonies /Techniques	CMMI Process Areas
PROJECT ESTIMATES ARE UNREALISTIC OR UNKNOWN.	• Team Estimating Game • Planning Poker • Sprint Planning • Backlog Grooming • Requirements Development • Task Estimation • Release Planning • Sprint Backlog	• Project Planning • Integrated Project Management • Requirements Management & Development • Measurement and Analysis
TIP: Prior to starting work, a Team Estimating Game or Planning Poker should be used to estimate the high-priority user stories from the Product Backlog to determine what can be part of the upcoming sprint; this will help to understand the complexity and cost of requirements. During Release and Sprint Planning, team velocity, if the team is largely unchanged, should be considered. This will enable the team to select an appropriate number of user story points to include in each Sprint.		
PROJECTS DO NOT GET DELIVERED ON SCHEDULE.	• Daily Standup/ Daily Scrum • Release Burn-Down • Sprint Burn-Down • Task Estimation • Release on Demand • Incremental Release	• Project Monitoring and Control • Measurement and Analysis
TIP: During a project, there should be regular backlog grooming to continue breaking down user stories for upcoming sprints. Some organizations have had positive results with more frequent “micro-grooming” sessions that focus on smaller user stories and tasks. As part of project monitoring, the Daily Stand Up meeting should review Burn-Down Charts to understand whether the Release/Sprint is on schedule.		

Problems	Agile Ceremonies /Techniques	CMMI Process Areas
REQUIREMENTS ARE ALWAYS CHANGING.	• Backlog Grooming • Sprint Planning • Product Backlog • User Stories/Epics • Definition of Done (for User Stories) • Top '10' Features • Release Planning	• Requirements Management • Project Monitoring and Control • Requirements Development • Measurement and Analysis • Verification
TIP: A project should have a Product Backlog to track all user stories (requirements). The Product Owner (customer) is responsible for keeping this backlog up to date, so if requirements change the backlog should be updated. During each Sprint Planning session, user stories are selected to be included. When Sprint Planning is completed, the Sprint Backlog should be frozen. Any changes that come in during the Sprint/Iteration should go into the Product Backlog. During a project, there should be regular backlog grooming to continue breaking down user stories for upcoming sprints.		
CUSTOMERS WON'T COMMIT TO PROJECT.	• Release Planning • Sprint Planning	• Requirements Management • Project Planning
TIP: The Scrum Product Owner (SPO) is either the customer or a customer representative/advocate. The SPO is a primary participant in Release Planning and Sprint Planning sessions where user stories are selected to be worked on. The project work should not begin unless the SPO agrees with the user stories that were selected. All User Stories should have acceptance criteria defined. During each Sprint/Iteration there is a Sprint Demo / Review where the work that has been completed is presented to the customer for acceptance.		
CUSTOMERS ARE FRUSTRATED WITH PROGRESS.	• Release Planning • Sprint Planning • Sprint Demo • Sprint Retrospective	• Integrated Project Management • Validation • Project Planning • Project Monitoring and Control • Measurement and Analysis
TIP: When a customer is a Product Owner of a project, they will be part of the team and less likely to be frustrated. The Product Owner should regularly participate in Release Planning, Sprint Planning, and Sprint Demo/Reviews so they are always up to date on project status and have ample opportunity to express concerns. The project should identify a Definition of Done for each user story to ensure things like acceptance criteria and test cases are consistently completed.		

Problems	Agile Ceremonies /Techniques	CMMI Process Areas
CUSTOMERS ARE NOT SATISFIED WITH THE END PRODUCT.	• Definition of Done • User Stories/Epics • Test Driven Development • Sprint Demo	• Validation • Verification • Requirements Development • Requirements Management
TIP: All user stories should have acceptance criteria that can be verified. Validation of user stories should occur during the Sprint Demo/Review. If both of these things don't satisfy the customer, then a new user story should be added to the Product Backlog so it can be addressed in the future.		
THERE ARE TOO MANY BUGS DURING A PROJECT OR POST PROJECT.	• Test Driven Development • Continuous Build/Continuous Integration • Refactoring	• Validation • Verification • Product Integration • Technical Solution
TIP: Continuous Build / Continuous Integration, Test-Driven Development, and Pair Programming are techniques that can improve the quality of deliverables. User Stories should always have acceptance criteria identified prior to work beginning. This will ensure that appropriate test cases can be created. All agile teams should have a Definition of Done that includes guidance for how to progress a user story and when it can be closed. This may include guidance on testing and bugs. Bugs found during a Sprint / Iteration can be addressed several ways. They should be discussed in the Daily Stand Up and the project team should determine whether the bug can be resolved during the Sprint / Iteration or not. If a story closed with a bug, then a new user story can be created to address the known bug in the Product Backlog. Refactoring can also be used to improve the product, but only if a conscious decision is made to deliver lower quality code during sprint planning, with a plan to re-design or re-code at a later date to improve code quality. Technical Debt should be considered when planning a project so that the team makes time for bug fixes and technical improvements that are needed for longevity of the product.		
UNABLE TO SECURE OR RETAIN PROJECT RESOURCES.	• Team Agreements • Release Planning • Daily Standup • Vision	• Integrated Project Management • Project Planning • Risk Management • Project Monitoring and Control • Organizational Training
TIP: During Release Planning, work to ensure that you have enough team members with the right skillset for the project. During the project, if there is a resource issue that is creating an impediment it should be raised at the Daily Stand Up.		

Problems	Agile Ceremonies /Techniques	CMMI Process Areas
LACK OF INTERNAL AND EXTERNAL SUPPORT FOR PROJECT.	• Team Agreements • Release Planning • Sprint Planning • Daily Standup • Product Owner Meeting with Customer	• Integrated Project Management • Risk Management • Project Planning • Project Monitoring and Control
TIP: The Project Team should be included in Release and Sprint Planning sessions; this will help ensure that everyone buys into the project. Project Teams can create a Team Agreement to help provide guidance for team behavior. During a project, all concerns should be raised during the Daily Stand Up.		
POOR PROJECT COMMUNICATION.	• Daily Standup • Release Planning • Sprint Planning • Spring Demo • Sprint Retrospective • Product Owner Meeting with Customer	• Project Monitoring and Control • Project Planning • Integrated Project Management
TIP: One of the primary advantages of adopting agile approaches is the superior way it deals with real-time communications. Agile ceremonies and techniques such as Release / Sprint Planning, Daily Stand Up, Spring Demo / Reviews, and Retrospectives encourage communication between team members on a daily, incremental basis. Any additional decisions regarding frequency of communication can be captured in the Team Agreement.		
REQUIREMENTS ARE VAGUE OR OPEN-ENDED.	• User Story/Epic • Definition of Done • Backlog Grooming	• Requirements Management • Requirements Development
TIP: An Epic is a type of User Story that is normally considered high level, complex, or not completely described. Epics should be refined during Backlog Grooming or Release Planning sessions until it is detailed enough to be estimated. All user stories (requirements) should be estimated and include a Definition of Done prior to being selected in a Sprint Planning session. The project team Definition of Done should include criteria for an acceptable user story.		

Problems	Agile Ceremonies /Techniques	CMMI Process Areas
PROJECT TEAM RESPONSIBILITIES ARE NOT CLEAR.	• Team Agreements • Sprint Planning • Release Planning • Incremental Release	• Integrated Project Planning • Project Planning
TIP: The project team responsibilities may be included in the Team Agreement or Definition of Done. During the Spring Planning session, all user stories should be broken into tasks and an owner self-subscribes for the work. During a Sprint/Iteration each team member is responsible for reporting on their self-assigned tasks.		
PROJECT TEAM MEMBER TRAINING IS INADEQUATE FOR THE TASK.	• Release Planning	• Organizational Training • Project Planning
TIP: During Release Planning, when the team members begin to understand the scope and content of the Release, training needed for team members should be identified and considered during planning.		
FAILURE TO SUCCESSFULLY PLAN THE PROJECT.	• Release Planning • Sprint Planning • Backlog Grooming	• Project Planning • Integrated Project Planning • Requirements Management
TIP: Scrum Ceremonies such as Release Planning and Sprint Planning, if executed with integrity, ensure project planning occurs effectively. But executing the ceremonies is not enough – guidelines related to estimation, the history of team performance, current team knowledge, and team behaviors should be developed and adhered to.		
FAILURE TO FORESEE POTENTIAL PROBLEMS.	• Daily Standup • Release Planning • Sprint Retrospectives	• Project Monitoring and Control • Risk Management • Integrated Project Management
TIP: The daily standup (daily scrum) is an excellent tool for teams to incrementally and iteratively identify risks and issues. Scrum Masters should take care to elicit these risks and issues proactively from team members, lest they fall into the plain "what I did, what I'm doing, what's blocking me" routine without providing important content. It is important that risks and issues are raised and discussed during Daily Stand Ups. Anything that qualifies as technical debt should also be added to the Product Backlog as a Risk and captured as a Story or some other unit of work.		

Problems	Agile Ceremonies /Techniques	CMMI Process Areas
PROJECT INFORMATION ISN'T AVAILABLE WHEN NEEDED.	• User Stories/Epics • Sprint Planning • Release Planning • Daily Standup • Release Burn down • Sprint Burn down • Sprint Demo • Vision	• Project Monitoring • Integrated Project Management • Risk Management • Measurement and Analysis • Verification
TIP: The Daily Standup should, in fact, occur daily for approximately 15 minutes. The Daily Standup should be attended by core Scrum Team members, the Product Owner, and other relevant stakeholders as planned. The Daily Standup should not be canceled simply because a team member may not be available at that time on any given day. All user stories should be well understood, estimated, include acceptance criteria, and be able to be completed during a Sprint. The project team Definition of Done should include criteria for an acceptable user story size. During Spring Planning, all User Stories should be broken down into tasks. Once this level of detail is achieved, progress is tracked with a Burn Down Chart. This should ensure that project progress is being tracked at an appropriate level.		
CODE QUALITY IS POOR.	• Pair Programming • Test Driven Development • Continuous Build/Integration • Retrospectives • Definition of Done	• Technical Solution • Verification • Validation • Integrated Project Management • Process and Product Quality Assurance
TIP: There are several agile techniques for improving code quality, including Pair Programming, a construct where one person codes and the other "navigates" and monitors code quality, and Continuous Build/Integration where automated regression testing will lead to improved final builds.		
ORGANIZATIONAL PERFORMANCE ISN'T IMPROVING.	• Retrospectives	• Integrated Project Management • Organizational Process Focus • Organizational Process Definition • Organizational Training • Process and Product Quality Assurance • Organizational Process Performance • Quantitative Project Management • Causal Analysis and Resolution
TIP: Central to all agile approaches is the concept of the Retrospective. "Retros" are useful for iteratively and incrementally improving team performance, but are weak in affecting enterprise-wide change.		

Elements of a Successful Agile Process Improvement Program

Agile teams rely on the "Retrospective" to provide frequent and incremental information that can lead to improved team performance. There is no construct in agile to address "enterprise" improvement specifically, although a series of Retrospectives in a "Scrum of Scrums" environment (groups of Scrum teams working in a synchronized fashion with some aspects of control vested in a single team) has proven effective in some agile environments. To conduct useful and effective organizational process improvement in any environment, including an agile one, a proven approach is to establish an integrated process team drawn from a diverse group of practitioners (sometimes called a "Software Engineering Process Group" or "SEPG") supported by one or more working groups. These working groups are focused on agile ceremonies, techniques, or other discrete areas of a performance that need to be defined or improved.

PROCESS ACTION TEAMS (PATS)

A common term for working groups that focus on improving agile ceremonies, techniques, or other processes, is "Process Action Team." Other terms in common use include "Special Interest Group," "Process Improvement Team," or "Agile Action Team." PATs are enduring groups that own, mature, and institutionalize specific agile ceremonies, techniques, or other processes. PATs design, develop, and document the sub-process, conduct and facilitate their training sessions, address prioritized corrective actions and maintain their existing process assets. Agile teams have had luck with decentralized process improvement through the adoption of this technique.

Each PAT is structured like a cross-functional support team with a PAT Leader (Process Owner) and one or more subject matter experts (SMEs) in the process area from across the organization. Each PAT gathers, analyzes, reports, and acts on metrics related to the organizational use of their process area.

Relationship to the Process Improvement Program

ORGANIZATIONAL PROCESS DEFINITION (OPD)

OPD sets the high-level expectations for the architecture of the organizations' set of standard practices, and these expectations are an input for the mission of the SEPG. As in software engineering, a good process architecture lays the foundation for future success. The architecture should be flexible enough to allow for multiple ways to accomplish an outcome. A "one size fits all" process approach is a common mistake made when interpreting the CMMI, and will most assuredly fail in an agile environment. Few if any organiza-

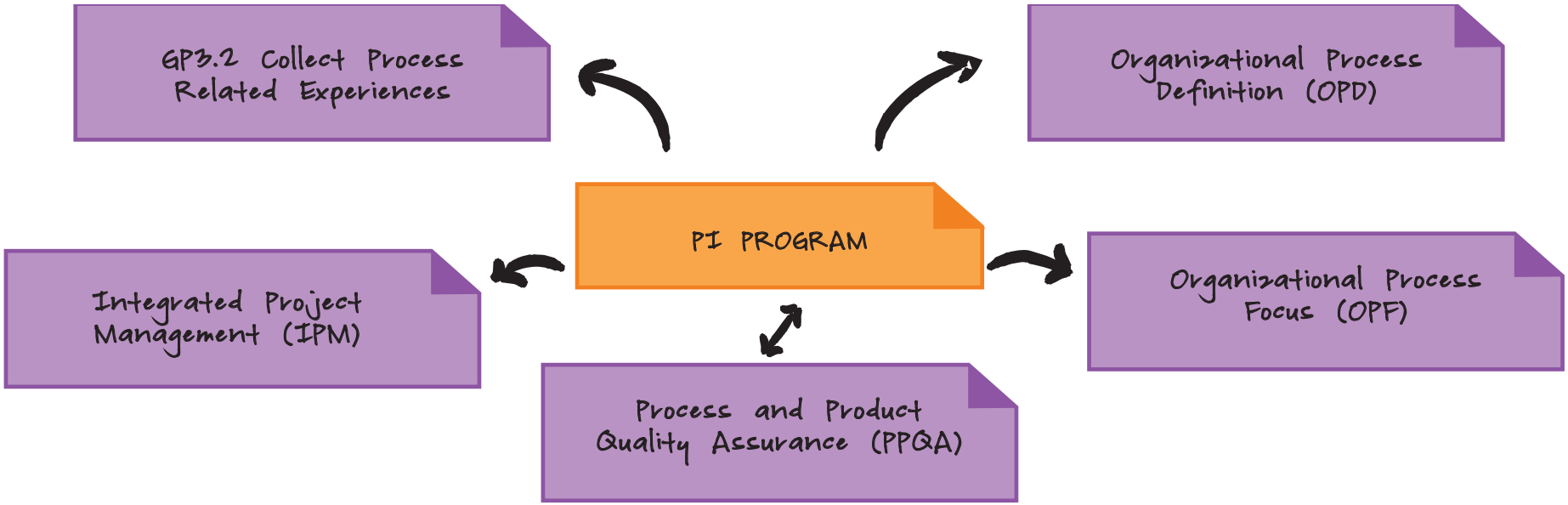
tions have projects that are exactly the same in size and scope. Don't make this mistake!

ORGANIZATIONAL PROCESS FOCUS (OPF)

OPF sets the high-level expectations for the architecture of an improvement program itself, and these expectations are an input for a way to divide responsibilities between the business, the SEPG, and PATs doing the process creation / improvement work. Working with the business, the SEPG identifies the process needs and improvement needs of the overall organization. Working with the PATs, the SEPG creates and then executes the improvement plans. And finally, working with the projects, the SEPG and PATs deliver improvements, and make sure they have the desired impact to the business. Inspect and adapt in short increments should be the goal.

INTEGRATED PROJECT MANAGEMENT (IPM)

IPM sets the high-level expectations for how the projects select from and then consume the outputs of the process improvement program. Done well, a project should be able to choose what best fits their unique circumstances while receiving enough structure to be efficient and productive.



COLLECT PROCESS RELATED EXPERIENCES

The agile value of inspect and adapt is at the core of GP3.2, Collect Process Related Experiences. A Process Improvement program that embraces this behavior returns significant value back to the projects and the business. Many times organizations gather lessons learned, but then lose them in a black hole because there is no organization that has clear responsibility for them. In a well architected process improvement program, the SEPG is the primary channel for process related experiences. New methods that work well are added to the appropriate architectural slot, and existing process that need improvement can be tasked back out to the responsible PAT. In this way the feedback loop is closed and lessons learned result in behavioral changes and improvements to the business.

PROCESS IMPROVEMENT IN AN AGILE ENVIRONMENT

Agile approaches like Scrum are perfect for running a successful process improvement program. Many agile practices and ceremonies can be easily adapted to support a process improvement program. For example:

- A Product Backlog containing User Stories can be created that defines process needs from a user perspective and an organizational perspective. An agile organization most likely has tools and methods for Product Backlogs.
- Scrum can be used to plan and manage the work of an SEPG and any groups supporting the program. A Process Action Team is one possibility. Once again, an agile organization most likely has tools and methods for Scrum.
- Planning Poker can be used in Backlog Grooming and Sprint Planning.
- Standard Information Radiators such as Burn-up / Burn-down charts can be used to show progress.
- Sprint reviews can be used to demonstrate new processes and improvements to the customers of the process improvement program.
- Sprint duration can be increased, and process improvement work can be time boxed to a maximum amount of hours each week depending on the business' ability to commit resources to improvement.

The goal of any agile process improvement program should be increased predictability and transparency resulting in an improvement program that has the confidence and support of key executive sponsors.

Using CMMI Generic Practices to Institutionalize Agile

The CMMI's "Generic Practices" provide guidance and content for assisting organizations with the institutionalizing of agile values, frameworks, ceremonies, and techniques.

Intended to be applied to all aspects of an agile implementation, the Generic Practices are grouped into three "Generic Goals," each one of them providing an appropriate level of guidance for the maturity of the organization.

Generic Goal	Generic Practice	Description
ACHIEVE SPECIFIC GOALS (GG1)		GG1 provides guidance for organizations just beginning with agile or process improvement, and guides teams to begin focusing on achieving the goals within each CMMI process area.
GG1	Perform Specific Practices	GP1.1 guides organizations that are just beginning with agile or process improvement to ensure that they are performing all appropriate practices within each ceremony or technique.
INSTITUTIONALIZE A MANAGED PROCESS (GG2)		GG2 helps organizations as they begin to institutionalize agile goals across multiple projects or scrum teams.
GG2	Establish an organizational policy (GP2.1)	GP2.1 assists organizations in setting clear expectations for performance, and the adoption of agile values such as transparency, collaboration, fail fast, fail early, iterative and incremental, and people over process, along with their related frameworks, ceremonies, and techniques.
GG2	Plan the Process (GP2.2)	GP2.2 assists agile teams by providing clarity in the definition, use and execution of the various techniques and ceremonies within each agile framework.
GG2	Provide Resources (GP2.3)	GP2.3 guides organizational leadership and teams towards the appropriate resources to support agile values. Examples include collaborative workspace, consistent teams, scrum masters, and tools.

Generic Goal	Generic Practice	Description
GG2	Assign Responsibility (GP2.4)	GP2.4 helps ensure the Product Owners, Scrum Masters, and Team Members are all aware of their responsibilities, and are given authority to perform their roles effectively.
GG2	Train People (GP2.5)	GP2.5 guides the agile organization to provide appropriate levels of training to Product Owners, Scrum Masters, and Scrum team members so they can perform their roles effectively.
GG2	Control Work Products (GP2.6)	GP2.6 assists agile teams by providing guidance towards the configuration control of work products produced through execution of agile techniques.
GG2	Identify and Involve Relevant Stakeholders (GP2.7)	GP2.7 is foundational to agile teams, who require steady and consistent participation by project participants to be successful. Not limited to the Scrum teams, participants in all ceremonies, including Sprint Demos and Retrospectives, should be considered.
GG2	Monitor and Control the Process (GP2.8)	GP2.8 guides agile teams to monitor the performance of the process itself. Example include: Are we meeting spring commitments? Do Sprint Planning, Retrospectives, and Sprint Demos occur as planned? How much technical debt are we incurring? Information gleaned through the execution of GP2.8 is invaluable for successful Retrospectives.
GG2	Objectively Evaluate Adherence (GP2.9)	GP2.9 is a valuable practice for new agile teams, who sometimes need assistance in clarifying their approach to various ceremonies and techniques. GP2.9 provides a service to the organization to ensure teams are performing in ways that support agile values.
GG2	Review Status with Higher Level Management (GP2.10)	Management has a keen interest in learning the results that come from successful agile execution, especially as it pertains to the demonstration of agile values. GP2.10 provides guidance to share that information with higher level management.
INSTITUTIONALIZE A DEFINED PROCESS (GG3)		GG3 helps organizations as they continue to mature beyond basic adoption of agile approaches by individual teams to an enterprise-wide, architectural approach.
GG3	Establish a Defined Process (GP3.1)	GP3.1 provides Scrum teams with guidelines to tailor the process to meet their specific objectives based on the organizational standards and descriptions. GP3.1 is pivotal in the scaling and enterprise-wide adoption of agile approaches.
GG3	Collect Process Related Experiences (GP3.2)	Retrospectives are the primary mechanism for capturing experiences and lessons at the project level. GP3.2 provides guidance for extending that out to the organizational level so all teams can benefit from the learning experienced at the team level.

Integrating CMMI with Agile Ceremonies & Techniques

Agile Values

Transparency. Collaboration. People over process. Fail-Fast. Iterative and incremental. These values, set to paper by the authors of the Agile Manifesto, provide a clear definition of expectations for any agile team. When these values are traced to, and integrated with, the related frameworks and techniques that are part of the agile toolbox, teams are provided with a fully-functional process architecture.

These values are particularly pronounced in Scrum and Extreme Programming (XP), where there is direct traceability between agile values and the frameworks and techniques they support.

CMMI LEAD APPRAISER PERSPECTIVE

Agile values manifest themselves throughout the CMMI model, and there is ample opportunity in the CMMI model to strengthen and improve them. This is especially true within the Generic Goals "Institutionalize a Managed Process" (GG2) and "Institutionalize a Defined Process" (GG3), which both provide guidance for the implementation and adoption of agile values, as well as the frameworks and techniques that trace directly to them.

Agile values should be considered "policy" in the vernacular of the CMMI, assuming that the accompanying frameworks and techniques do, in fact, align with them.

Traditional "waterfall" organizations may set expectations through the use of corporate policies, where agile organizations may do so by declaring Scrum to be their standard, and ensuring that all practitioners adopt it along with its related agile values. In this way, the traceability of agile values to agile techniques is "bi-directional," meaning that the adoption of one will also trace to other in either direction.

Agile Values

Practice/Goal	Strengthens agile implementation by:	Could satisfy the practice/goal if:	Could be demonstrated by:
INSTITUTIONALIZE A MANAGED PROCESS (GG2) ***GENERIC PRACTICES:*** ***GP2.1*** ***GP2.2*** ***GP2.5*** ***GP2.6*** ***GP.2.7*** ***GP2.9*** ***GP2.10***	Formalizing the adoption of agile values by providing a clear definition to be adopted by all levels of the organization. Should be supported by a plan for rollout, training, identification of relevant stakeholders, mentoring during project evaluations, and management review of project performance vis-à-vis agile values.	The values are well understood by all practitioners, and the related frameworks and techniques are adopted in a consistent manner across projects and support groups.	Training materials, policy definition, review at all-hands meetings, review and discussions during project and performance evaluation.
INSTITUTIONALIZE A DEFINED PROCESS (GG3) ***GENERIC PRACTICES:*** ***GP3.1*** ***GP3.2***	Providing guidance for providing clarity and understanding of agile values throughout the enterprise. Continuously improving and evaluating values through the use of Retrospectives and the collection of process related experiences.	Retrospectives include the collection of values-related data, including process-related experiences, technical experiences, and ideas for improving team performance The Values are clearly defined and tied to related frameworks and techniques	Retrospective records, policy and values statements, training materials.
INTEGRATED PROJECT MANAGEMENT: ***ESTABLISH THE PROJECTS DEFINED PROCESS***	Agile teams can leverage the guidance in IPM to clearly establish the process they will execute for their unique team or project, including sprint length, Release duration, sprint planning, backlog grooming, retrospectives, and more.	Agile teams select appropriate frameworks and techniques from the collection of process assets that align with agile values	Release plan; Sprint Plan, Agile Team Agreement

Agile Values, con't

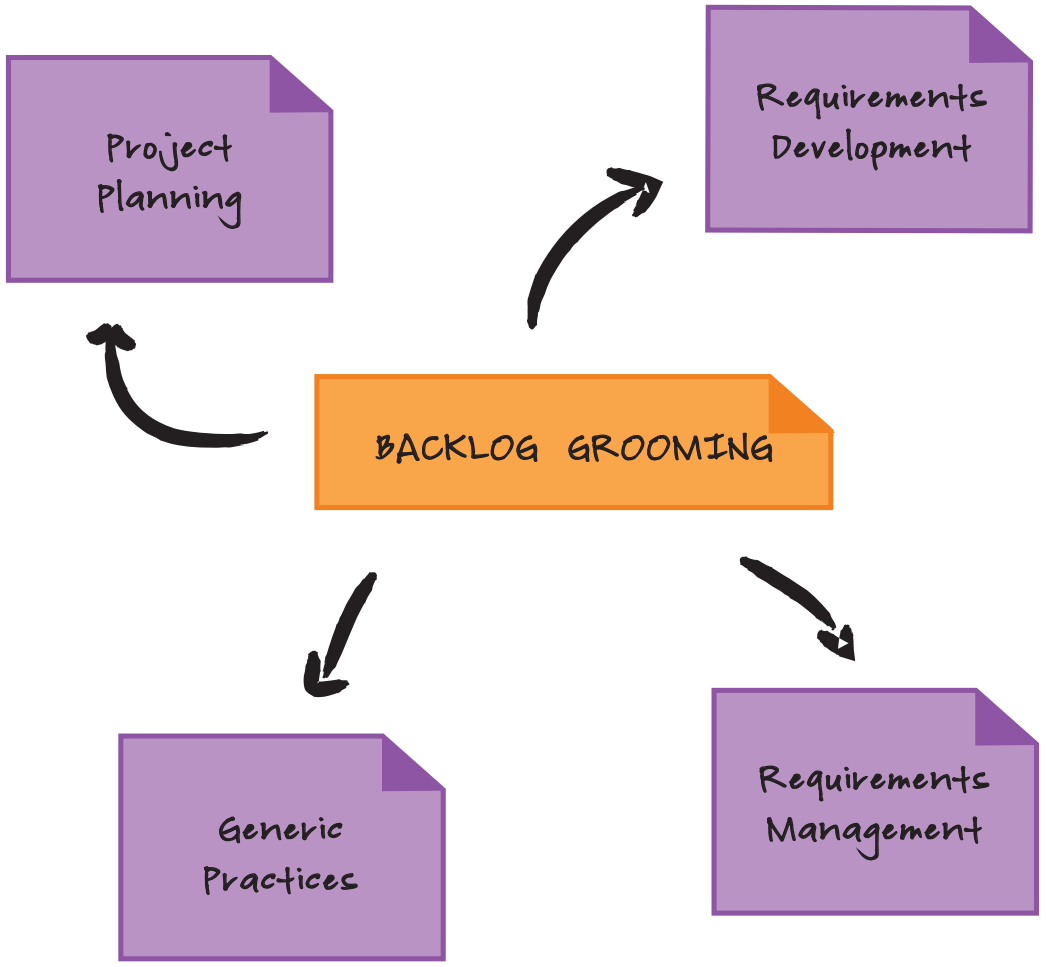
Practice/Goal	Strengthens agile implementation by:	Could satisfy the practice/goal if:	Could be demonstrated by:
ORGANI-ZATIONAL PROCESS FOCUS ***ESTABLISH ORGANI-ZATIONAL PROCESS NEEDS***	The CMMI provides guidance throughout the model on the micro-level view of agile process needs. This guidance can be applied to all ceremonies and techniques to ensure alignment with agile values.	The process needs are aligned with the agile frameworks and techniques being adopted in the organization.	A Values road-map than defines the many-to-many relationship between the agile values and frameworks and techniques being adopted.
ORGANI-ZATIONAL PROCESS DEFINITION ***ESTABLISH RULES AND GUIDELINES FOR TEAMS***	Agile teams are the primary consumers of agile values, and the CMMI provides guidelines for establishing teams so they are prepared to fully adopt the values, frameworks, and techniques defined by the organization.	A Team Charter or Team Agreement is used to define team norms, processes, frameworks, and techniques that are aligned with agile values.	Team Agreements, Team Charters, Team Contracts.

Backlog Grooming

"Using established criteria for creating or changing a user story will strengthen sprint backlogs & optimize sprint reviews."

Backlog grooming (sometimes called "story-time") is a common agile technique used by Scrum teams to produce a prioritized backlog of epics and user stories before and during a sprint. Grooming that takes place frequently during a sprint is sometimes referred to as "micro-grooming."

Backlog grooming can benefit from a wide variety of practices in the CMMI model, but Requirements Management, Requirements Development, and Project Planning are the focus. In order for backlog grooming to meet the specific practices in the process areas, an organizational structure that supports regular interaction with the customer or customer advocate (product owner) should be in place with frequent collaboration between the advocate, business SMEs, customers/end users, and the Scrum team.



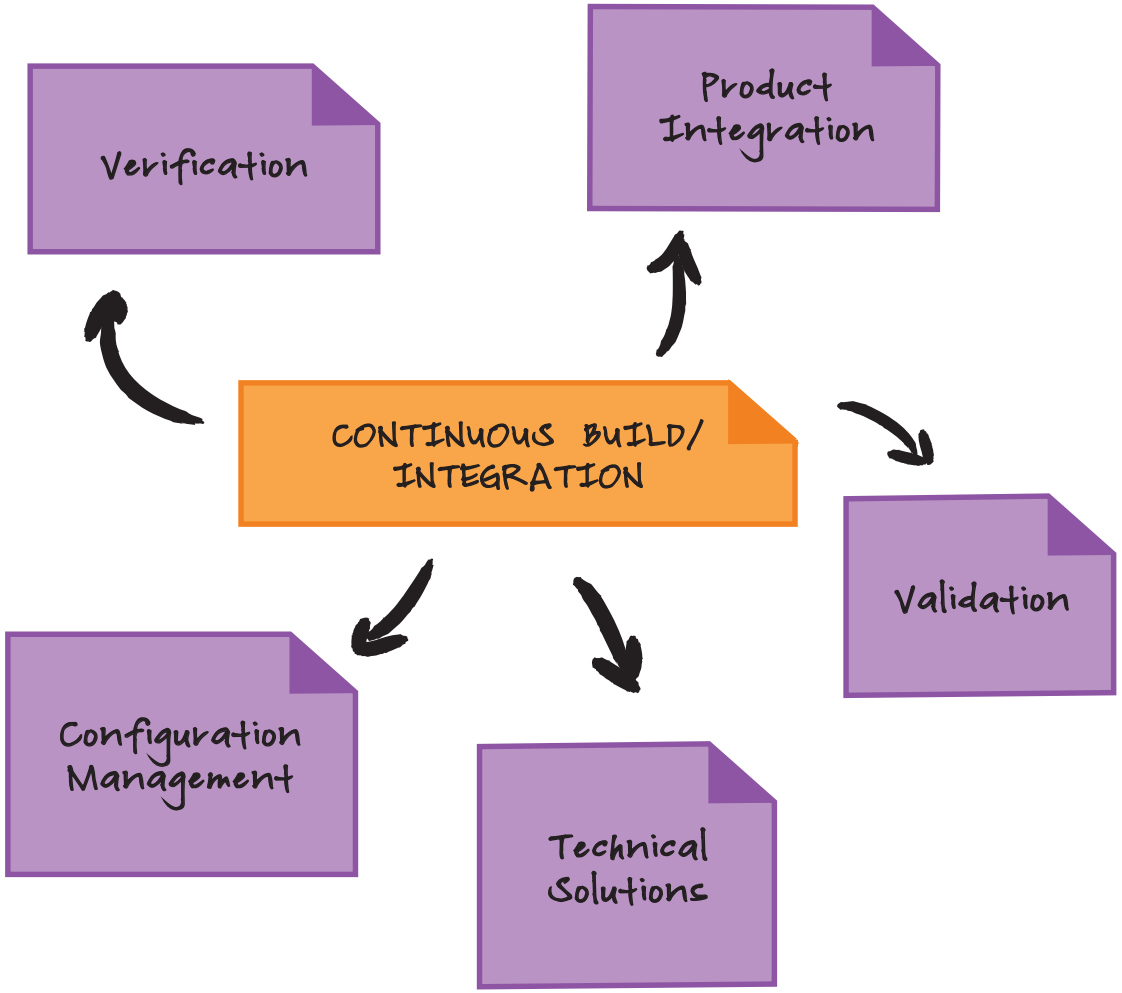
FOR MORE INFORMATION, SEE PAGE 52.

Continuous Build / Continuous Integration

"When a project assembles product components in incremental stages, using a purposeful strategy and defined procedures, successful integration is highly likely."

Continuous build/continuous integration is an approach to continuous testing and product integration popular with agile teams that was first introduced in Extreme Programming (XP).

CB/CI can benefit from CMMI practices contained in Verification, Technical Solution, Product Integration, and Configuration Management.

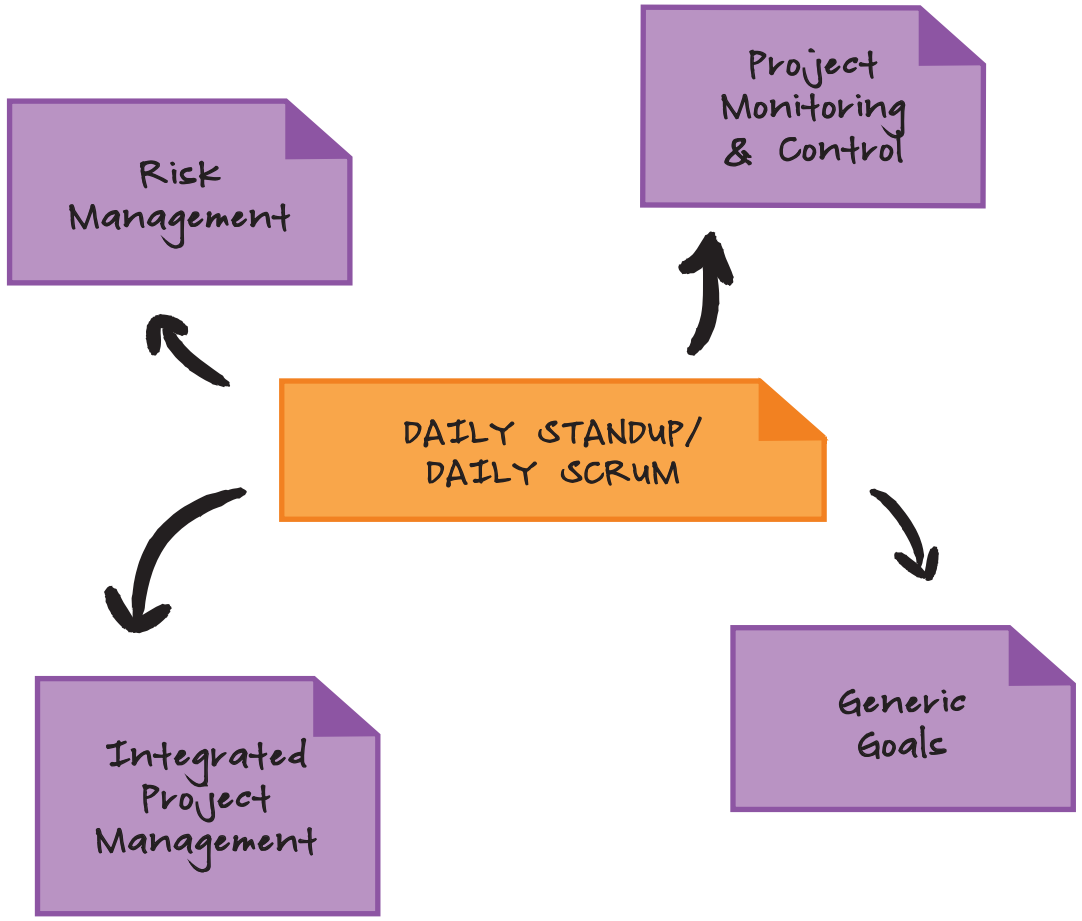


FOR MORE INFORMATION, SEE PAGE 58.

Daily Standup / Daily Scrum

"One area that is often overlooked by teams during the daily standup is the identification and analysis of risks."

The Daily Standup Meeting (aka "Daily Scrum" or "Daily Meeting") is an agile technique that is often identified with Scrum, but is popular with all types of agile approaches. It is used as a way to identify issues and risks earlier than a traditional project ("fail fast"), and to increase collaboration between agile team members. The Daily Standup can demonstrate practices within Project Monitoring and Control, Risk Management, Measurement and Analysis, Integrated Project Management, and Generic Practices GP2.8 and GP2.10 from various process areas depending on attendance.



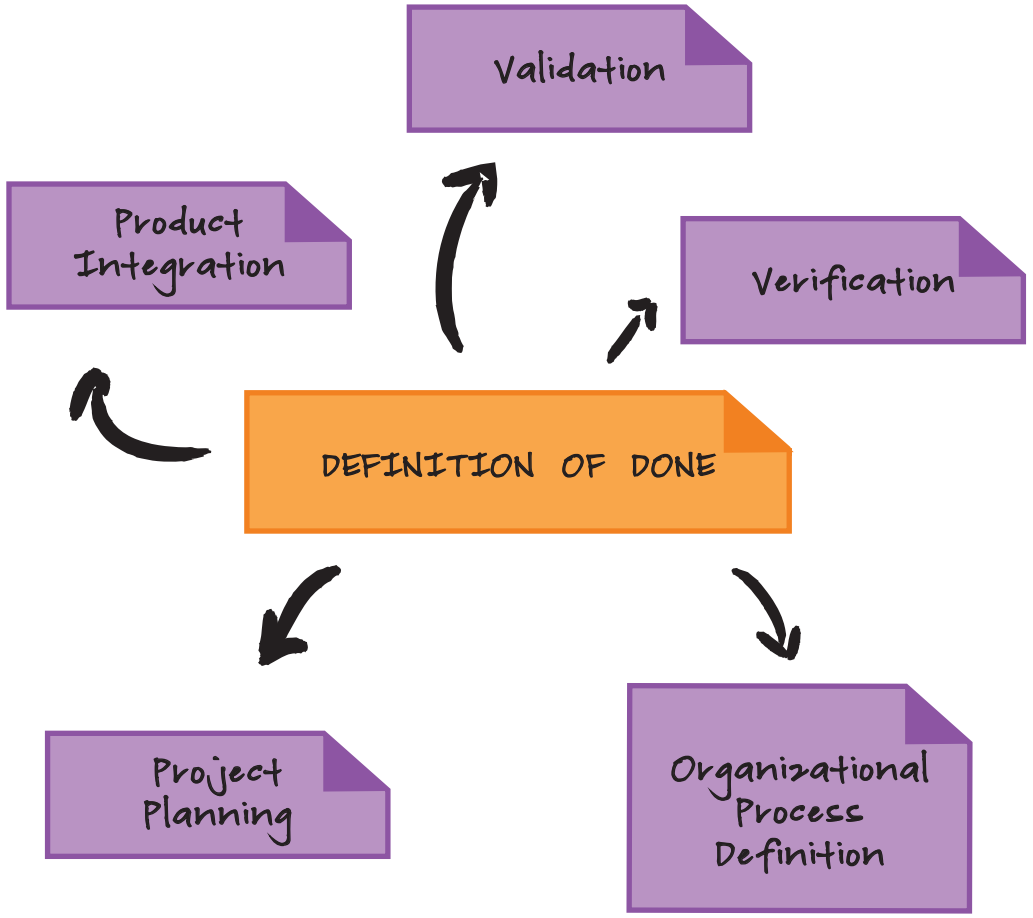
FOR MORE INFORMATION, SEE PAGE 62.

Definition of Done

"Definition of Done can be more effective when separately applied to epics, user stories, & tasks using unique criteria."

The Definition of Done (DOD) is a critical element of any agile project. It is an agreement within the agile team that indicates what must be completed for the product to be ready for the sprint review by the product owner. For most agile projects, the definition of "Done" is at the user story level, agreed to by the team and defined within each story. A story is considered "Done" when the DOD defined within the story is complete.

The DOD is a powerful tool that, when implemented properly, can provide valuable information while attempting to verify practices in Requirements Management (SP1.1 in particular), Project Monitoring and Control, Validation, Product Integration, and Technical Solution.



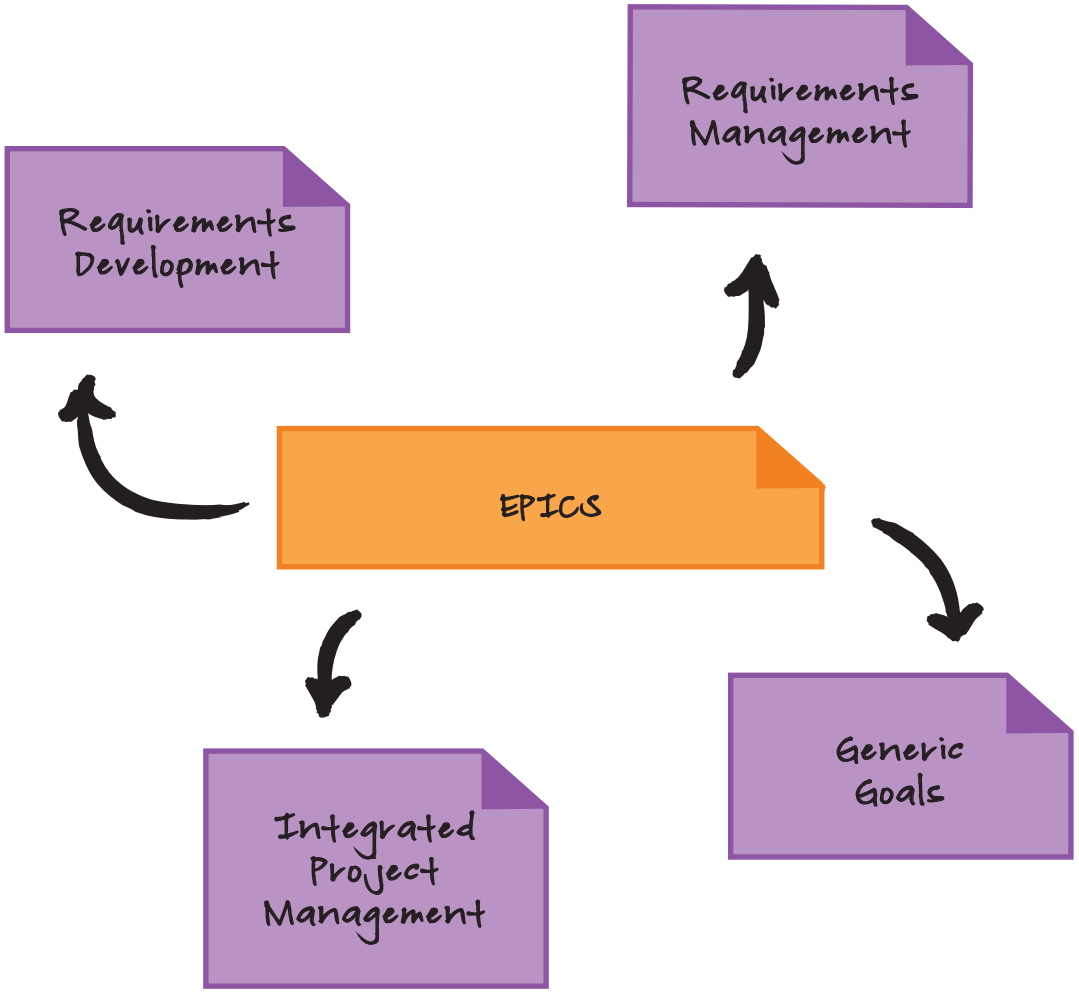
FOR MORE INFORMATION, SEE PAGE 67.

Epics

"Because epics must map to their subordinate user stories, this forms the first level of requirements traceability."

In agile terms, an "epic" is simply a large user story that will require later breakdown into individual user stories. Typically, a team will choose to break down an epic into smaller, more easily manageable user stories that can be completed in a single sprint.

An epic is a work product that is created as a result of a requirements elicitation and development process, and therefore has close ties to Requirements Development (RD) SP1.1 and RD SP1.2. Epics can be created in a joint development workshop (JAD), conference room pilot (CRP), via reverse engineering, brainstorming session, or via other requirements elicitation technique.



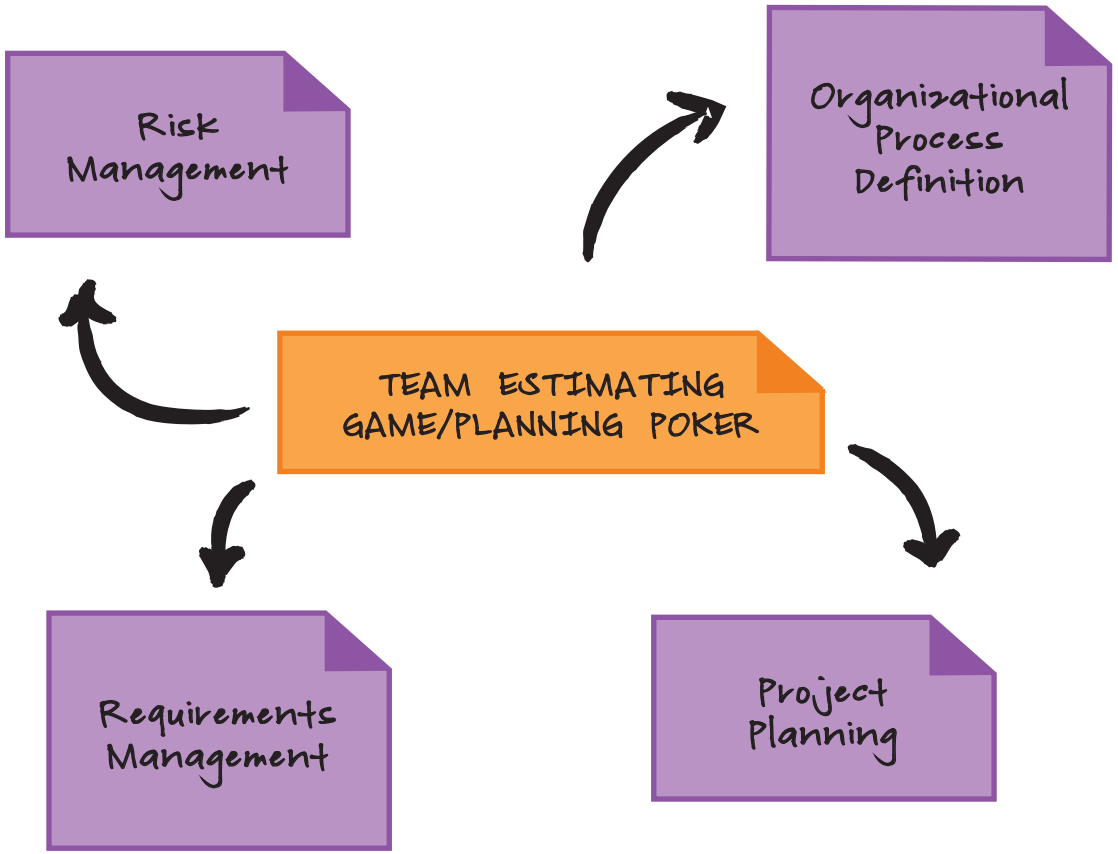
FOR MORE INFORMATION, SEE PAGE 69.

Team Estimating Game (Fibonacci Game) & Planning Poker

"The Team Estimating Game includes an in-depth discussion of the functionality of each user story by team members with a diverse set of skills."

The Team Estimating Game (sometimes called the "Fibonacci Game") is an agile estimation technique that establishes relative sizing using story points and rough order of magnitude estimation.

The Team Estimating Game primarily reflects use of Project Planning SP1.1 when placed within the context of the product backlog, but only if the user stories are fleshed out with tasks and a completed Definition of Done.

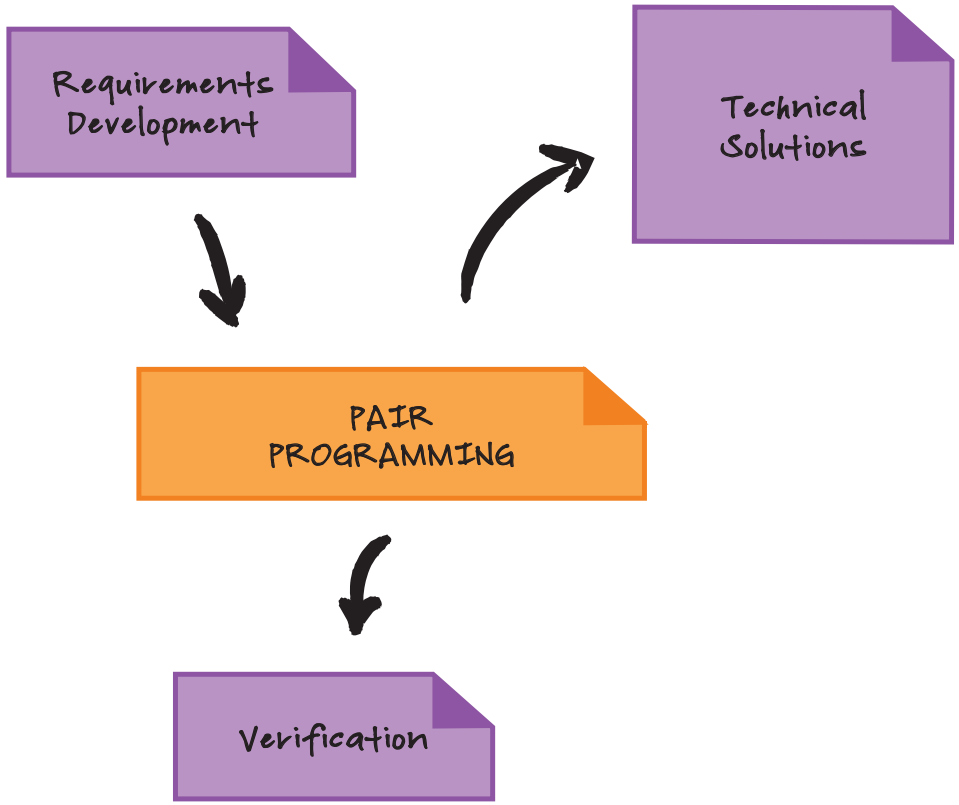


FOR MORE INFORMATION, SEE PAGE 73.

Pair Programming

"Regardless of how the pairing is done, the direct & immediate feedback loop of working together makes verification simple and efficient."

Pair Programming is an agile technique in which two software developers work together to accomplish a coding task. They generally work at one workstation with one programmer being "the driver" and the other being a "navigator" who reviews code as it is being entered by the driver. While this usually increases the initial cost of programing, it more than pays for itself in increased code quality.



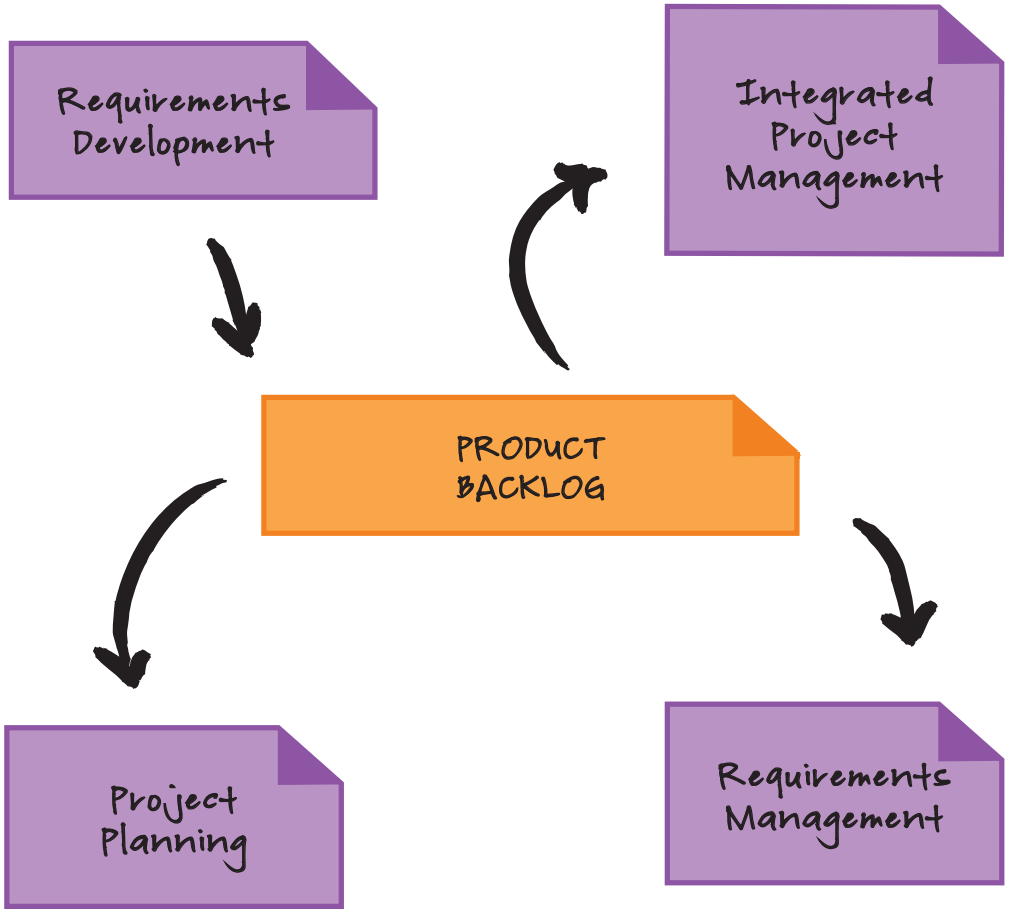
FOR MORE INFORMATION, SEE PAGE 76.

Product Backlog

"The product backlog is the foundation of an agile project regardless of the agile practices being used."

The product backlog is a prioritized list of everything that may be included in the product. It includes features, bugs, documentation changes, and any other tasks required by the product owner.

The establishment of the contents of the product backlog, user stories, initial estimates, and priorities demonstrates that the product owner understands and is committed to the requirements of the product. This data also provides the foundation for the development of project plans and the integration of related plans.

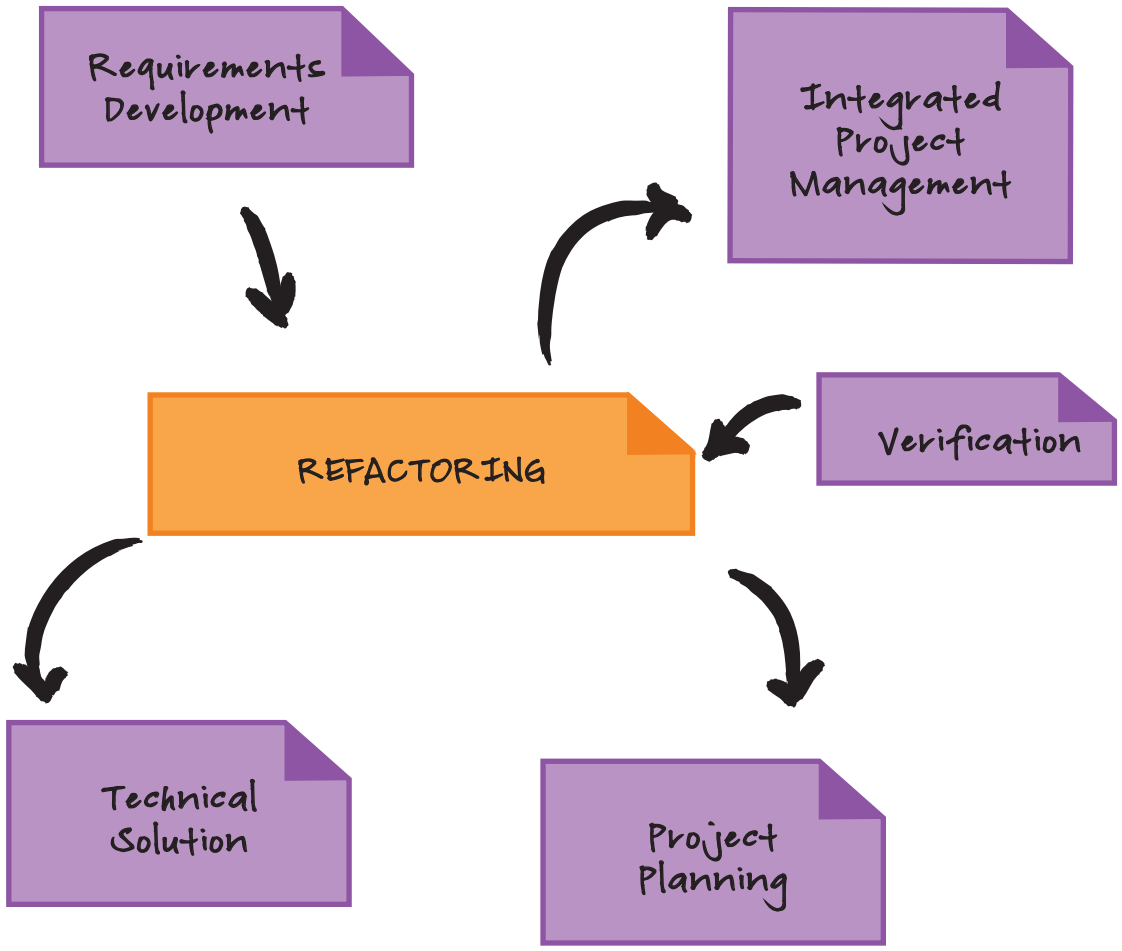


FOR MORE INFORMATION, SEE PAGE 81.

Refactoring

"Refactoring can be focused equally on code quality, requirements quality, testing, integration, or any other work product that falls within the development lifecycle."

Refactoring is a technique that is focused on improving the internal structure, style, operations, and condition of software code while maintaining its external functionality and behavior. Being focused on continuously improving the products, refactoring can be improved by applying the guidelines present in Integrated Project Management (SP1.7), Generic Goal 3 (GP3.2), and Technical Solution (SP3.1).



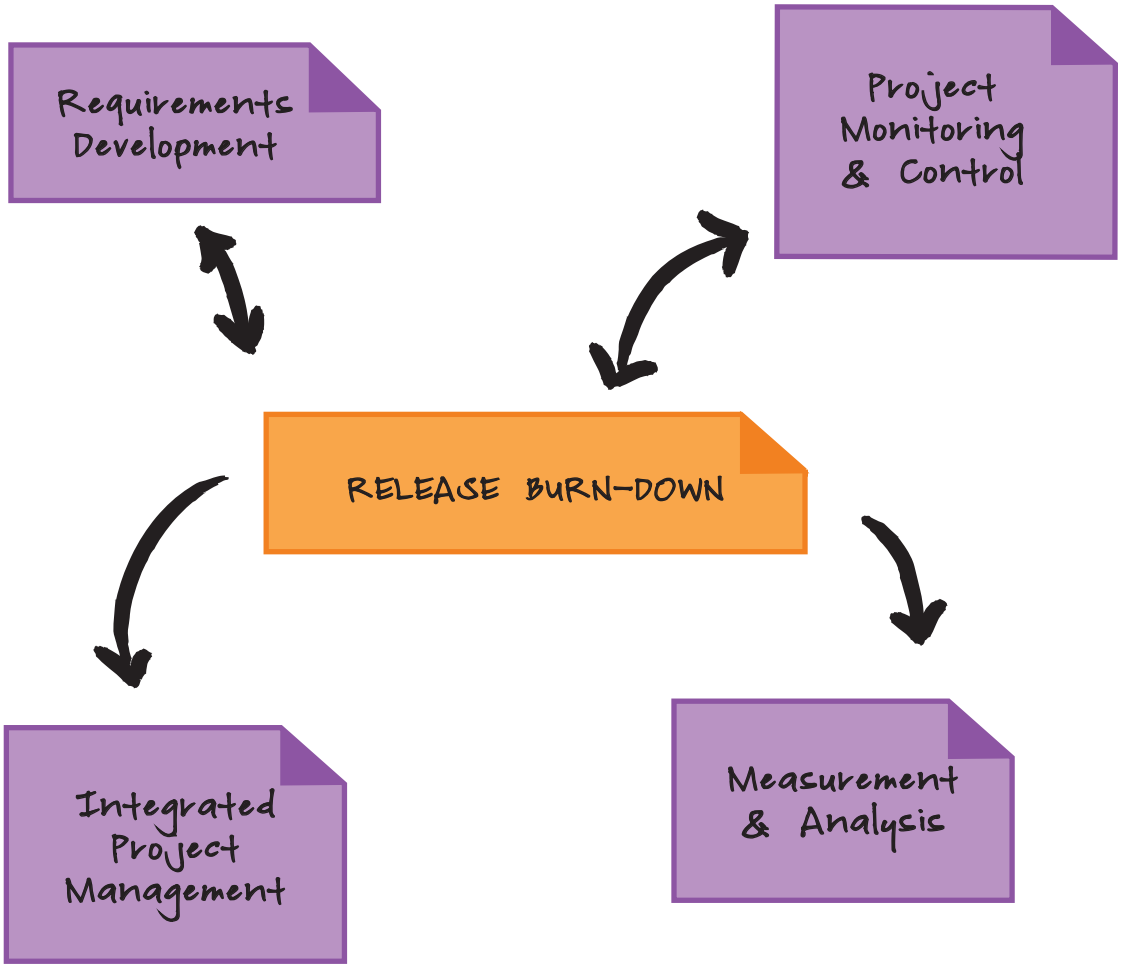
FOR MORE INFORMATION, SEE PAGE 85.

Release Burn-Down Chart

"A Release Burn down is an information radiator that provides team members with information about high-level product development progress."

A Release Burn-Down chart is an information radiator that visually depicts the trajectory of the overall release by sprint.

Release burn chart is a mechanism to monitor the release implementation progress on a high level across the sprints or iterations. The sprint burn charts and daily standups track the detailed activities during the sprint.

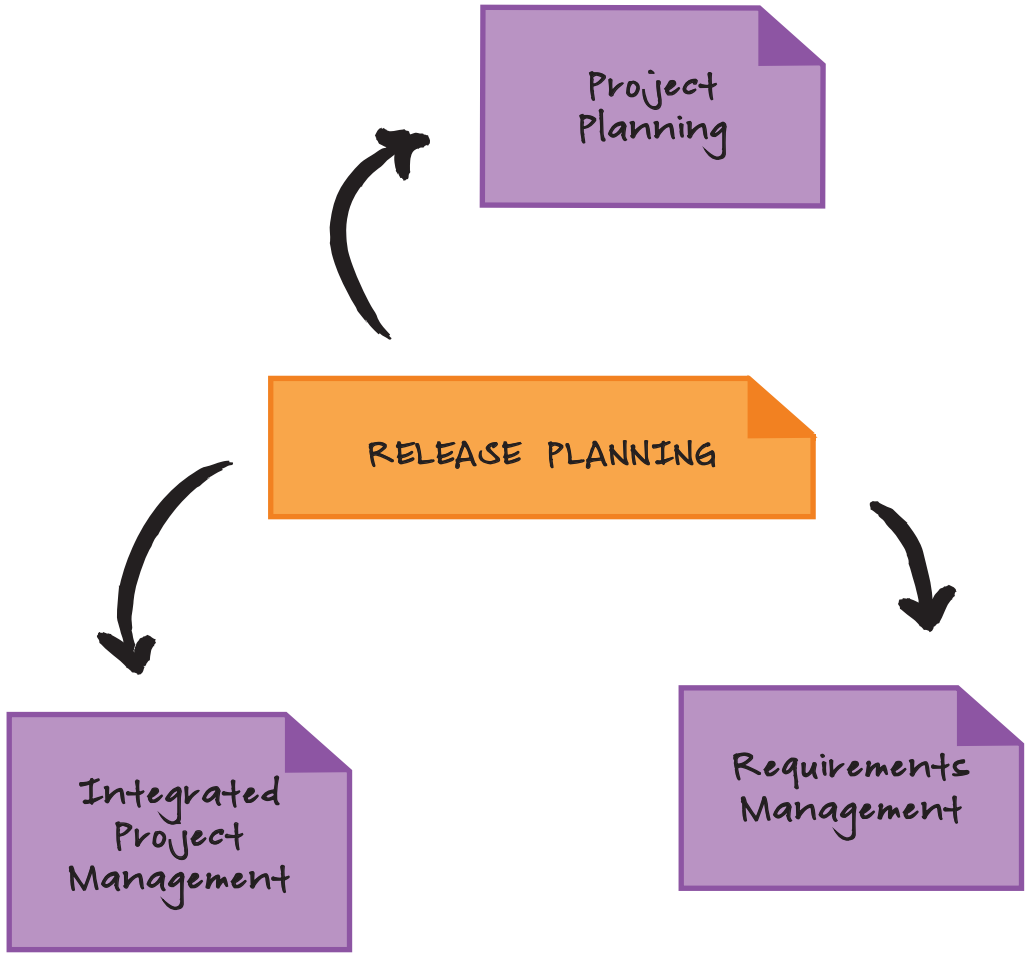


FOR MORE INFORMATION, SEE PAGE 87.

Release Planning

"Release planning forms a foundation for the overall product, and could benefit from the guidelines provide by the CMMI in Project Planning."

The purpose of release planning is to commit to a plan for delivering an increment of product value. During Release Planning the team does not plan for each detailed activity, but defers the detailed commitments such as determining who is going to do what and when, for the iteration/sprint planning.

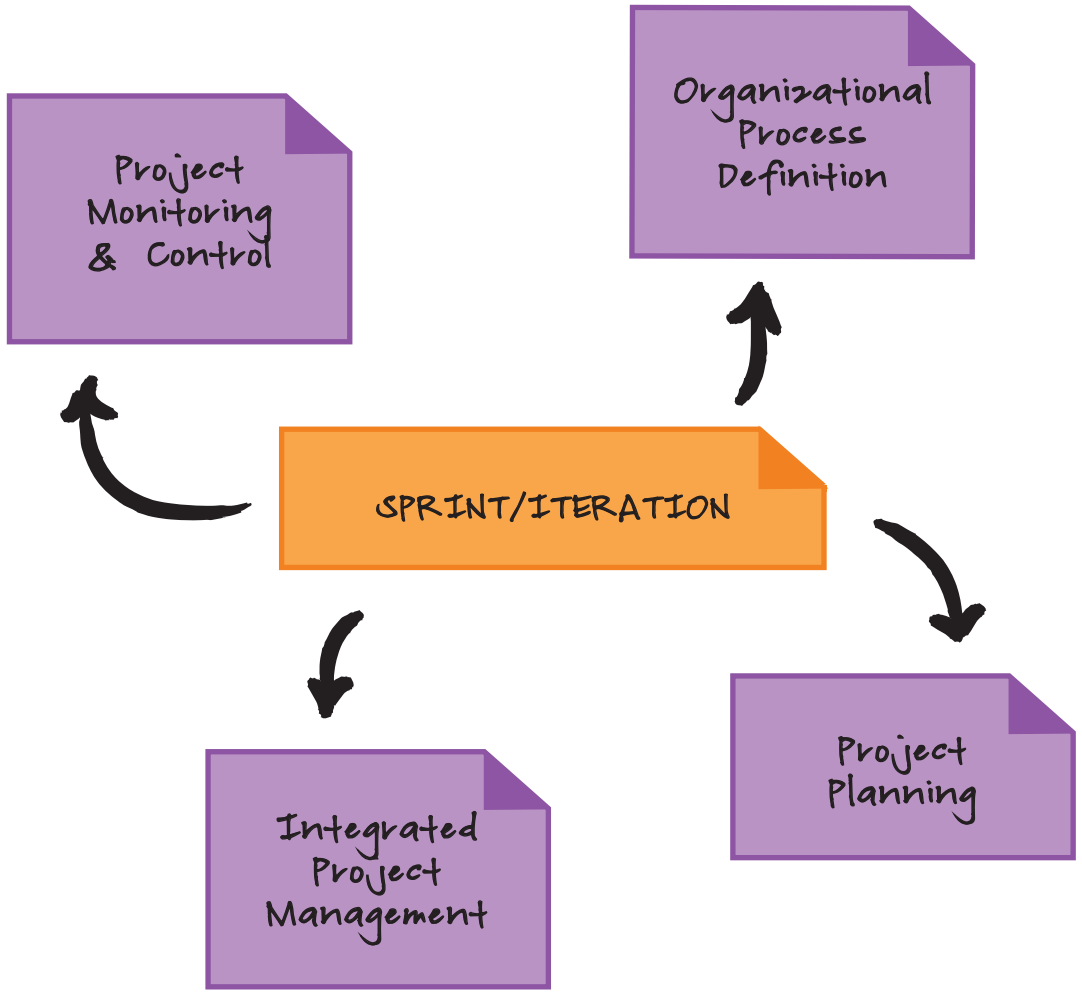


FOR MORE INFORMATION, SEE PAGE 91.

Sprint / Iteration

"Scrum teams can experiment with Sprint durations and adjust them based on factors such as risk and team experience"

The primary construct for getting work done with an agile team is the fixed-length Sprint. A Sprint is a time-boxed event where team members self-subscribe to User Stories and all work, including design, coding, testing, and backlog grooming takes place. Scrum teams vary on the inclusion of Sprint Planning, Sprint Demo, and Sprint Retrospective within the confines of a Sprint, with most reserving time inside the timebox to accomplish those tasks.



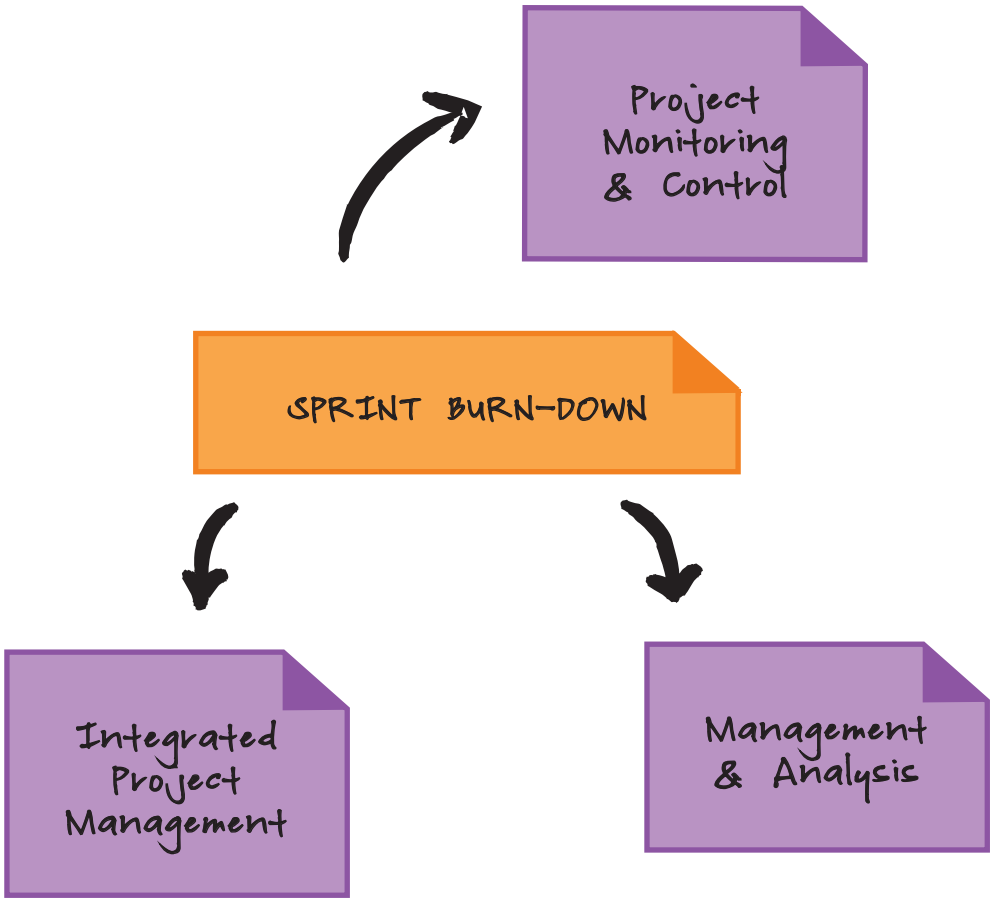
FOR MORE INFORMATION, SEE PAGE 96.

Sprint Burn-Down Chart

"Appraisal teams can learn much from information radiators, and should observe Burn-down charts while they are being updated by each team member."

A Sprint Burn-Down chart is an information radiator that visually depicts the effort remaining to complete the sprint. The Sprint Burn-Down chart helps the agile team to understand whether or not it will deliver the desired functionality by the end of the sprint as planned.

The use of burn charts in sprint tracking is to monitor teamwork progress and to manage the risks of achieving the iteration goal to deliver potentially shippable product. The team can benefit from the CMMI's Measurement and Analysis process area to enhance its capabilities for analyzing the burn chart.



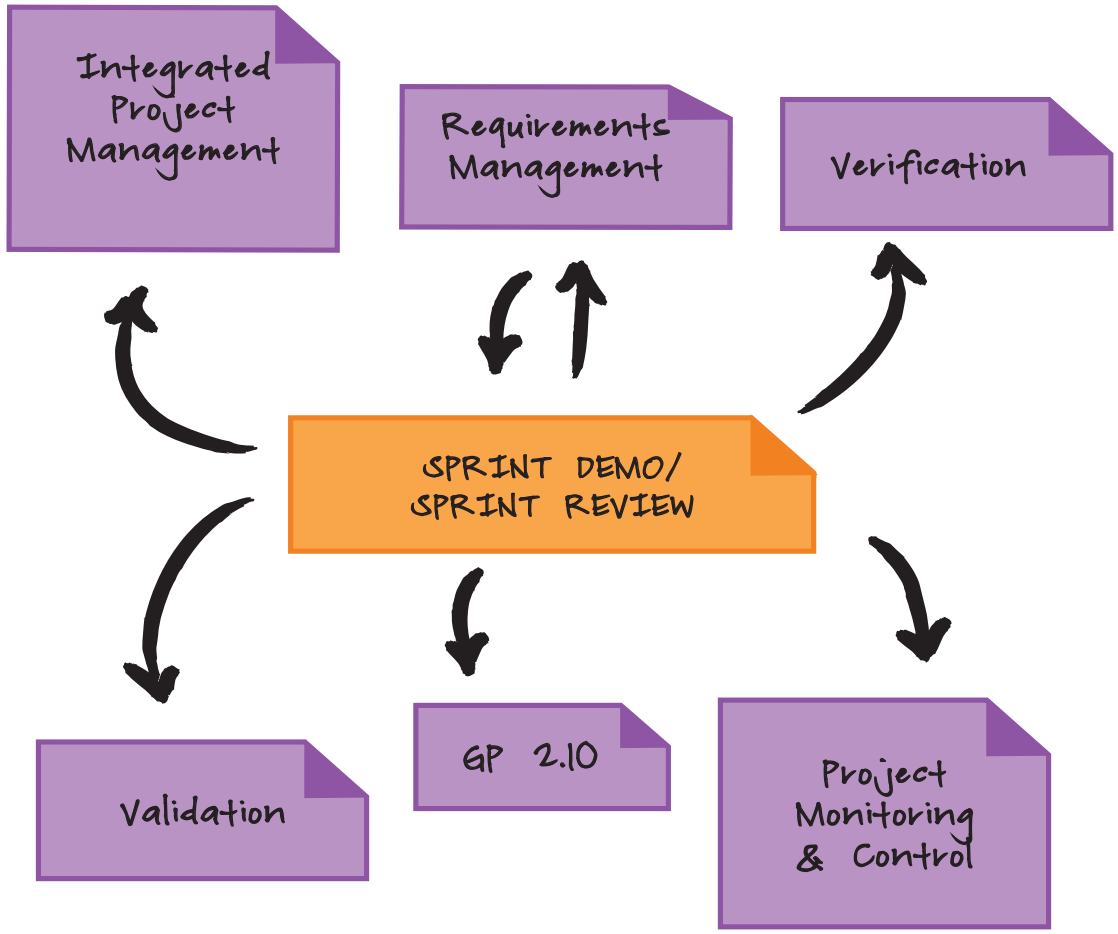
FOR MORE INFORMATION, SEE PAGE 98.

Sprint Demo / Sprint Review

"The involvement of key stakeholders is critical to the success of a sprint demo."

A Sprint Demo ("Sprint Review") is an iterative and incremental collaborative technique to ensure that all stakeholders are aware of the value that is being delivered at the end of each sprint by an agile team.

The Sprint Review/Sprint Demo can be improved through the application of practices from Project Monitoring and Control, Integrated Project Management, Verification, and Validation.



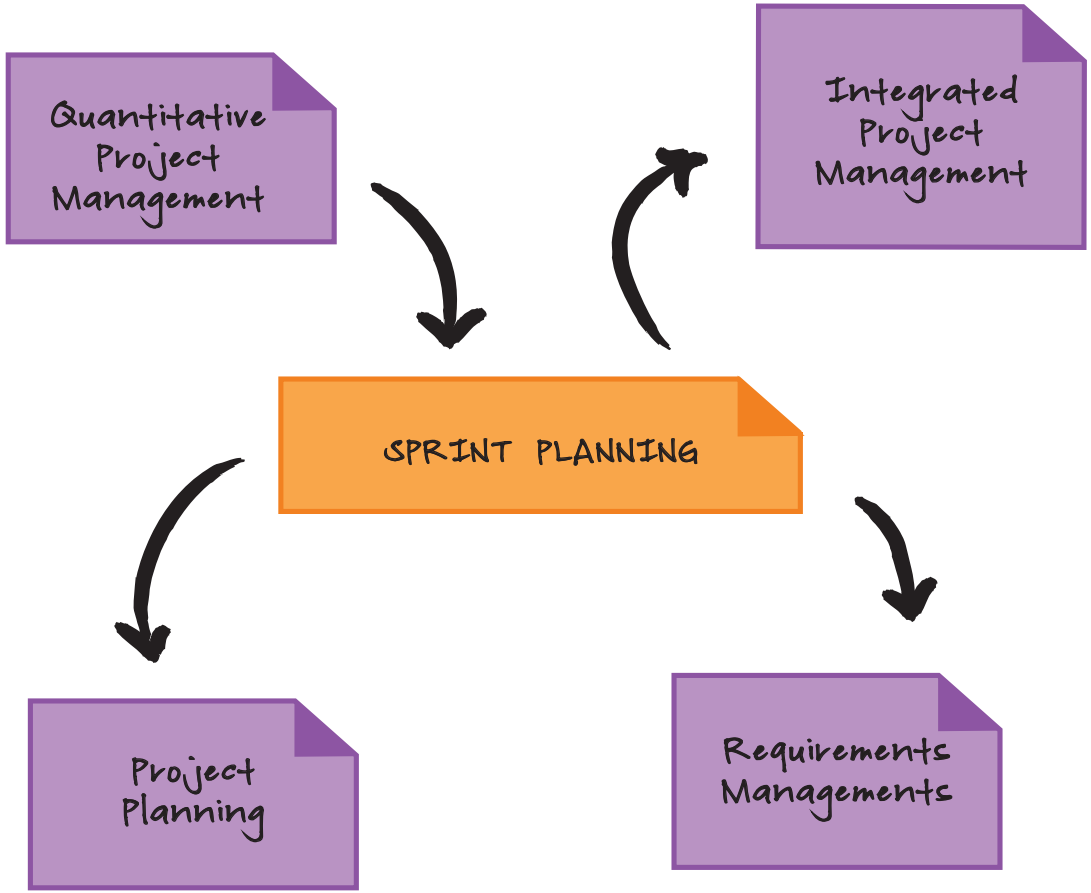
FOR MORE INFORMATION, SEE PAGE 101.

Sprint Planning

"Sprint planning can, and should, re-prioritize the team's workload as conditions change and the team learns more about the product."

The Sprint Planning Meeting occurs at the beginning of each sprint, and is a negotiation between the agile team and the product owner (or customer) as to what value will be delivered in the upcoming sprint.

Sprint planning occurs frequently during an agile project, and is an iterative and incremental form of planning, as well as a validation of the product backlog.



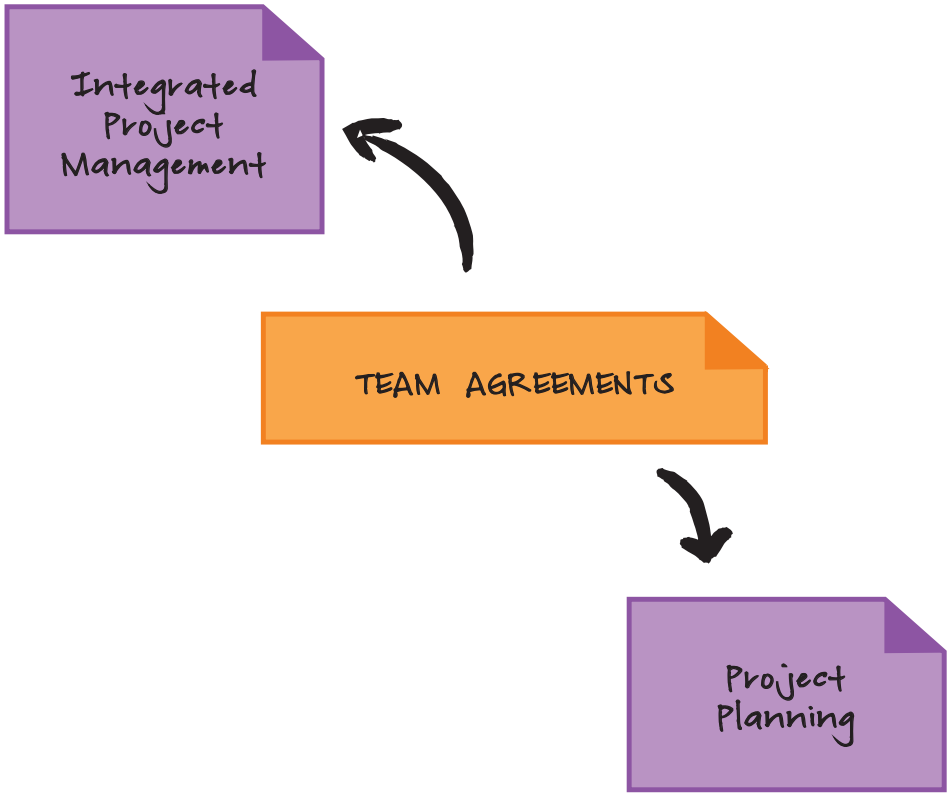
FOR MORE INFORMATION, SEE PAGE 107.

Team Agreements

"Team Agreements set team expectations for performance, and represent a micro-policy for each agile team."

A Team Agreement is a social contract entered into by members of an agile team to define team behaviors, expectations, and agreed norms. Sometimes it is referred to as a "Working Agreement."

Team Agreements can be improved through the application of practices in the CMMI from Integrated Project Management and Organizational Process Definition, where guidance for assembling teams can be found.



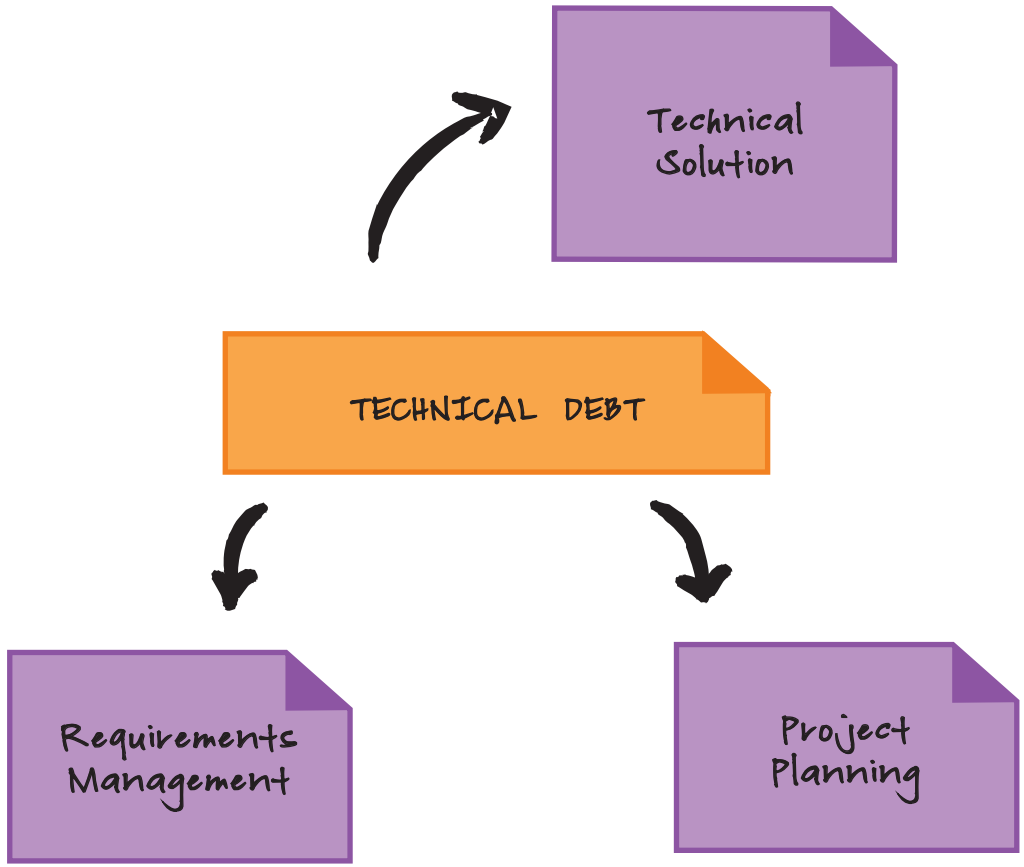
FOR MORE INFORMATION, SEE PAGE 113.

Technical Debt

"Technical debt stories can be developed and placed in the product backlog to allocated into future Sprints."

Technical Debt is incurred when the agile team proactively determines that a less optimum, less efficient, or less elegant solution is appropriate given constraints of time, budget, or resources.

As technical debt increases, the costs and effort to continue development, or maintain an existing system, become too high, and a "technical debt sprint" can be scheduled within a release to improve code quality.



FOR MORE INFORMATION, SEE PAGE 116.

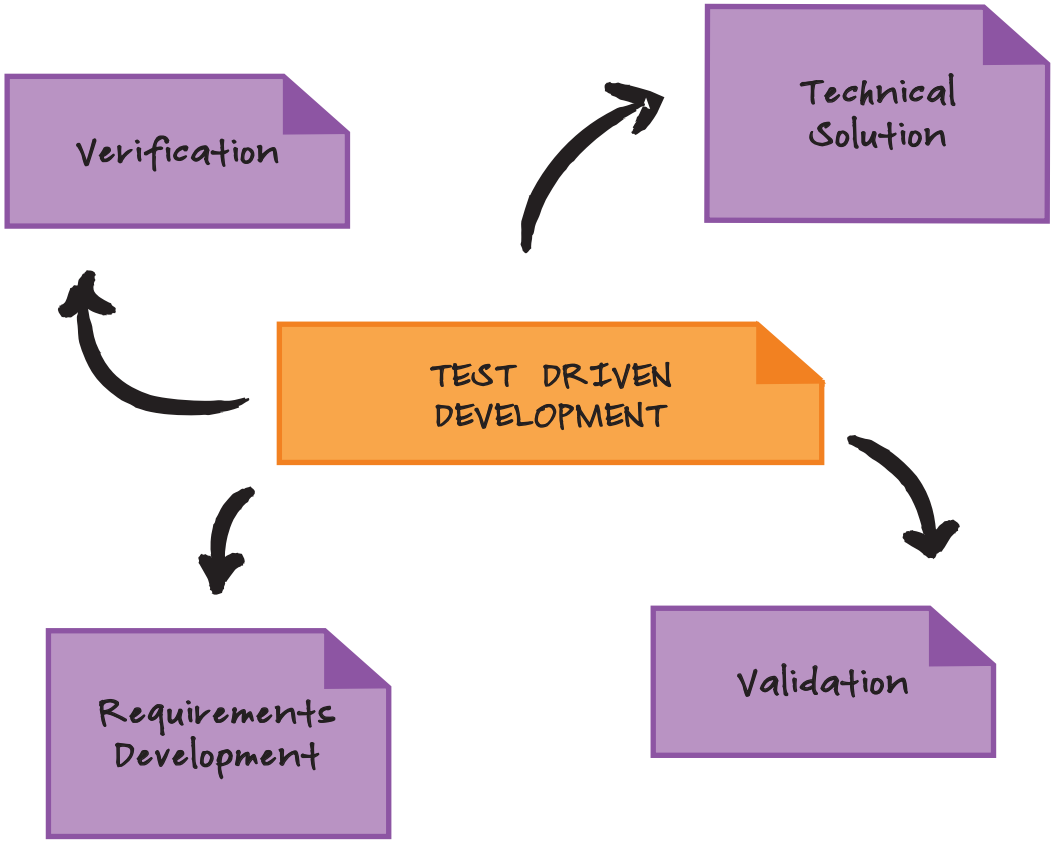
Test Driven Development

"TDD represents perhaps the best opportunity to drive defects out of the product before they are coded."

Test Driven Development, or TDD, is an agile technique most commonly used as part of an Extreme Programming (XP) project.

Used as both an early Validation and Verification technique, an individual developer will write a basic test case to verify the desired functionality, knowing that it will fail, and then writes the minimum amount of code to pass the test. The developer will then clean up the code to ensure that it meets acceptable performance and coding standards and principles.

TDD is a powerful technique that can improve the quality of code and requirements, so therefore has a strong relationship to Validation, Verification, and Requirements Development.



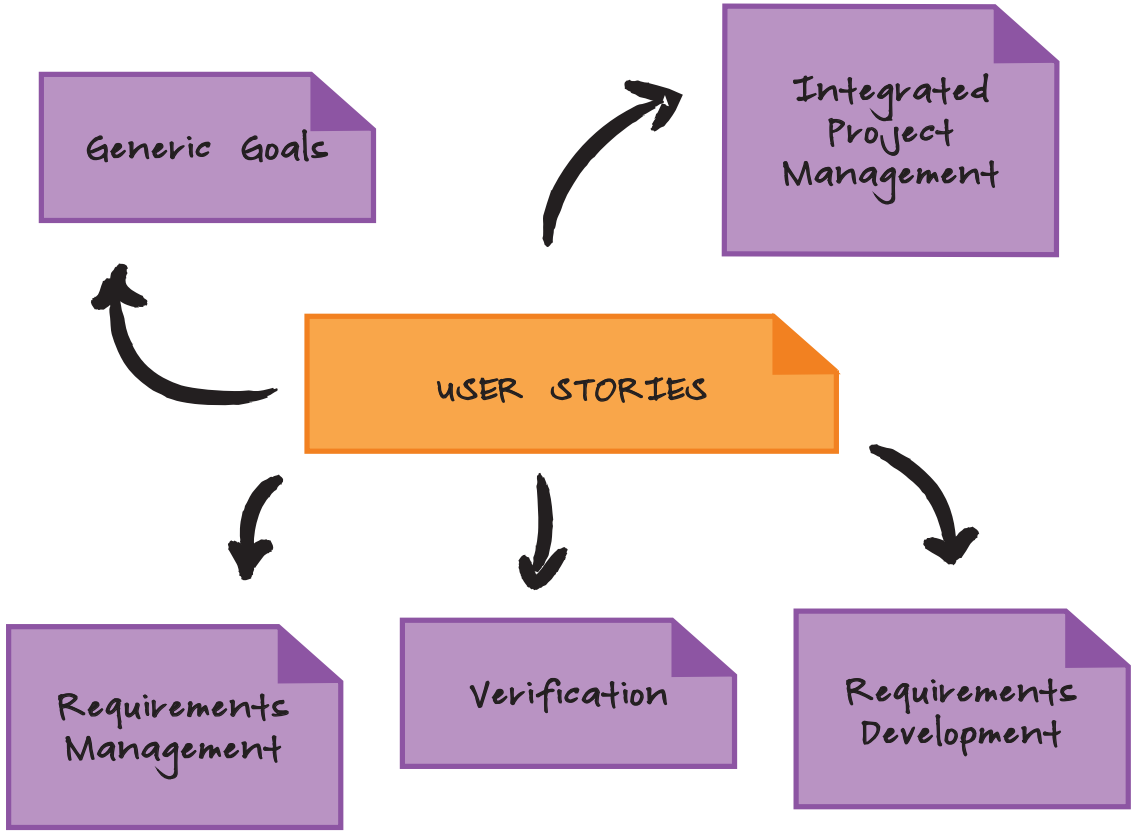
FOR MORE INFORMATION, SEE PAGE 118.

User Stories

"Users stories are most effective when used as part of a requirements architecture that includes a backlog, epics, users stories, child stories, & tasks."

A user story is a simple-language description of a discreet piece of functionality, described from the user's standpoint. Typically broken down from a larger "epic," or simply a larger user story, it is a self-contained piece of functionality that can be delivered in a single sprint.

In environments where Test Driven Development (TDD) is employed, and there is a robust Definition of Done (DOD) that identifies the criteria for acceptance of the story, agile teams experience lower defects injected during the requirements development process.



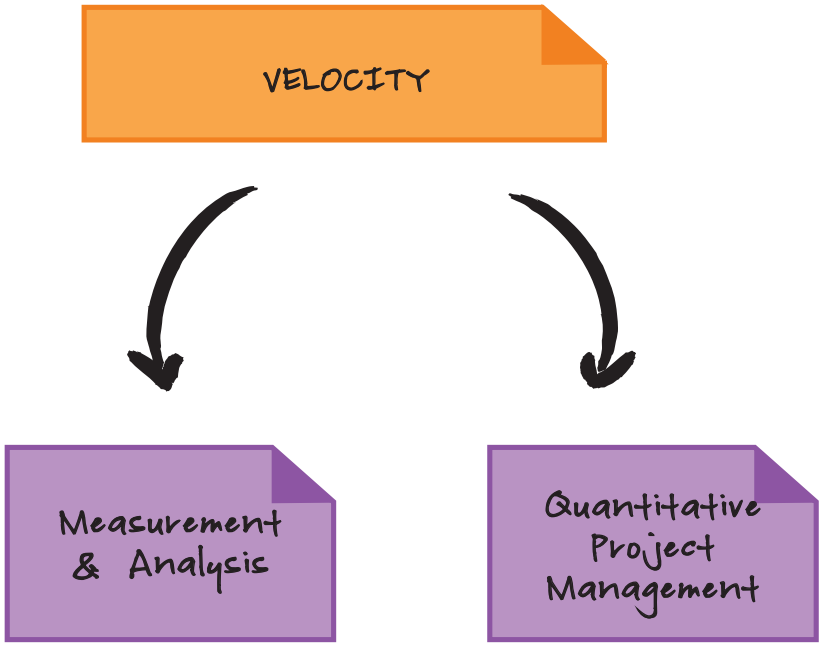
FOR MORE INFORMATION, SEE PAGE 123.

Velocity

"Velocity is one of agile's core metrics, and is used to predict whether the team can meet its sprint commitment."

"Velocity" is a tool used by agile teams for capacity planning and tracking of ongoing work during sprints and releases. Based on an agreed-upon unit of work, usually a story point, velocity is an indicator of how much value a given Scrum team can be expected to deliver in an upcoming sprint, or how much value they have actually delivered.

Velocity is a core metric for any agile team, but it is also the foundation for estimating and tracking of project work. Because agile teams work in a "fixed-time-box," "fixed-team-size" environment, measurements such as CPI and SPI are superfluous, making velocity an important measure of project success.



FOR MORE INFORMATION, SEE PAGE 126.

Using CMMI to Enhance Agile Results

Agile ceremonies and techniques can all be strengthened by appropriately applying the CMMI. The following pages address those that are most commonly used in agile software development environments.

Backlog Grooming

Continuous Build/Integration

Daily Standup/Daily Scrum

Definition of Done

Epics

Open Space Technology

Pair Programming

Planning Poker

Refactoring

Release Burn Down

Release Planning

Scrum Board/Task Board

Scrum of Scrums

Sprint Burn Down

Sprint Demo/Sprint Review

Sprint Planning

Sprints/Iterations

Team Agreements

Team Estimating Game

Test-Driven Development

User Stories

Values

Velocity

Backlog Grooming

SUMMARY

Backlog grooming (sometimes called "story time") is a common agile technique used by Scrum teams to produce a prioritized backlog of epics and user stories before and during a sprint. Grooming that takes place during a sprint is sometimes referred to as "micro-grooming".

Backlog grooming typically includes a negotiation between the product owner and Scrum team on the epics or user stories that will be added, removed, or revised for each sprint. All relevant stakeholders have input into this collaborative decision. As such, it is a critical activity related to the planning and execution of a sprint.

The product owner is primary owner of the product backlog. New epics and user stories may emerge as a result of inputs from other business SMEs, Scrum teams, or from customers/end users. It is the responsibility of the product owner to capture these within the product backlog along with associated acceptance criteria.

USING CMMI TO STRENGTHEN BACKLOG GROOMING

CMMI can strengthen backlog grooming by guiding team members toward better and stronger ways to produce a streamlined backlog with robust user stories. If the purpose of backlog grooming is to ensure clarity and readiness for the sprint planning, then Requirements Management (REQM) offers a framework for ensuring that an agile team can agree to, understand, and manage its requirements using criteria that identify a "Definition of Done" for each story or epic.

Using established criteria for a creating or changing a user story will strengthen sprint backlogs and optimize sprint reviews. A consistent structure for each user story that is clearly stated, complete, consistent with other stories, uniquely identified, appropriate, testable, traceable, achievable, and prioritized will reduce defects that are created during story development. Stronger tools and processes to manage changes will make backlog grooming and sprint reviews stronger, provide traceability of requirements to enable complete testing and simplified maintenance, and align changes in stories with the work of the upcoming sprint.

Developing a Requirements Architecture that is guided and validated by the Requirements Development process area can clarify and strengthen the relationship between customer needs, epics, user stories, tasks, and test cases. This architecture is critical to the use of agile estimation techniques, such as the use of affinity sizing, planning poker, or the team estimation game.

CMMI LEAD APPRAISER PERSPECTIVE: BACKLOG GROOMING

Backlog grooming can benefit from a wide variety of practices in the CMMI model, but Requirements Management, Requirements Development, and Project Planning are the focus. In order for backlog grooming to meet the specific practices in the process areas, an organizational structure that supports regular interaction with the customer or customer advocate (product owner) should be in place with frequent collaboration between the advocate, business SMEs, customers/end users, and the Scrum team.

The appraisal team should be able to verify that backlog grooming occurs iteratively throughout the sprint. The team should see ongoing reprioritization of the existing epics and user stories, and it is not uncommon for user stories to be removed from the backlog based on their low priority or a change in the product roadmap.

Requirements Management can be demonstrated during backlog grooming if there is consistent dialogue related to changing the product roadmap with appropriate stakeholders with the use of a "Definition of Done" to clarify the criteria used to accept and finalize the user story. This is typically done in the release planning meetings where the product vision is reviewed and modified based on feedback from the customer and the business. You may also see Scrum teams identifying new, required stories during this ceremony.

Requirements Management (REQM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
UNDERSTAND REQUIREMENTS	Provides criteria for evaluation and acceptance of requirements that are critical to writing clear and implementable user stories. The emphasis on ensuring understanding is crucial to successfully prioritize and implement these stories.	Backlog grooming occurs as scheduled; product owner, Scrum Master, and team members are in attendance; stories and epics are in the backlog; clarifying questions are asked leading to an understanding of the size and complexity of the user story or epic; definition of done for the user story indicates criteria for acceptance were met	Attend a backlog grooming session; view backlog inventory; story points are assigned and epic sizing is completed; backlog is documented on a board or in a tool
OBTAIN COMMITMENT TO REQUIREMENTS	The CMMI has an emphasis on gathering feedback and ensuring dialogue with key and extended stakeholders throughput the lifecycle of the project. Identification of extended stakeholders, and monitoring their participation in committing to requirements will help avoid misunderstandings of the meaning of requirements.	Backlog grooming occurs as scheduled; all members are in attendance; and new, modified, or deleted user stories and epics are reviewed by each member	Attend a backlog grooming session; view backlog inventory; story points are assigned and epic sizing is completed; backlog is documented on a board or in a tool
MANAGE REQUIREMENTS CHANGES	CMMI focuses on capture of the change purpose and history. This could be done so informally in agile organizations that you move closer to chaos. Some change structure ensures most important business value is delivered.	Epics and user stories are tied to tasks and a Definition of Done in a tool like Jira or Version One, or a manual process is employed.	Attend a backlog grooming session; mapping from the epics, to parent user stories, to child user stories exists (could be in a tool); backlog is documented on a board or in a tool
MAINTAIN BI-DIRECTIONAL TRACEABILITY	Bi-directional traceability helps ensure that validated epics and stories are fully implemented and identify what gaps still exist. Used to support and validate metrics related to release burn down.	Epics and user stories are tied to tasks and a Definition of Done in a tool like Jira or Version One, or a manual process is employed.	Attend a backlog grooming session; mapping from the epics, to parent user stories, to child user stories exists (could be in a tool); backlog is documented on a board or in a tool
ENSURE ALIGNMENT BETWEEN PROJECT WORK & REQUIREMENTS	The CMMI reminds us to constantly review commitments, practices in use, and how they support user story implementation. This practice would also feed the retrospective.	Backlog grooming occurs as scheduled; all members are in attendance; and user stories are clarified and expectations related to the requirement then shape team skills required, environments required, architectures required, and so on	Backlog grooming session occurs; attend a backlog grooming session; backlog is documented on a board or in a tool

Requirements Development (RD)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ELICIT NEEDS	This practice places emphasis on proactive elicitation of needs from the business, customers, SMEs, the team, and others. Annual product planning, release planning, and backlog grooming combine to help meet this practice.	Backlog grooming sessions are continuous, and can be held as scheduled meetings to identify new and changing requirements that are documented as epics and user stories and stored in the backlog	Recorded epics and user stories are documented on the wall or in a tool; mapping between epics (business requirements) and user stories is documented in each user story; user stories are documented on a board or in a tool
TRANSFORM STAKEHOLDER NEEDS INTO CUSTOMER REQUIREMENTS	This practice fully supports creation of epics and user stories. This practice includes prioritization of the backlog, which defines constraints on implementation.	Parent and children user stories are created, modified, or deleted in regularly scheduled backlog grooming sessions	User stories are documented on a board or in a tool; parent and child user stories are visible on a wall or in a tool
ESTABLISH PRODUCT AND PRODUCT COMPONENT REQUIREMENTS	This practice provides context for architecture, interfaces, non-functional, and derived requirements. These would be captured as user stories.	The modification of user stories and epics due to changes by the product owner addresses the "maintained" aspect of this SP, and establishment of requirements is met via the creation of a partner or children user stories that contain more specific data required to implement the requirement	User stories are documented on a board or in a tool
IDENTIFY INTERFACE REQUIREMENTS	This practice supports the need to capture interface requirements as user stories	Interface requirements are identified and addressed through user stories	Interface user stories are documented on a board or in a tool
ESTABLISH OPERATIONAL CONCEPTS AND SCENARIOS	This practice supports defining the user scenario (business scenarios) for how a story is implemented and the need to do so. This should be evident in the set of user stories.	Backlog grooming sessions result in the creation of user stories and epics that clarify how the user story must behave using the format, "As an XXX, I need to XXX so that I can XXX."	User stories are stored in the backlog on a board or in a tool; user stories use a standard format that clearly articulates the business scenario for how the story will be used to implement a business need.

Requirements Development (RD)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH A DEFINITION OF REQUIRED FUNCTIONALITY AND QUALITY ATTRIBUTES	This practice supports definition of "what good looks like" for the story. The logical grouping of functions supports understanding sequencing of stories, which is important to ensure shipping a fully functioning system.	The user story populated during backlog grooming contains acceptance criteria and quality attributes that define "what success looks like" and is used to define testing and user story demonstration acceptance criteria	Acceptance criteria are incorporated into the user story. Test scripts implement the acceptance criteria. Demonstrations and sprint reviews validate that the acceptance criteria are met.
ANALYZE REQUIREMENTS	This practice focuses on ensuring that all epics and user stories are necessary and support higher-level requirements (parent/child relationships). This also supports the tracing of requirements.	During backlog grooming sessions, the product owner and team analyze the epics and user stories and ask clarifying questions to understand what is required to implement them	User stories and epics are updated in the backlog to reflect the results of the analysis; attend backlog grooming sessions
ANALYZE REQUIREMENTS TO ACHIEVE BALANCE	This practice focuses on ensuring balancing of stories based on priority, size, and effort. This could help foster creation of "spikes," or experimental stories, focused on prototyping, simulations, or proof of concepts.	During backlog grooming sessions, user stories are prioritized 1 to n to ensure that the most critical stories are implemented first. This is an input to the sprint planning session where further balancing of the work is performed.	Epics and stories are documented in the backlog and are prioritized 1 to n
VALIDATE REQUIREMENTS	This practice supports ensuring epics and user stories meet the end user requirements. This can be done from spike user stories and from the basic story format if it provides inference on how the story will be used (which it should if written correctly). This is in line with one of the key purposes of backlog grooming.	During backlog grooming sessions, identification of methods required to validate the requirements (stories) will be discussed and stories created for prototypes, simulations, and demonstrations	Attend backlog grooming session; confirm that stories for validation are documented (prototypes, simulations, and demonstrations)

Project Planning (PP)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTIMATE THE SCOPE OF THE PROJECT	This practice helps the team focus on the datasets required to understand the scope. While the practice expects a "WBS," it will usually be presented in a non-traditional sense. Epics/stories arranged in a prioritized backlog, while containing the metadata that defines each story's "definition of ready/done," along with sizing, make up the functional equivalent of a traditional WBS, especially once stories are allocated to an individual sprint or iteration.	During backlog grooming sessions, epics and user stories are sized using T-Shirt sizing for epics and story sizing for user stories (typically performed using planning poker or equivalent approach)	Attend backlog grooming session; review epics and user stories and ensure sizing data is evident
ESTABLISH ESTIMATES OF WORK PRODUCT AND TASK ATTRIBUTES	This practice expects the teams to define the estimate for the work. This is done in release and sprint planning, but is also done when the backlog epics and stories are sized. During backlog grooming, this is confirmed with the creation of parent and child stories. Other factors in the practice can help the team think about options to creating code like reuse factors.	During backlog grooming sessions, parent and children user stories are sized using story sizing (typically performed using planning poker or equivalent approach) Story tasks are typically managed via sprint planning, but a number of organizations will include standard tasks in the user story	Attend backlog grooming session; review parent and children user stories and ensure sizing data is evident; review stories for task level data based on the processes employed by the organization
ESTIMATE EFFORT AND COST	This practice ensures the organization is estimating both effort and cost. In agile, the work is typically time and people constrained, so cost is a factor of # of people over time. The practice does encourage thinking about new technologies and how more stories may be required to achieve the required result.	During backlog grooming sessions, sizing of stories is used to estimate the duration required to accomplish specific work (see sprint planning), and the same techniques used during sprint planning should be used during backlog grooming	Attend backlog grooming session; review stories and ensure sizing information is provided
CONTROL WORK PRODUCTS	This practice identifies the fact that CM is a critical part of the planning process, and maintenance of versions of code and stories is important to understanding history and future needs	Epics and user stories are stored in an organized fashion and mapped to appropriate epics; parent child relationships are evident, and when updated are controlled	A tool or other appropriate mechanism may be used to maintain versions of epics and user stories
IDENTIFY AND INVOLVE RELEVANT STAKEHOLDERS (GP2.7)	This practice identifies the need to ensure that all stakeholders are identified and involved at appropriate parts of the lifecycle. Its benefit is that this is a thoughtful activity that is planned and should be discussed during backlog grooming.	Product owner, subject matter experts, and team members attend the backlog grooming session to ensure alignment.	Attend backlog grooming session; ensure that all appropriate stakeholders are present; a tool or other appropriate mechanism may be used to maintain versions of epics and user stories

Continuous Build / Integration

SUMMARY

Continuous build/continuous integration is an approach to continuous testing and product integration popular with agile teams that was first introduced in Extreme Programming (XP). CB/CI can be used in any type of software engineering environment and is not limited to agile teams. In a CB/CI environment, an application is built, and unit and (in some cases integration) tested, using automated tools each time new code is "checked-in" to the code management system. It was originally introduced as a companion to "test-driven development (TDD)," a technique in which test cases are written concurrently with user stories.

USING CMMI TO STRENGTHEN CONTINUOUS BUILD & INTEGRATION

CMMI practices for Verification, Validation, and Configuration Management map the route to a successful CB/CI experience. Selecting the products for VAL/VER, establishing the VAL/VER environments, and developing the procedures and criteria are all essential for a successful CB/CI environment. These practices ensure that the continuous build yields time saving results, repeatable tasks, and increasingly valuable increments of demonstrable functionality. Automated results and alerts will provide valuable knowledge for analysis to improve the product incrementally and contribute to the overall development.

CMMI's Product Integration practices support the iterative and incremental nature of continuous build environments. When a project assembles product components in incremental stages, using a purposeful strategy and defined procedures, successful integration is highly likely.

Using CMMI practices to guide the development and improvement of the CM infrastructure will strengthen and support an environment for constant change and continuous or daily builds. Once developed, a culture of clear and consistent behaviors supported by CM practices will increase the integrity of product baselines.

CMMI LEAD APPRAISER PERSPECTIVE: CONTINUOUS BUILD / CONTINUOUS INTEGRATION

The practice of CB/CI requires a code repository (a major component of any configuration management system) as well as servers and scripts that automatically (and at frequent intervals) collect newly checked-in code, build and compile the system, run automated unit and integration tests, and publish the results. An extension of CB/CI is to automatically deploy the built system to a test or sometimes even a production environment (this extension is not common). CB/CI can benefit from CMMI practices contained in Verification, Technical Solution, Product Integration, and Configuration Management.

Verification (VER)/Validation (VAL)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH A VALIDATION / VERIFICATION ENVIRONMENT; PERFORM VALIDATION/ VERIFICATION	Providing guidelines and examples of which work products should be subject to CB/CI (e.g., regression testing of existing code, scripts, etc.)	Developers are expected to check-in their code often (e.g., on a daily basis), that an automated build and test is run frequently (e.g., at each check-in or on a regular schedule), and that the results are made available to developers and testers; the user stories and the tasks being worked on during each sprint indicate which modules will be part of CI.	Automated unit and integration tests cases are written; TDD (Test Driven Development) is used; a "definition of done" that includes test cases in a user story, sometimes found in a tool or on a sticky note on a board; some way of tracing user stories in the sprint backlog to tasks and modules
ESTABLISH A VALIDATION/ VERIFICATION ENVIRONMENT	Providing guidelines for Validation and Verification environments	An automated build server is used to build and run unit- and integration tests; appropriate servers are installed and operating to support CB/CI; environment required to support CI has been defined and implemented—simulators, emulators, scenario generators, and interfaces with other systems.	Results of the automated build and test activity are published, for example on a web page or monitor; architecture documentation for implementation of a CB/CI implementation; continuous build servers; testing harnesses
ESTABLISH VALIDATION/ VERIFICATION PROCEDURES AND CRITERIA	Promoting a clear, documented understanding of procedures for validation of build events	The procedures for executing CB/CI are clear to team members; scripts are written to automate the process	Documented scripts used to automate the process; criteria for how quickly issues that broke the builds must be corrected
ANALYZE VALIDATION/ VERIFICATION RESULTS	Providing guidelines for relevant stakeholders, parameterization of defects discovered	The CB/CI is automated and requires no human intervention The CB/CI tools produce reports and data regarding results, and that data is routed automatically to the appropriate team members	Documented scripts Installed hardware to support CB/CI Installed software to support CB/CI Results of the automated build & test activity are published, for example on a web page or monitor
ANALYZE VALIDATION AND VERIFICATION RESULTS	See above	The CB/CI tools produce reports and data regarding validation results, and that data is routed automatically to the appropriate team members for review. Analysis of the reports and data will provide input to PMC for scheduling impact, possible corrective action, and what further activities may be required.	Build reports Emails generated by tools Defect reports

Product Integration (PI)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH AN INTEGRATION STRATEGY; ESTABLISH THE PRODUCT INTEGRATION ENVIRONMENT	Providing guidelines for policies, stakeholders, and plans for developing a strategy and environment for product integration	A CB/CI system (automated build, unit and integration tests) are in place and in use	Architecture documentation for implementation of a CB/CI implementation; build scripts; results of the automated build and test activity are published, for example on a web page or monitor
CONFIRM READINESS OF THE PRODUCT COMPONENTS FOR INTEGRATION	Providing guidelines for integration validation	A CB/CI system (automated build, unit and integration tests) are in place and in use	Build scripts; criteria for how quickly issues that broke the builds must be corrected
REVIEW INTERFACE DESCRIPTIONS FOR COMPLETENESS	Providing guidelines for interface validation	A CB/CI system (automated build, unit and integration tests) is in place and in use; assumes scripts are developed to validate interfaces as new code is checked in	Results of the automated build and test activity
EVALUATE ASSEMBLED COMPONENTS	Providing guidelines for interface validation	CB/CI system in place and in use; complete set of scripts is executed throughout the development lifecycle	Results of the automated build and test activity are published, for example on a web page or monitor

Daily Standup/Daily Scrum

SUMMARY

The Daily Standup Meeting (aka "Daily Scrum" or "Daily Meeting") is an agile technique that is often identified with Scrum, but is popular with all types of agile approaches. It is used as a way to identify issues and risks earlier than a traditional project ("fail fast"), and to increase collaboration between agile team members. Typically 15 minutes in length, the Daily Standup features input from each team member that focuses on four key areas:

- What I've done ("yesterday's weather")
- What I'll be working on
- What is blocking me

During the team round-up, members collaborate on how to react best to issues and risks that were identified by team members.

Scrum identifies attendees as the "Scrum team" or core team members, and does not recommend that customer or management representatives be involved. In practice, extended team members are often in attendance. It is typical that each person who contributes toward completion of the current sprint participates in the meeting.

Teams may use some kind of token to indicate the current person allowed to speak. This ensures that only one person speaks at a time and is fully heard by the team without interruption.

There are two methods of running a daily standup. The default method is to conduct a standup where status is provided based on user stories. The other method is called "round-robin" and is used for discussion related to tasks that are not user story bound like defect management, continuous improvements, and so on. The benefit of a user story focused discussion is a shift in focus from the individual to the story, as opposed to turning the daily standup into a status report without benefits.

USING CMMI TO STRENGTHEN DAILY STANDUP

Project Monitoring and Control (PMC) practices provide clarity and direction for many agile ceremonies, including daily standups. CMMI provides clarity on the areas that strengthen monitoring activities and ensure resolution of issues and deviations from planned progress. Daily standups offer a peer-to-peer opportunity to assess progress, report planned activities, and identify impediments. The practices in PMC can help this ceremony focus on the right things: monitoring the progress, commitments, and risks. When issues are identified, they are communicated, analyzed, and managed to closure in an open productive environment.

One area that is often overlooked by teams during the daily standup is the identification and analysis of risks. Using practices in PP, PMC, and Risk Management (RSKM), agile teams can implement a simple process to ensure that an appropriate level of risk

management is being employed by the team. During the daily standup, the Scrum Master should take responsibility for recording risks that fall within predefined categories or sources, perhaps measured by the same t-shirt sizes that are used in some agile estimation techniques.

Teams may begin to skip daily standups if they are viewed as less than valuable and productive. Other teams may let them drag on if they are not disciplined and time-boxed. Integrated Project Management (IPM) offers sound practices to help. Establishing the team's work environment will ensure that all teams have the facilities and tools needed to conduct effective daily standups. This could define tools for including remote team members, Scrum rooms with plenty of white boards, and agile management tools. IPM also provides guidance on establishing teams. Who comes to the daily standup? When does it start, how is it run, and how are decisions and activities captured? IPM practices also inform the Scrum team's role in managing some of the project's stakeholder issues, dependencies, and resolution of coordination issues.

CMMI's generic practices are vital to the success of any agile environment. CMMI's generic practices, while often viewed as "common sense," are too often overlooked. Providing the resources, training, and guidance for how projects are to perform is essential to predictable and consistent outcomes.

CMMI LEAD APPRAISER PERSPECTIVE: DAILY STANDUP/DAILY SCRUM

The Daily Standup can demonstrate practices within Project Monitoring and Control, Risk Management, Measurement and Analysis, Integrated Project Management, and Generic Practices GP2.8 and GP2.10 from various process areas depending on attendance.

Because of the collaborative and casual nature of the daily standup, appraisal teams are encouraged to observe these meetings during an appraisal, as they occur every day, and lead appraisers should be made aware of informal artifacts, such as white boards, Scrum or Kanban board, and sticky-notes, that may exist to help the team members monitor and track their work.

Project Monitoring and Control (PMC)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
MONITOR PROJECT PLANNING PARAMETERS	This practice encourages the use of the daily standup to monitor daily progress and to use the standard agile metrics. This also enables the team to determine if it has the skills required, well-defined stories, and so on when impedi-ments are identified.	Daily standups occur as scheduled, all members are in attendance, and status of user stories and tasks are reviewed by each member	Attend a daily standup; view multiple Scrum boards or tools used to collect status; Burn-Down Charts for story points and effort remaining; sprint goals and release goal completion data
MONITOR COMMITMENTS	This practice enables use of the sprint burn down and ETC	Daily standups occur as scheduled, all members are in attendance, and structured format of "what I did, what I'm doing, what's blocking me," or similar alternative is followed	Attend a daily standup; view multiple Scrum boards; Release Burn-Down chart, Sprint Burn-Down change and effort remaining; dates in a tool (e.g., Jira)
MONITOR PROJECT RISKS	This practice will help to ensure that team risks are captured and either immediately discussed, or recorded by the team for follow-up. It also encourages capture of the risk on the board or in a tool.	Risks are specifically discussed and recorded. Risks are discussed as a specific agenda item at a specific standup, not just in connection with a current blocker	Risk identified on an information radiator such as a Scrum board; risk list in a tool (e.g., SharePoint, Jira); risks notated on a user story; posting on the Scrum board on the day that risks will be discussed
CONDUCT PROGRESS REVIEWS	This practice aligns well with the purpose of a daily standup. It supports meeting conduct, metrics collection, risk identification, and imped-iment collection.	Daily standups occur as scheduled, all members are in attendance, and structured format of "what I did, what I'm doing, what's blocking me," or similar alternative is followed	The daily standup is a daily measure of team progress. Information radiators (burn-down charts, scrum board, etc.) provide additional information.
CONDUCT MILESTONE REVIEWS	This practice supports the use of a structured event to understand accomplishments at key points in the project. agile teams sometimes hold "end-of-sprint" celebrations during their last daily standup of the sprint, although this practice is most often instantiated as a sprint review.	Team reviews or retrospectives are conducted to mark the end of sprints and releases	Recorded comments and planning adjustments made at review or retrospectives conducted at the conclusion of sprints and releases

Integrated Project Management (IPM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
MANAGE THE PROJECT USING INTEGRATED PLANS	This practice helps the team to focus on the use of processes that are used consistently across the team and encourages a standard set of agile practices that can be improved based on data collected by the teams. This practice also supports requirements management and reflection against release plans.	The team is managing the daily work in accordance with the Sprint Plan and priorities confirmed on almost a daily basis with the product owner	Attend a daily standup; review the Sprint Plan and confirm alignment with the stories addressed in the standup
MANAGE STAKEHOLDER INVOLVEMENT	This practice encourages documentation of issues (impediments) based on the collaboration across teams and with stakeholders. Documented impediments are the key outputs.	The team, product owner, and other impacted stakeholders are present and participating; dependencies and any risks/impediments are identified	Attend a daily standup; observe attendees and ensure the team is present; dependencies are identified in the story, on the Scrum wall, or in a tool
MANAGE DEPENDENCIES	This practice focuses on managing critical dependencies from other teams and the need to document and monitor those dependencies. While in agile, all efforts are made to reduce the work such that these types of dependencies either do not exist or are marginalized. They will occur, and it is important the team think this through and monitor it carefully.	The team will ensure that cross-team dependencies are identified within the stories and are discussed	Attend a daily standup; dependencies are identified in the story, on the Scrum wall, or in a tool

Risk Management (RSKM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
IDENTIFY RISKS	This practice ensures that risks and impediments are captured as a result of the daily standup.	Risks and impediments are specifically discussed and recorded	An information radiator such as a white board with sticky notes; a tool (e.g., Jira or SharePoint)
EVALUATE, CATEGORIZE, AND PRIORITIZE RISKS	This practice ensures that the team takes the time to understand each risk and impediment so that each is addressed based on urgency and criticality to achieving the sprint goals.	Risks and impediments are specifically discussed and prioritized, and their categories and sources are recorded.	An information radiator such as a white board with sticky notes; a tool (e.g., Jira or SharePoint)
DEVELOP RISK MITIGATION PLANS	This practice ensures that each risk and impediment is not only understood but that someone is assigned to work it and a plan to address it exists (could result in a user story to specifically address the impediment.	Risks and impediments, and their mitigation plans, are specifically discussed and recorded.	An information radiator such as a white board with sticky notes; a tool (e.g., Jira or SharePoint)

Definition of Done

SUMMARY

The Definition of Done (DOD) is a critical element of any agile project. It is an agreement within the agile team that indicates what must be completed for the product to be ready for the sprint review by the product owner. For most agile projects, the definition of "Done" is at the user story level, agreed to by the team and defined within each story. A story is considered "Done" when the DOD defined within the story is complete. The DOD might include:

- Code and test cases written
- Code peer reviewed (or produced with pair programming) and meets development standards
- Code unit tested and passed
- User story tested and passed
- Code deployed to system test environment and passed system tests
- Code deployed to integration environment and passed integration tests
- User story/test cases passed UAT (User Acceptance Testing). The UAT is based on the acceptance criteria that were established for the user story/test case.
- The remaining hours for task set to zero, user story/test case is moved to "Done"
- Relevant documentation/diagrams produced, reviewed, and approved including product documentation (User Guide, Help, etc.), architectural documentation, and any other as required by the client
- Any build/deployment/configuration changes implemented/documented/communicated

Each element of an agile project can be subject to a DOD, including sprint planning sprints, user stories, sprint demos, retrospectives, and backlog grooming. The DOD defines the tasks and work items required to complete each element.

CMMI LEAD APPRAISER PERSPECTIVE: DEFINITION OF DONE

The DOD is a powerful tool that, when implemented properly, can provide valuable information while attempting to verify practices in Requirements Management (SP1.1 in particular), Project Monitoring and Control, Validation, Product Integration, and Technical Solution. The documented description of various DODs can provide verification for practices in Organizational Process Definition and Integrated Project Management.

Project Planning (PP)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTIMATE THE SCOPE OF THE PROJECT	Identifying all relevant components of project scope	The collective "DODs" for each element make up a virtual WBS. Therefore, DODs must be complete to meet the needs of the project team to understand exactly what is included in the user story and with sufficient detail to assign story points to the actions during the sprint planning meeting.	Standard DODs documented in a WIKI; attached to a user story or epic; identified with a story point value in a sprint plan
ESTABLISH ESTIMATES OF WORK PRODUCTS AND TASK ATTRIBUTES	Providing guidance on package of work products to be estimated	The DOD is defined during sprint planning, and the effort for each is considered in the overall story point value for the sprint; must be coupled with agile estimating technique such as Planning Poker	Estimate reflecting tasks in DOD; sprint plan demonstrating consideration of DOD

Validation (VAL) and Verification (VER)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
PERFORM VALIDATION/ VERIFICATION	Providing guidelines for validation and verification	The DOD is defined in sufficient detail to verify the completion of each related work product or validate each user story	Tasks completion indicator from an information radiator that reflects DOD for a particular work product; work remaining is reduced to zero

Epics

SUMMARY

In agile terms, an "epic" is simply a large user story that will require later breakdown into individual user stories. Typically, a team will choose to break down an epic into smaller, more easily manageable user stories that can be completed in a single sprint.

Sprint allocation is not the only reason to break down an epic. Another consideration is clarity of functionality. Customers often request features that are complex, and need to be broken down into smaller components to be understood and implemented. Another consideration is "who will do the work?" If a user story requires more than one Scrum team to complete the work, the story would be considered an epic, as this story will need to be divided into at least two user stories, one for each Scrum team.

Epics are often broken down during backlog grooming, or at times during release or product planning, or on an ongoing basis by the product owner.

The break-down process is often iterative. A product owner can request a feature and present it to the team as a user story. The team may choose to refer to this as an epic, and break it down further. Agile practitioners have no standard measurement for what makes a user story an epic, although it is commonly understood that an epic can span multiple sprints, but a user story should not.

USING CMMI TO STRENGTHEN EPICS/ REQUIREMENTS DEVELOPMENT

Epics are large user stories that require further refinement prior to assignment to a sprint backlog. Uncertainty about requirements, especially early in a project, drives the need to document what is known at a high level, and leads to the creation of epics in an agile project. But how can stakeholders who are responsible for the definition of requirements take a structured approach? Agile does not mean anarchy, so looking to the CMMI for guidance is common sense.

The CMMI specific goals and practices in the Requirements Development process area can give clarity to the process of creating epics and user stories, and can help bring robustness to the product backlog. Some examples are:

- Making sure that all possible sources of needs for the product, not just needs from an external customer, are considered in the creation of the product backlog. Missing an important need early in a project can have a large negative future impact.
- Making sure that the relevant stakeholders are involved in the process of creating and reviewing the epics and user stories to mitigate potential misunder-standings and rework in the sprints.
- Making sure that the product backlog items driving sprint planning have been analyzed and validated. How a backlog item is tested or verified should be clear.

Agile teams thrive when feedback loops are short and learning occurs. The informative components of Requirements Development are excellent sources for learning about ways to make sure the epics and user stories in the product backlog are complete and robust.

One of the goals of an agile project is to be able to respond quickly and efficiently to the changes that inevitably occur throughout the life of a project. Project timing changes, team members change, information needs change, but the most disruptive changes are requirements changes.

Scrum is a lightweight project management framework that leaves the details of implementation up to the team that adopts it. The Requirements Management process area in CMMI offers many best practices that can help an agile team make sure important items are not missed in dealing with the management of requirements via a product backlog. Some examples follow:

- Making sure epics and user stories are clear through the application of acceptance criteria and open communication channels between requirements stakeholders. Agile expects transparency, and the CMMI reinforces this need, especially around requirements.
- Reinforcing the idea that commitment by the team doing the work is a critical part of product development. CMMI clarifies that commitment is also important for requirements changes, something that is often missed.
- Setting the expectation that traceability is needed for requirements, no matter what form they take. In an agile project, accomplishing good traceability can be difficult, especially if purely visual management methods are being used.

Agile teams thrive when feedback loops are short and learning occurs. The informative components of Requirements Management are excellent sources for learning about ways to make sure that the inevitable changes to epics and user stories in the product backlog are robustly managed.

CMMI LEAD APPRAISER PERSPECTIVE: EPICS

An epic is a work product that is created as a result of a requirements elicitation and development process, and therefore has close ties to RD SP1.1 and RD SP1.2. Epics can be created in a conference room pilot (CRP), brainstorming session, or via other requirements elicitation processes. A product owner can also present the agile team with a completed set of user stories, absent a single epic. In that case, the representation of the related practice will have a different context, and will reside in the section titled "User Stories." In another credible scenario, RD SP1.1 and RD SP1.2 could be found in both the development of epics and user stories. This is less redundant and is a reflection of the iterative and incremental nature of agile epic and user story development.

Because epics must map to their subordinate user stories, this forms the first level of requirements traceability … needs to epic to associated stories. Therefore, REQM SP1.4 directly maps to this activity. In addition, when you take into account that defining and understanding requirements is an important aspect of this work, all of Requirements Management will likely apply.

Requirements Development (RD)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ELICIT NEEDS	This practice ensures epics are captured early in the product lifecycle (e.g., during the creation of the product roadmap, during release planning, from focus groups or teams) in concert with all key stakeholders.	The Product Owner presents high-level features or requirements ("needs") to the agile team with the expectation, or requirement, that they be broken down into User Stories (see "Backlog Grooming")	A User story Card on a Scrum Board; A User story in an appropriate tool (SharePoint, Jura, etc.)
TRANSFORM STAKEHOLDER NEEDS INTO CUSTOMER REQUIREMENTS	The epic is often used to represent the business need, and this practice supports the evolution from a traditional requirements specification to the creation of epics and stories that can be loaded in a tool or placed on a team wall for all to see	The Scrum team needs to break the Epic into a smaller user story, and does so during Backlog Grooming	A user story card on a Scrum Board or backlog list; A user story in an appropriate tool (SharePoint, Jura, etc.)

Requirements Management (REQM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
UNDERSTAND REQUIREMENTS	This practice focuses on ensuring that the business requirements and functions required are clearly stated to enable creation of user stories	The product owner develops the epics and provides clarity of them to the Scrum team	Epic in the product backlog (on a Scrum board or in a tool); attend a backlog grooming session to see the epics and how they are used to develop user stories
OBTAIN COMMITMENT TO REQUIRE-MENTS	This practice focuses on ensuring that the business requirements and functions required are clearly understood to support creation of user stories that are meaningful to the business and implementable by the teams	Epics are discussed until they are clearly understood by the team resulting in team buy-in to the requirement (epic)	Epic in the product backlog (on a Scrum board or in a tool); attend a backlog grooming session to see the epics and how they are used to develop user stories
MANAGE REQUIREMENTS CHANGES	This practice provides expectations related to managing the backlog of epics and ensuring that processes are in place to evaluate proposed changes and to track them over time. This helps to reduce chaos for teams where POs constantly adjust work expectations without regard for impact to the team.	Epics are modified, added, and deleted from the product backlog by the product owner, and changes are discussed with the Scrum team	Epic in the product backlog (on a Scrum board or in a tool); attend a backlog grooming session to see the epics and how they are used to develop user stories
MAINTAIN BI-DIRECTIONAL TRACEABILITY OF REQUIRE-MENTS	This practice emphasizes the value of under-standing how epics are implemented via user stories/tests and helps the team to ensure that appropriate epics are adequately addressed prior to deployment	Epics are traced to higher-level epics and/or associated user stories in a way that ensures that the product owner can state with confidence that the set of user stories will meet the acceptance criteria identified for the epic. There is clear traceability between original items on the product backlog, epics, user stories, and tests identified as part of the "Definition of Done."	Epic in the product backlog (on a Scrum board or in a tool); user story mapped to the associate epic and higher-level user stories on a Scrum board or in a tool; attend a backlog grooming session to see the epics and how they are used to develop user stories
MANAGE THE PROJECT USING INTEGRATED PLANS	Epics impact delivery of the overall product roadmap and prioritization/timing of what is delivered. They are the highest-level require-ment used to scope the work.	The breakdown of an epic may result in the need to categorize additional work. This should be added to the product backlog for prioritization (grooming) during the backlog grooming activity.	User story on a Scrum board or in the backlog list; user story in an appropriate tool

Team Estimating Game & Planning Poker

SUMMARY

The Team Estimating Game (sometimes called the "Fibonacci Game") is an agile estimation technique that establishes relative sizing using story points and rough order of magnitude estimation.

Planning Poker is a similar technique that uses playing cards to depict story points.

The Team Estimating Game uses a set of cards or sticky notes with user stories that are placed on a board. The game consists of three rounds of collaborative sizing negotiation by the agile team. During each round, a team member is allowed to change the relative sizing of one of the user stories on the board. After each round, the team discusses the sequence of stories on the board. Once relative sizing is determined, the board is reorganized, again through three rounds, to assign story points to each user story based on input from all members of the agile team. The Team Estimating Game uses the Fibonacci sequence of numbers as its basis for relative sizing.

USING CMMI TO STRENGTHEN THE TEAM ESTIMATING GAME

Good estimates are an elusive artifact in many projects. This agile technique tries to solve the estimation problem by using an estimating game to size backlog items relative to one another. The relative size of a backlog item is intentionally disconnected from effort in hours to force the team to think in a different way. "Roughly right" versus "accurately wrong" is the ultimate goal of the team estimation game, and the Project Planning process area of CMMI can help bring additional clarity and robustness to estimates. Some examples are:

- Strengthening the need to make sure the complete scope of the project is understood as early as possible. Sufficient detail is needed so that relative sizing can be done for each item whether it is an epic or user story. Completeness is the goal.
- Setting the expectation that all types of work products need to be considered in the estimation. Software is only one work product; there are many more.
- Making sure that effort estimates are the final outcome of planning. Effort allows the agile team to measure its velocity and improve future estimates.

Many more examples and suggestions are found in the informative components of the Project Planning process area.

In the Team Estimating Game, the Scrum team, working with a Scrum Master and the product owner, attempts to size each epic or user story as part of product backlog development or sprint planning.

The Team Estimating Game includes an in-depth discussion of the functionality of each user story by team members with a diverse set of skills.

Clear epics and user Stories minimize the need for the product owner to spend time clarifying product backlog items, and improves the ability of the team to "get it right the first time" during a sprint. So how can the CMMI help? SP1.1 in the Requirements Management process area provides some guidance. For example:

- Sources of requirements should be clearly identified to minimize confusion and possible scope creep.
- The definition of what makes a good requirement should be documented, understood, and used by the team.
- Communication channels should be transparent and open during the creation and review of requirements.

CMMI LEAD APPRAISER PERSPECTIVE: TEAM ESTIMATING GAME & PLANNING POKER

The Team Estimating Game primarily reflects use of Project Planning SP1.1 when placed within the context of the product backlog, but if the user stories are fleshed out with tasks and a completed Definition of Done, could provide verification of Project Planning SP1.2 and SP1.4. Because the Team Estimating Game is a collaborative event, it is recommended that the appraisal team attend a working session to observe how the game is played.

Project Planning (PP)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTIMATE THE SCOPE OF THE PROJECT	The practice makes sure the complete breadth of the project is captured through epics and user stories early in the project lifecycle. Detailed estimates are not as critical as making sure the scope is complete.	The user stories are sufficiently defined, and an appropriately diverse set of skills is represented during the session	Observe a Fibonacci Game; white board with sticky notes or cards representing tasks that have been sequenced; points are recorded on the user story card
ESTIMATE WORK PRODUCTS	The practice makes sure clarity exists for all items produced by the project. Expectations are clear.	The Definition of Done has been sufficiently determined so that there is enough detail to estimate effort	Observe a game being played; points recorded on the user story cards; DOD describes Fibonacci Game as an estimation method
ESTIMATE EFFORT	The practice makes sure the sprint planning captures all activities that must be complete prior to calling a user story done.	Once the effort estimate has been completed, costs can be created based on the resource skill level required and the work effort needed.	Updated project cost estimates

Requirements Management (REQM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
UNDERSTAND REQUIREMENTS	The practice makes sure that discipline is applied to the activity of creating or clarifying epics or user stories. Discipline in this area optimizes and increases the efficiency of backlog development and sprint planning.	The product owner and Scrum team clarify the requirements during one of the three rounds of the Team Estimating Game	Updated user stories; user stories sequenced by priority or complexity

Pair Programming

SUMMARY

Pair Programming is an engineering technique in which two software developers work together to accomplish a coding task. They generally work at one workstation with one programmer being "the driver" and the other being a "navigator" who reviews code as it is being entered by the driver. While this usually increases the initial cost of programing, it more than pays for itself in increased code quality. In some scenarios a developer codes, while the other unit tests. In other cases a developer codes while another sits next to the developer, helping to interpret the requirements and ensuring a higher level of code quality.

Pair programming has several benefits. These include:

- Having one of the partners develop unit test cases and using them to "test" the code of the other partner in real-time
- Refactoring to improve the quality and maintainability of the code
- Improving knowledge of the code & structure because multiple developers will have intimate familiarity
- Having one partner serve as "navigator" to assist the other in writing efficient code
- Conducting code review in real time, and aiding in the compliance of coding standards and practices

USING CMMI TO STRENGTHEN PAIR PROGRAMMING / VERIFICATION

Pairing capitalizes on the idea that "two minds are better than one." One of the reasons an agile approach works is that short feedback loops are built into many of the practices. Pairing is just one of many practices that can be applied. Regardless of how the pairing is done, the direct and immediate feedback loop of working together makes verification simple and efficient. However, the CMMI can bring clarity and robustness to the pairing practice by expanding the concept of verification. For example:

- Work products can be more than just the code produced by the pair. User stories often result in other work products such as training material, specifications, or user instructions.
- For efficient sprints, inputs to peer reviews such as tools and methods for reporting defects should be identified and available. Each individual in a pair should be trained and capable to do the review.
- Completion of planned peer reviews is a critical success factor in the delivery of a quality product. Peer reviews should not be skipped.
- Information learned during the peer reviews should be captured and fed into the sprint retrospective. Some examples are tools that worked well, tools that didn't work well, a process step that was missing, or a problem with any of the inputs to the peer review.

Many more examples and suggestions are found in the informative components of the Verification process area.

CMMI LEAD APPRAISER PERSPECTIVE: PAIR PROGRAMMING

Pair Programming is intended to increase code quality independent of requirements, and to help ensure that the code meets the intended requirement, so it connects in part to both Validation (VAL) and Verification (VER), although not the complete inventory of practices in each process area. However, the fact that it is focused on code quality means that Technical Solution (TS) is another process area that could assist pair programmers. Because of the collaborative and iterative nature of this technique, Technical Solution GP3.2 is also implied.

Verification (VER)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
PLAN FOR PEER REVIEWS	The practice makes sure that an approach is considered that is more than just the code being produced. Pairing can benefit other work products. The practice also makes sure that all the supporting methods and tools are in place for the pair.	The Driver and Navigator routinely work on code together, are assigned to work together, and have both reviewed and understand the requirements and coding principles	Requirements, user stories, team charter, and pair programming process
CONDUCT THE PEER REVIEWS	The practice makes sure that peer reviews are done as part of pairing	The Driver and Navigator routinely work on code together, and follow a structured Pair Programming Process	Code comments; defect reports
ANALYZE PEER REVIEW RESULTS	The practice makes sure that data from the peer reviews is captured and used to improve future reviews. Feedback from learning is fed into a retrospective so the rest of the team learns and improves.	The Driver and Navigator routinely work on code together, defects are captured and analyzed, and a retrospective is held at the end of the sprint that covers their pair programming activities and analysis of the data	Retrospective notes; defect reports

USING CMMI TO STRENGTHEN PAIR PROGRAMMING / VALIDATION

One of the reasons an agile approach works is that short feedback loops are built into many of the practices. Pairing is just one of many practices that can be applied, and the idea for Pairing is that "two minds are better than one." So how can the CMMI Validation process area be used to make pair programming more robust? The answer lies in several of the validation practices described. Some examples are:

- Putting thought into how validation is done on the software. If pairing is the method, developing and validating the software in the intended environment is key, and one idea could be to have one of the individuals represent the end user.
- Setting the expectation that all necessary equipment is identified and available for use during the paired software development.
- Making sure each individual in the pair is trained on the methods and the tools used to do the software validation. Pairing can be an efficient way to transfer knowledge from a more experience individual to a less experienced individual.

Many more examples and suggestions are found in the informative components of the Validation process area.

Validation (VAL)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
SELECT PRODUCTS FOR VALIDATION	The practice sets the expectation that what will be validated is identified. In the case of pairing, the most common item is the software code.	Pair programming is a selected process to be performed, and all code is to be composed using pair programming, and this technique is used to also address code quality, unit testing, and standards	Standard processes define pair programming as the standard
PLAN THE VALIDATION ENVIRONMENT	The practice sets the expectation that everything needed to do the planned validation is identified and made available	Workstations are assembled for pair programming with two seats, appropriate desk, and multiple monitors. In a virtual environment, screen sharing is available and used.	Visually inspect pair programming environment; PPQA reports may indicate pair programming environment; training indicates the environment is available
PLAN THE VALIDATION METHOD	The practice sets the expectation that a method for the validation is selected and made available for the team. The pair must be trained if needed.	Scrum or XP is selected as the standard, and pair programming is defined in the organizational policy / expectations	Training material or policy materials; WIKI or PAL

USING CMMI TO STRENGTHEN PAIR PROGRAMMING / TECHNICAL SOLUTION

An organization that uses agile development must use sound technical practices to deliver a quality product. Agile does not mean anarchy. A good definition of done should contain, at a minimum, common criteria that are applied to all user stories. The benefit of working in sprints is that the common definition of done criteria are met during the sprint duration, and not put off until the end of a long development phase: after the fact usually means never. The Technical Solution process area of the CMMI requires sound technical decisions and practices be used in the development of a product. Some examples are:

- The method of programming should be identified; examples include object oriented, Test Driven Development, or automatically generated code
- The reuse strategy should be clear
- The coding standard that applies to the development
- Any code complexity metrics that must be met
- Whether unit tests will be done, and how they will be done
- The use of static or dynamic code analysis tools, and the requirements for resolving associated warnings
- How defects will be reported and managed

Generic Practice 3.2 can also help to make the pairing process more robust through the expectation that improvements found in satisfying the done criteria are brought to the sprint retrospective meeting, and ultimately shared beyond the originating team. Agile is institutionalized learning, and the CMMI fits well here.

Many more examples and suggestions are found in the informative components of the Technical Solution process area.

Technical Solution (TS)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
IMPLEMENT THE DESIGNS	The practice makes sure that all planned design objectives are identified and implemented.	Navigators are tasked with ensuring that coding standards (complexity, naming conventions, variable naming, intra-application interfaces) are followed	Code; code comments; defect reports
COLLECT PROCESS RELATED EXPERIENCES FOR FUTURE IMPROVEMENTS	The practice makes sure a mechanism is in place to collect good practices and improvements, and implement them in the future.	Navigators provide improvements for code and process	Lessons learned; commentary at daily standup; "improvement of the week" at daily standup or retrospective

Product Backlog

SUMMARY

The product backlog is a prioritized list of everything that may be included in the product. It includes features, bugs (if the product is in production or if there have been releases of the product in prior sprints), documentation changes, and any other tasks required by the product owner. The product owner owns the backlog, the tasks on the backlog, and the priority of the tasks on the backlog. The owner may use input from stakeholders, business representatives, and the Scrum team to help set the priority, but the owner is responsible. The backlog is maintained during the sprints through the backlog grooming process.

USING CMMI TO STRENGTHEN PRODUCT BACKLOG / REQUIREMENTS MANAGEMENT

The product backlog is an ordered list of everything that might be needed in the product, and is the single source of requirements for any changes to be made to the product. The product backlog is the foundation of an agile project regardless of the agile practices being used: the better the backlog items, the better the project outcome. The Requirements Management process area in CMMI offers many best practices that can help an agile team make sure important items are not missed in the creation and refinement of the product backlog. Some examples are:

- Making sure epics and user stories are clear through the application of acceptance criteria and open communication channels between requirements stakeholders. Agile expects transparency, and the CMMI reinforces this need, especially around requirements.
- Reinforcing the idea that commitment by the team doing the work is a critical part of product development. This commitment ultimately comes during sprint planning meetings, and is enhanced by high-quality backlog items.
- Reinforcing the need to manage changes to the product backlog items over time. The CMMI emphasizes the need to have stakeholders involved in the analysis and decisions about changes, and this is something that can easily be missed.

The informative components of Requirements Management are excellent sources for information about ways to make sure that the product backlog items are robust.

CMMI LEAD APPRAISER PERSPECTIVE: PRODUCT BACKLOG

The establishment of the contents of the product backlog, user stories, initial estimates, and priority demonstrates that the product owner understands and is committed to the requirements of the product. This data also provides the foundation for the development of project plans and the integration of related plans. Because the product backlog is modified during the backlog grooming process, it becomes a living document throughout the life of the project.

Requirements Management (REQM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
UNDERSTAND REQUIREMENTS	This practice focuses on ensuring that the business requirements and functions required are clearly stated to enable creation of user stories	The product owner develops the epics and user stories and provides clarity of them to the Scrum team	Epic in the product backlog (on a Scrum board or in a tool); attend a backlog grooming session to see the epics and how they are used to develop user stories
OBTAIN COMMITMENT TO REQUIREMENTS	This practice focuses on ensuring that the business requirements and functions required are clearly understood to support creation of epics and user stories that are meaningful to the business and implementable by the teams	Epics and user stories are discussed until they are clearly understood by the team resulting in team buy-in to the requirement (epic)	Epic in the product backlog (on a Scrum board or in a tool); attend a backlog grooming session to see the epics and how they are used to develop user stories
MANAGE REQUIREMENTS CHANGES	This practice provides expectations related to managing the backlog of epics and user stories, and ensuring there are processes in place to evaluate proposed changes and to track them over time. This helps to reduce chaos for teams where POs constantly adjust work expectations without regard for impact to the team.	Epics and user stories are modified, added, and deleted from the product backlog by the product owner, and changes are discussed with the Scrum team	Epic in the product backlog (on a Scrum board or in a tool); attend a backlog grooming session to see the epics and how they are used to develop user stories

USING CMMI TO STRENGTHEN PRODUCT BACKLOG / INTEGRATED PROJECT MANAGEMENT

The product backlog is a prioritized list of everything that is needed in the product. Large projects with extensive product backlogs may require multiple teams working in synchronized sprints to deliver the final product, and the coordination of multiple teams creates project management complexity that can have a potential negative impact on cost, quality, and timing. Because agile frameworks like Scrum are lightweight and leave the implementation details up to the users, there is not much guidance for how to manage this situation. Fortunately, the Integrated Project Management process area in the CMMI can help. Some examples are:

- Identification of other plans that may need to be captured in the product backlog for large projects. Training, logistics, facilities, configuration management, and integration are a few.
- The need to consider how to gather consistent and useful improvement information, and share it across multiple agile teams. Lessons learned from sprint retrospectives are an example.
- A backlog prioritization method that accounts for critical path management across the multiple agile teams. The goal is to understand and account for interdependencies.

The management of complex projects is always a challenge and a source of risk. The informative components of Integrated Project Management are excellent sources for information about ways to make sure that multiple agile teams are managed efficiently.

Integrated Project Management (IPM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
INTEGRATE PLANS	The practice provides useful information for product owners of large projects where coordination between multiple agile teams is necessary (i.e. Scrum of Scrums)	The product owner coordinates with other product owners where products have interdependencies	A product backlog with priorities set on user stories that connect to other project requirements

USING CMMI TO STRENGTHEN PRODUCT BACKLOG / PROJECT PLANNING

The product backlog contains all the epics and user stories needed to deliver a complete solution to the internal or external customer of the product. The focus on delivering functional software at the end of each sprint is the most obvious application of agile practices; however, many items go into the delivery of the software. All of these should be part of the product backlog to make sure the estimates are complete and commitments are met by the project. So what kind of content should be addressed by a thorough product backlog? The CMMI Project Planning process area offers answers to the question. Some examples are:

- Key customer or internal project milestones. Delivery of prototypes to the customer and information required for a gate or phase exit should be considered. The support functions interfacing with the agile team need to have confidence that the high-level milestones are understood.
- Project members who require training. Product backlog items could exist that cover the training details such as method of training, the timing constraints, and the trainer. An experienced agile team member might have a sprint backlog item to train less experienced team members during a sprint.

The creation of a complete project plan is a key input to the success of any project. The informative components of the Project Planning are excellent sources for information about ways to make sure that critical items are not missed in the product backlog.

Project Planning (PP)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH THE PROJECT PLAN	The practice gives guidance for the content of a robust project plan. The product owner must make sure that the product backlog is complete so the team knows the timing and deliverables.	The product owner and Scrum team collaborate during release planning, sprint planning, and backlog grooming process to determine initial and on-going priorities for user stories, and provide estimates in the form of story points and/or effort	A product backlog with user stories in priority sequence and work efforts attached to each story

Refactoring

SUMMARY

Refactoring is a technique, commonly practiced in agile environments, which is focused on improving the internal structure, style, operations, and condition of software code while maintaining its external functionality and behavior. Refactoring often means rewriting code, fixing bugs, or improving observable or functional aspects of the code. The intent of refactoring is to improve the maintainability of code by improving the attributes (length, duplication, and complexity), provide better understanding, and providing the benefit of having developers implement better design decisions.

Refactoring is a form of code improvement that can be focused equally on code quality, requirements quality, testing, integration, or any other process-based deliverable that falls within the development lifecycle.

USING CMMI TO STRENGTHEN REFACTORING / INTEGRATED PRODUCT MANAGEMENT

One of the benefits of agile is that learning is built in and occurs through many closed feedback loops. The feedback is done at the point of the work, and is not delayed until late in the project when time does not allow for big bang learning. Lean engineering teaches us that any word that begins with the letters "re" is probably generating waste. However, refactoring is one of the closed feedback loops used by agile teams to continuously improve software. Refactoring, which is a good practice in agile, can also be applied to improve many other artifacts. The Integrated Project Management process area offers guidance that
can be used to expand the benefits of refactoring. Some examples are:

- Identifying best practice examples of refactored code, improved user stories, or accurate story sizing outcomes
- Setting a goal to share the knowledge beyond a single sprint retrospective by capturing the knowledge and creating training opportunities for other teams
- Submitting the knowledge gained back to the support organization responsible for the definition, deployment, and support of documented agile practices

The informative components of Integrated Project Management are excellent sources for information about ways to enhance refactoring in an agile project.

CMMI LEAD APPRAISER PERSPECTIVE: REFACTORING

Refactoring, being focused on continuously improving the products, has a relationship to IPM SP1.7, GP3.2, and TS SP3.1.

Integrated Project Management (IPM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
CONTRIBUTE TO ORGANIZATIONAL PROCESS ASSETS	The practice gives guidance for the types of learning that should be shared throughout the organization. Refactoring is a potential source of learning.	Assuming common refactoring is documented and shared beyond the core team	Contributions to lessons learned repository; inclusion of refactoring results and examples in training and mentoring materials

USING CMMI TO STRENGTHEN REFACTORING / TECHNICAL SOLUTION

Refactoring is one of the closed feedback loops used by agile teams to continuously improve software. Knowledge gained from other teams can be applied to improve implementation practices across a wide range of topics. The Technical Solution process area offers guidance that can be used to expand the benefits of refactoring in the implantation of the design. Some examples are:

- Using proven configurations of tools in the implementation of the code; static analysis tools, dynamic analysis tools, unit test suites, and simulation tools are a few examples
- Using best practice examples of refactored code, improved user stories, or accurate story sizing outcomes in the implementation
- Making sure the team understands implementation best practices by taking advantage of training opportunities

The informative components of Technical Solution are excellent sources for information about ways to enhance refactoring in the implementation of a product.

Technical Solution (TS)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
IMPLEMENT THE DESIGN	The practice sets the expectation that the design is fully and correctly implemented. Prior learning generated by refactoring practices was captured, and is part of the implementation.	Inputs to OPD on how refactoring can be accomplished, and what methods can be used; documents provided as lessons learned	Assuming refactoring practices and methods are shared beyond the core team

Release Burn-Down Chart

SUMMARY

A Release Burn-Down chart is an information radiator that visually depicts the trajectory of the overall release by sprint. Based on the number of story points an agile team is able to "burn down" in each sprint (also known as "velocity"), the Release Burn chart helps the product owner and agile team to understand whether or not they will deliver the desired functionality in the required time frame as planned.

The source of the raw data is the product backlog. Each sprint that has been assigned to the team (multiple teams in a Scrum of Scrums environment) appear along the horizontal axis. The vertical axis indicates the total remaining story points at the start of each sprint.

Agile teams may use the Burn-Up chart. The Burn-Up chart is basically the same. The only difference is that instead of tracking how much backlog remains to be done, teams track how much they have completed, so the curve is going up, not down.

AGILE TEAM MEMBER PERSPECTIVE

Release burn chart is a mechanism to monitor the release implementation progress on a high level across the iterations. The iteration burn charts and daily standups track the detailed activities during the iteration.

To build harmony across the projects and organization experience, CMMI Process Areas such as Organizational Process Definition (OPD), Organizational Process Focus (OPF), and Measurement and Analysis (MA) help teams to define stranded templates for different charts, guidelines for use in different projects, historical data, and chart analysis guidelines. These templates are always enhanced by different project experiences.

Also, teams can share their experiences across different projects when the benefit from PMC GP 3.2 and Integrated Project Management.

Project Monitoring & Control

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
MONITOR PROJECT PLANNING PARAMETERS	The practice monitors actual values of Release planning parameters against the Release plan	The Release Plan is adjusted based on the data presented in the Release Burn-Down Chart (RBDC)	Updated Release Plan
MONITOR COMMITMENTS	The practice works to ensure that all commitments that were made in the Release Plan, Sprint Plan, or other relevant plans are monitored and tracked	Release scope is revised with customer based on the development performance presented in the RBDC. If the story points at the Y axis are sorted and grouped by themes that present a module or group of features, the chart will provide information about which themes will not be delivered as committed based on the burn chart data.	Updated release scope Updated planned work products Updated Release Plan
MONITOR STAKEHOLDER INVOLVEMENT	The practice ensures that appropriate interactions occur so that stakeholders are involved	A standard RBDC is used and reviewed with stakeholders at the Sprint Demo and Retrospective	Release Burn-Down Chart Documented causes and actions taken to improve and to mitigate release risks
CONDUCT PROGRESS REVIEWS	The practice expects that project's progress is periodically reviewed for progress, performance, and issues	Data from RBDC is reviewed and used to adjust the Release Plan and Sprint Plans during the Sprint Planning Meeting	Updated Release and Sprint Plans based on the data in the RBDC

Measurement & Analysis

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
SPECIFY MEASURES	The practice defines measures that will be used to address measurement objectives	An operational definition of measured, that meets organizational objectives, is used. Velocity based on story points should be used as the primary indicator of schedule and budget success	Release Burn-Down Chart
OBTAIN MEASUREMENT DATA	The practice collects actual measurement data	The data is recorded at the end of each sprint and displayed for all team members to view	Release Burn-Down Chart
ANALYZE MEASUREMENT DATA	The practice analyzes and interprets measurement data and generates results	Story points are being used as the primary indicator, and upper management does not expect more traditional metrics (e.g., Cost Performance Index (CPI) and Schedule Performance Index (SPI) instead of velocity)	Release Burn-Down Chart
COMMUNICATE RESULTS	The practice ensures that all measurement and analysis activities are communicated to relevant stakeholders	The RBDC is used as an information radiator and displayed for all team members to see	Release Burn-Down Chart

Organizational Process Definition

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH THE ORGANIZATION'S MEASUREMENT REPOSITORY	The practice focuses on establishing and maintaining measures in an Organizational Repository so that it can be accessed and used to plan future projects	Data is stored in a public area for all team members to view	Release Burn-Down Chart

Integrated Project Management

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH THE PROJECT'S DEFINED PROCESS	The practice ensures that agile teams can leverage the corporation's standard processes to clearly establish what will be executed for their unique team or project, including sprint length, Release duration, sprint planning, backlog grooming, retrospectives, and more.	Release burn chart templates and processes and guidelines have been documented in the corporation's standard processes	Corporation's set of standard processes with tailoring guidelines and criteria
USE ORGANIZATIONAL PROCESS ASSETS FOR PLANNING PROJECT ACTIVITIES	The practice uses the results of previous planning and execution activities as predictors of the relative scope and risk of the effort being estimated. The release burn chart helps the team understand previous performance.	Team uses one of the organization standard templates for release burn chart; when team has number of data points in the release burn chart, it can use these points for revising its estimates for the remaining release work and for further product releases	Selected and tailored release burn chart for the release; revised estimates based on velocity history

Release Planning

SUMMARY

The purpose of Release Planning is to commit to a plan for delivering an increment of product value (product increment release). In this stage, the team does not plan for each detailed activity. The team defers the detailed commitments such as determining who is going to do what and when, for the iteration planning.

Before getting started, Release Planning needs a ranked product backlog managed by the product owner. The product owner is responsible for transferring the product vision, market, and business objectives for the team and to acknowledge them if new product backlog items may be needed during the release implementation. The product owner transfers the business expectations for the release (fixed release date or fixed release scope).

During Release Planning, the product owner and the agile team collaborate on which features will be delivered by which date. It is, in most cases, the "master plan" that identifies system functionality by deadline. From the Release Plan, the agile team can plan for needed capabilities and team size, select iteration length, and estimate velocity and the expected number of sprints to meet the release deadline.

The team usually defines contingencies for the changes from the customer demos by adding a buffer iteration(s) for the release. However, the release plan is a living artifact. The product owner by collaboration with the agile team and the business side (customers) may revise the release plan when they receive feedback from demos.

The Release Plan will often prioritize the features so that the agile team can focus on the highest-value features in the early sprints, leaving the low-value features until later, or even to be negotiated away if they are deemed to have low value to the business after the team learns more about them.

Project Planning

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTIMATE THE SCOPE OF THE PROJECT	The practice ensures that the project's tasks are identified in enough detail to understand the scope of the project	Functionality is defined to be delivered during a particular sprint or set of sprints over the duration of a release	Release plan with defined release scope
ESTABLISH ESTIMATES OF WORK PRODUCT AND TASK ATTRIBUTES	The practice expects that an appropriate estimation method will be used when estimating a project. Attributes of work products and tasks are needed for estimation.	Agile estimating processes and guidelines have been documented in the corporation's standard processes	The organization encourages the use of agile estimation techniques based on story points; user stories include a robust "Definition of Ready" that includes tasks as well as work products
DEFINE PROJECT LIFECYCLE PHASES	The practice defines project lifecycle phases that the project will follow. Understanding the lifecycle is crucial in planning the effort and timing of a project.	Selected sprint length, planning for iteration "0" and stabilization iteration if needed; DOD for release/iteration/story describe the release lifecycle; expected internal and customer demos are determined	Release plan with number and length of sprints defined; DOD and planned demos
ESTIMATE EFFORT AND COST	The practice estimates the project's effort and cost for work products and tasks based on estimation rationale	The planned velocity, number of sprints, and resource requirements are determined; an estimate of the project cost can be determined	Release plan with number and length of sprints and team size defined
ESTABLISH THE BUDGET AND SCHEDULE	The practice establishes a project's budget and schedule	Determined release dates and customer demos date, release total size, allocating budget for required human/nonhuman resources	Release high-level plan with dates for release and planned demos; release size and budget
IDENTIFY PROJECT RISKS	The practice expects that all project risks will be identified and analyzed during project planning	Adding iteration(s) as contingency for requirements ambiguity and uncertainty or any other reason defined by the team	Planned buffer for contingency; defined and planned time box for spikes; documented business feedback

Project Planning, con't

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
PLAN STAKE-HOLDER INVOLVEMENT	The practice gives guidance to make sure that all relevant stakeholders are identified, and that their responsibilities are clear. This concept is also important for a team using agile development practices.	Assuming team allocation for the release with needed skills and buffer, the release plan identifies the relevant stakeholders and indicates what agile ceremonies they will need to participate in (e.g., Sprint Planning, Backlog Grooming, Sprint Demo, etc.)	Release Plan identifying the stakeholders and ceremonies; team allocation with specified skills and roles, team velocity
ESTABLISH THE PROJECT PLAN	The practice gives guidance for the content of a robust project plan. This includes tying all aspects of the project together.	Assuming that the Scrum Master, agile team, and product owner worked together and ensured that the committed release scope and date can be achieved by the planned team velocity	Release date, release total size, and team velocity
OBTAIN PLAN COMMITMENT	The practice obtains commitment from relevant stakeholders responsible for performing and supporting plan execution. This would include the agile team and product owner in this case.	Release plan was created by agile team with presence of product owner	Evidences for agile team involvement in release planning

Measurement & Analysis

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
SPECIFY MEASURES	The practice specifies measures to address measurement objectives. All measures should include enough detail so they are precise, quantifiable measures.	Release goals such as deliverables, defect rates, code quality, team performance, etc. are defined with described measurable targets	Release goals and related measures

Integrated Project Management

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH THE PROJECT'S DEFINED PROCESS	The practice establishes and maintains a project's defined process from project startup through the life of the project	Scrum, sprints, and releases are all defined as organizational assets	Definition of the process; standard duration, or guidance, for sprint length and frequency

Requirements Management

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
OBTAIN COMMITMENT TO REQUIREMENTS	The practice obtains commitment to requirements from project participants	Product owner and agile team agree to the functionality to be delivered at each sprint (other practices required)	Clear definition of what functionality is to be delivered during which sprint
MANAGE REQUIREMENTS CHANGES	The practice manages changes to requirements as they evolve during a project	The release plan is iteratively and incrementally updated as the sprints progress	Updated release plans
MAINTAIN BIDIRECTIONAL TRACEABILITY OF REQUIREMENTS	The practice ensures that bidirectional traceability exists among requirements and work products. A Release Plan that includes how requirements are linked to sprints and code modules supports this.	Functionality is allocated to appropriate sprints and code modules	Release plan that allocated requirements across sprints and code modules

Sprint / Iteration

SUMMARY

The primary construct for getting work done with an agile team is the fixed-length Sprint. A Sprint is a time-boxed event where team members self-subscribe to User Stories and all work, including design, coding, testing, and backlog grooming takes place. Scrum teams vary on the inclusion of Sprint Planning, Sprint Demo, and Sprint Retrospective within the confines of a Sprint, with most reserving time inside the timebox to accomplish those tasks.

AGILE TEAM MEMBER PERSPECTIVE

Sprint duration, along with decisions about where to place the events described above, can be defined through the guidance and support of the CMMI's Organizational Process Definition. Data used to determine the optimum number of Sprints in a release cycle can be gleaned from information gathered and analyzed with the assistance of the Measurement and Analysis (MA).

Sprints should be carefully planned to ensure that Sprint commitments will be met, and that the end of the Sprint will results in useful and potentially shippable product. The CMMI's Project Planning and Project Monitoring and Control Process Areas provide comprehensive guidance as to the planning of projects that contain many sprints amongst multiple releases.

CMMI LEAD APPRAISER PERSPECTIVE: SPRINT/ ITERATION

Sprints are the primary construct used to allocate work in an agile environment. They are closely tied to Project Planning Define Project Lifecycle Phases (SP1.3) when coupled with a release schedule that contains multiple sprints.

Appraisal teams should verify that agile teams are meeting sprint commitments, tracking velocity, and not extending the length of sprints to complete the projected work.

Project Planning

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
DEFINE PROJECT LIFE-CYCLE PHASES	Assisting in the formalization and consistency of sprint duration and content	Release plans are defined that structure sprints within it with defined duration and content	Release Plan, Sprint Plan

Organizational Process Definition

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Some Ways You Might See It Demonstrated:
ESTABLISH LIFECYCLE MODEL DESCRIPTIONS	Defining the duration, content, and management of sprints by the organization, especially in a "scrum-of-scrum" environment	Organizational Process Assets maintain a clear definition of spring content and duration	An organizational process asset

Sprint Burn-Down Chart

SUMMARY

A Sprint Burn-Down chart is an information radiator that visually depicts the effort remaining to complete the sprint. The Sprint Burn-Down chart helps the agile team to understand whether or not it will deliver the desired functionality by the end of the sprint as planned.

The Burn-Up chart is basically the same. The only difference is that instead of tracking how much work is left to be done, we track how much work we've completed, so the curve is going up, not down.

AGILE TEAM MEMBER PERSPECTIVE

The use of burn charts in sprint tracking is to monitor teamwork progress and to manage the risks of achieving the iteration goal to deliver potentially shippable product.

The team can benefit from the CMMI Measurement and Analysis (MA) process area to enhance its capabilities for analyzing the burn chart. It can use MA SP 1.4 and SP 2.2 to define and perform the measurement analysis.

Applying Integrated Project Management (IPM) and Organizational Process Definition (OPD) helps the team to share problems, solutions, and improvements—experiences that are gained from iteration burn-chart tracking.

Measurement & Analysis

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
SPECIFY MEASURES	The practice defines measures that will be used to address measurement objectives	Story points, actual effort, and effort remaining are being used as the primary indicator of schedule and budget success	Sprint Burn-Down Chart
OBTAIN MEASUREMENT DATA	The practice collects actual measurement data	The data is recorded at the end of each sprint and displayed for all team members to view; also used for predicting velocity for the next sprint	Sprint Burn-Down Chart
ANALYZE MEASUREMENT DATA	The practice analyzes and interprets measurement data and generates results	Story points and effort remaining are being used as the primary indicator, and upper management does not expect more traditional metrics (e.g., CPI and SPI) instead of velocity	Sprint Burn-Down Chart
COMMUNICATE RESULTS	The practice ensures that all measurement and analysis activities are communicated to relevant stakeholders	The SBDC is used as an information radiator and displayed for all team members to see	Sprint Burn-Down Chart

Project Monitoring & Control

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
MONITOR STAKEHOLDER INVOLVEMENT	The practice ensures that appropriate interactions occur so that stakeholders are involved	A standard SBDC is used and reviewed with stakeholders at the Sprint Demo and Retrospective	Sprint Burn-Down Chart
CONDUCT PROGRESS REVIEWS	The practice expects that project's progress is periodically reviewed for progress, performance, and issues	Data from SBDC is reviewed and used to adjust the Release Plan and Sprint Plans during the Sprint Planning Meeting	Renegotiated Release and Sprint Plans based on the data in the SBDC

Integrated Project Management

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH THE PROJECT'S DEFINED PROCESS	The practice ensures that agile teams can leverage the corporation's standard processes to clearly establish what will be executed for their unique team or project, including sprint length, Release duration, sprint planning, backlog grooming, retrospectives, and more.	Sprint Burn Chart templates and processes as well as guidelines have been documented in the corporation's standard processes	Corporation's set of standard processes with tailoring guidelines and criteria
USE ORGANIZATIONAL PROCESS ASSETS FOR PLANNING ACTIVITIES	The practice uses the results of previous planning and execution activities as predictors of the relative scope and risk of the effort being estimated. The release burn chart helps the team understand previous performance.	Velocity indicated in the SBDC provides data that is used to adjust future planning (backlog grooming) activities	Sprint Burn-Down Chart Product Backlog

Sprint Demo / Sprint Review

SUMMARY

A Sprint Demo (Sprint Review) is an iterative and incremental collaborative technique to ensure that all stakeholders are aware of the value that is being delivered at the end of each sprint by an agile team.

During the Sprint Review, the work that was completed, and the planned work that was not completed, is reviewed. Completed work is "demo'ed" to the stakeholders. This is called the Sprint Demo.

The Sprint Review Demo occurs at the end of each sprint, immediately prior to the Sprint Retrospective. Sprint Demo Reviews are attended by an extended group of stakeholders, and can be considered a form of communication, validation, and recognition that the team has reached its intended goal.

USING CMMI TO STRENGTHEN SPRINT DEMO-SPRINT REVIEW / INTEGRATED PROJECT MANAGEMENT

The goal at the end of each sprint is to deliver potentially "shippable" product. The work completed for each user story delivered in the sprint is measured against the definition of done criteria. The project stakeholders participate in the process to make sure their expectations were met, and any issues are addressed on the spot. An explanation may resolve an issue, or an adjustment might need to be made to the product backlog. The customer may also be invited because it is a key stakeholder in the project.

Planning the Sprint Demo-Sprint Review meetings and getting the required stakeholders to participate can be a challenge. If not done well, negative impacts to the team and the product are likely. The Integrated Project Management process area contains practices that can be used to strengthen the Sprint Demo-Sprint Reviews. Some examples are:

- Making sure the Sprint Demo-Sprint Reviews are integrated into the milestones of the top-level plan. The milestones should be clearly identified and propagated down to the plans for stakeholders outside the immediate sprint team.
- Keeping the plans updated, and keeping the stakeholders informed. The Sprint Demo-Sprint Reviews are key milestones, and there should be no confusion on the part of the stakeholders.
- The day, time, and the goals of the Sprint Demo-Sprint Reviews should be distributed in advance

CMMI LEAD APPRAISER PERSPECTIVE: SPRINT DEMO/SPRINT REVIEW

The Sprint Review and Demo can be improved through the application of practices from Project Monitoring and Control, Integrated Project Management, and Validation.

Sprint Demo Reviews are project milestones as well as periodic reviews of the project's progress.

to ensure the stakeholders are prepared to participate effectively. Training about how the agile team works prior to the first meeting or with new stakeholders could be part of the process.

- Any issues with the attendance of key stakeholders should be identified and addressed immediately to minimize disruptions. Any questions about the work accomplished during the sprint should be addressed by the team during the meeting. Issues that can't be resolved should drive new backlog items or updates to existing backlog items. Teams should learn during the Sprint Demo-Sprint Reviews and feed the information into a sprint retrospective.

An effective Sprint Demo-Sprint Review meeting is one of the best ways to build credibility with key stakeholders. The informative components of Integrated Project Management are excellent sources for information about how to manage key stakeholders.

Integrated Project Management (IPM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
INTEGRATE PLANS	The practice makes sure all stakeholders are aware of key project milestones and their responsibilities. agile ceremonies should be identified.	There is a Product Increment, Release Plan, or similar plan that identifies relevant stakeholders and their responsibilities in attending the various agile ceremonies	Release Plan/Product Increment Plan or similar
MANAGE THE PROJECT USING THE INTEGRATED PLANS	The practice sets the expectation that the plan is used and maintained throughout the project	All relevant stakeholders are invited to the Sprint Reviews and Demos as documented in the "Release" Plan; completed work and planned work not completed are reviewed	Sprint Review/Demo attendance
MANAGE STAKEHOLDER INVOLVEMENT	The practice makes sure the stakeholders identified in the plan are participating in their activities. agile ceremonies are attended by the key stakeholders.	All relevant stakeholders are invited to the Sprint Reviews and Demos as documented in the "Release" Plan; invited stakeholders attend the Sprint Review/ Demo. If not, the meeting is moved to enable maximum attendance	Sprint Review/Demo attendance
RESOLVE COORDINATION ISSUES	The practice makes sure that any issues related to the participation by the stakeholders are captured and addressed. Any issues from the agile ceremonies are captured and resolved.	Invited stakeholders attend the Sprint Review/Demos. Issues with completed work that is demoed, as well as planned work not completed, are discussed and the product owner decides how to deal with these issues (keep item in product backlog or create new product backlog item)	Sprint Review/Demo attendance; notes from Sprint Reviews/Demos

USING CMMI TO STRENGTHEN SPRINT DEMO – SPRINT REVIEW / PROJECT MONITORING AND CONTROL

The goal at the end of each sprint is to deliver potentially "shippable" product. The work completed for each user story delivered in the sprint is measured against the definition of done criteria. The project stakeholders participate in the process to make sure their expectations were met, and any issues are addressed on the spot. An explanation may resolve an issue, or an adjustment might need to be made to the product backlog. The customer may also be invited because it is a key stakeholder in the project. The CMMI Project Monitoring and Control process area can help to strengthen the results of the Sprint Demo – Sprint Review by making sure the team is working as planned. Some examples are:

- Setting the expectation that the team attends the daily standups
- Making sure the team actually attends the daily standups
- Setting the expectation that the daily standups address what was done, what needs to be done, and whether impediments are blocking planned progress
- Making sure that the relevant stakeholders are invited to and participate in the defined Sprint Demo – Sprint Review milestones
- Setting the expectation that any issues identified during the standups or during the Sprint Demo – Sprint Reviews are addressed on the spot or through changes to the product backlog

Additional useful information can be found in the informative material of the Project Monitoring and Control process area.

USING CMMI TO STRENGTHEN SPRINT DEMO – SPRINT REVIEW / VALIDATION

The goal at the end of each sprint is to deliver potentially "shippable" product. The work completed for each user story delivered in the sprint is measured against the definition of done criteria. The project stakeholders participate in the process to make sure their expectations were met, and any issues are addressed on the spot. However, in the case of validation, what does this mean for an agile project when a Sprint Demo – Sprint Review is held? If the demo is led by the team doing the work, is validation occurring? The CMMI Validation process area clarifies expectations for validation, and can be used to strengthen the credibility of the team doing the work. For example:

- Making sure that validation is selected for items related to the end customer needs; for example, the end customer may need user manuals, installation instructions, training material, as well as working software.
- Making sure that validation is correctly done for each item being validated, and that data and results are documented. The goal is transparency.
- Involving the end user or end user advocates in the process of doing a Sprint Demo-Sprint Review. Some companies invite customers and ask them to interact with the product directly.
- Reporting the status of validation to higher-level management that supports it. Having higher-level management present at a Sprint Demo-Sprint Review is an effective way to build confidence in agile teams.

An effective Sprint Demo-Sprint Review meeting is one of the best ways to build credibility with key stakeholders including the customer. The informative components of Validation are excellent sources of additional information.

Project Monitoring and Control (PMC)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
MONITOR COMMITMENTS	The practice makes sure commitments made by the team are being met	Completed work and planned work not completed are reviewed	Daily standup attendance and participation
MONITOR STAKEHOLDER INVOLVEMENT	The practice makes sure the stakeholders are involved as planned. Attendance at the daily standup is the primary method.	All team members attend the daily standups	Daily standup attendance and participation
CONDUCT PROGRESS REVIEWS	The practice makes sure the team is completing the planned work. Attendance at the daily standup is the primary method to review progress.	Completed work and planned work not completed are reviewed; corrective action is taken if needed	Daily standup attendance and participation
CONDUCT MILESTONE REVIEW	The practice makes sure that stakeholders are brought in to make sure the release met their expectations	All relevant stakeholders are invited to the Sprint Reviews and Demos as documented in the "Release" Plan; invited stakeholders attend the Sprint Review/ Demo—if not, the meeting is moved to enable maximum attendance	Sprint Demo results; Defect Reports; notes from Sprint Review/Demo meetings
MANAGE CORRECTIVE ACTIONS	The practice makes sure that changes are made if needed to keep the project on track. New product backlog items are added or existing backlog items are updated.	New product backlog items are added or existing backlog items are updated	Issues from Sprint Review/Demo meetings; updated product backlog

Validation (VAL)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
SELECT PRODUCTS FOR VALIDATION	The practice makes sure that the correct products are being validated. For an agile team delivering software, the software is validated.	Prioritized items from the product/sprint backlog that have been completed during the Sprint are demo'ed for end users or their representatives	Sprint Demo results; Defect Reports; notes from Sprint Review/Demo meetings
PERFORM VALIDATION	The practice make sure that validation is completed based on the plan. For an agile team delivering software, the software is validated in its intended environment.	End users, or their advocates if they are present and engaged	Sprint Demo results
IDENTIFY AND INVOLVE RELEVANT STAKEHOLDERS	The practice makes sure that the relevant stakeholders responsible for the validation process are involved. For validation, the end users should be involved.	End users, or their advocates if they are present and engaged	Sprint Demo results; Defect Reports
REVIEW STATUS WITH HIGHER-LEVEL MANAGEMENT	The practice makes sure that higher-level management understands the effectiveness of the validation process	Higher-level management is present and engaged, and process steps or issues related to the Sprint Demo results are described and discussed	Sprint Review/Demo results

Sprint Planning

SUMMARY

A Sprint Planning Meeting occurs at the beginning of each sprint, and is a negotiation between the agile team and the product owner (or customer) as to what value will be delivered in the upcoming sprint.

During the Sprint Planning Meeting, a sprint backlog is developed, and tasks are identified to support the planned user stories. Team members assess how much work they can accomplish during the sprint, and the team members subscribe (or are assigned) to the various tasks to be completed. The team may decide to assign or subscribe to tasks during actual sprint execution instead.

A Sprint Planning Meeting should take place every 7 to 30 days, depending on the duration of each sprint, and should only identify the value to be delivered during that sprint.

USING CMMI TO STRENGTHEN SPRINT PLANNING / PROJECT PLANNING

The product backlog is estimated and prioritized over time to feed sprint planning, and sprint team members are occasionally involved in the backlog refinement based on their knowledge or experience. By the time a backlog item is ready for sprint planning, it is clearly described and sized so that it can be done in the next sprint. The velocity of the team is also known so the product owner can maximize the value of work done. The end goal of sprint planning is a sprint backlog that contains tasks in hours and owners for each task. So how can the Project Planning process area of the CMMI make the sprint planning more robust? Some of the practices map directly to the sprint planning activities, and provide additional information that can benefit an agile team. Some examples are:

- Making sure that the scope of the sprint is clear, and that nothing has been missed. Teams can easily overlook non-software work products in their planning, and these oversights can cause sprint goals to not be met.
- Setting the expectation that estimates are needed for everything produced during the sprint by the team.
- Setting the expectation that efforts are needed for the work being done by the team. Relative sizing based on story points and the planning game must in the end be translated into hours to complete the work. Each team member should estimate their own work.
- Making sure that each team member has an opportunity to review the plans and estimates for the sprint prior to making a commitment. One of the strengths of agile is individual ownership, and the CMMI emphasizes the importance of this.
- Setting the expectation that adjustments should be made to the plan based on vacations, training, or other external influences that could impact the plan.

- Making sure that each team member commits to the plan prior to beginning the work. Both agile and the CMMI emphasize and reinforce the need for committed team members. In Scrum, commitment is visual via a Scrum board or collaboration tool.

A good project plan that is understood, realistic, created by the team members doing the work, and committed to by each team member is a positive influencer of project success. The informative components of Project Planning are excellent sources of additional information.

LEAD APPRAISER PERSPECTIVE: SPRINT PLANNING

Sprint planning occurs frequently during an agile project, and can be considered an iterative and incremental form of planning, as well as a validation of the product backlog. Project Planning, Validation, and Requirements Management are all related to sprint planning, and their practices can be used to improve the Sprint Planning Meeting.

Project Planning (PP)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTIMATE THE SCOPE OF THE PROJECT	The practice expects work packages (stories) in a defined sequence (prioritized backlog), which can be confirmed (sprint burn down and release burn up)	The "scope" being planned is a fixed duration sprint or iteration	Sprint backlog
ESTABLISH ESTIMATES OF WORK PRODUCT AND TASK ATTRI-BUTES	The practice expects the teams to define the estimate for the work included in the next sprint. All work should be included.	Velocity is planned for the sprint based on story points	Story points assigned to each user story in the backlog
ESTIMATE EFFORT AND COST	The practice makes sure the organization is estimating both effort and cost. In agile, the work is typically time and people constrained, so cost is a factor of # of people over time.	The team agrees that it will be able to complete the tasks in the sprint backlog during the sprint without working over-time	Sprint backlog
REVIEW PLANS THAT AFFECT THE PROJECT	The practices make sure that all items that could impact the planning are considered. An open discussion about all sprint backlog items is expected.	All team members and product owner are present and in agreement	Sprint planning meeting notes
RECONCILE WORK AND RESOURCE LEVELS	The practice makes sure that the plan is realistic and achievable. The assumption is that commitment can only be given when the plan is feasible.	All team members and product owner are present and in agreement; team members' availability are considered when agreeing on the sprint backlog (e.g., a team member may be away on vacation)	Sprint backlog
OBTAIN PLAN COMMITMENT	The practice expects that each team member agrees to their assigned tasks. In agile, commit-ment is shown by individual names on stories and related sprint tasks.	All team members and product owner are present and in agreement	Completed Sprint Plan

USING CMMI TO STRENGTHEN SPRINT PLANNING / ORGANIZATIONAL PROCESS DEFINITION

By the time sprint planning is done, each backlog item is clearly described and sized so that it can be done in the next sprint. The velocity of the team is also known so the product owner can maximize the value of work done. The end goal of sprint planning is a sprint backlog that contains tasks in hours and committed owners for each task. The CMMI Organizational Process Definition process area gives guidance for what an organization needs to have in place to describe how sprint planning is done for an agile project. But is it necessary to create tailoring criteria, standards, processes, and work instructions? The answer is no and yes. Yes, knowledge should be captured and made available to teams that want to use agile approaches to plan their work. No, detailed documents of all kinds are not needed to describe every nuance and detail. Here are examples of what is needed:

- Tailoring information to help projects make a decision about whether or not to use agile planning methods. Not all types of development projects can benefit by adopting agile planning methods. Better to provide guidance so that the best decision is made than provide no guidance at all.
- Information about how to do agile planning methods. This could be done through formal training methods, or it could be done by pairing less experience people with more experienced people during a sprint planning meeting.
- Information about how to configure existing tools to support a team using agile planning methods.
- Simple checklists to help an inexperienced team make sure nothing is missed during its first few instances of sprint planning.

Agile is about learning and sharing knowledge. The informative material of Organizational Process Definition contains the best practices organizations have used over time to improve the way they work.

USING CMMI TO STRENGTHEN SPRINT PLANNING / REQUIREMENTS MANAGEMENT

In sprint planning, the team, working with the product owner, attempts to size the user story to create the sprint backlog. The user story is a translation of the user needs, including requirements, into a language that describes the outcome from the user perspective. The term user is very broadly interpreted, and can be the final customer, an organizational customer, the customer of the next stage, and so on. When sprint planning is in progress, requirements must be understood. Clear user stories minimize the need for the product owner to spend time clarifying sprint backlog items, and improves the ability of the team to commit to the requirements and make sure the planned work during the sprint is aligned with the requirements. The Requirements Management process area provides some guidance to strengthen sprint planning. Some examples are:

- Making sure user stories in the sprint backlog are clear through the application of acceptance criteria and open communication channels between requirements stakeholders. Agile expects transparency, and the CMMI reinforces this need, especially around requirements.
- Reinforcing the idea that commitment by the team doing the work during a sprint is a critical part of product development. CMMI clarifies that commitment is also important for requirements changes, something that is often missed.
- Setting the expectation that the project work is aligned with the user stories in the sprint backlog. This is closely related to the idea of commitment, and encourages the product owner to prevent changes and scope creep during a sprint.

The informative components of Requirements Management are excellent learning sources for other ways to make sure the sprint backlog facilitates efficient sprint planning.

Requirements Management (REQM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
UNDERSTAND REQUIREMENTS	The practice makes sure that discipline is applied to the activity of creating or clarifying epics or user stories. Discipline in this area optimizes and increases the efficiency of backlog development and sprint planning.	The product owner is present during the Sprint Planning Meeting and assists the team in understanding the user stories so that valid and sufficient tasks can be identified	Attendance at Sprint Planning meeting
OBTAIN COMMITMENT TO REQUIREMENTS	The practice makes sure the business requirements and functions required are clearly understood to support creation of user stories that are meaningful to the business and implementable in sprints by the teams	All team members and product owner are present and in agreement	Attendance at Sprint Planning meeting; completed Sprint Plan
ENSURE ALIGNMENT BETWEEN PROJECT WORK AND REQUIREMENTS	The practice finds inconsistencies between requirements and project plans and work products, and makes sure they are resolved prior to a sprint	User stories placed in the sprint backlog are removed from the product backlog, or negotiated back to the product backlog after being requested by the product owner	Updated sprint or product backlog

Team Agreements

SUMMARY

A Team Agreement is a social contract entered into by members of an agile team to define team behaviors, expectations, and agreed norms. Sometimes it is referred to as a "Working Agreement."

Some Team Agreements are simple ideas written on a whiteboard, while others are detailed project charters that contain important metadata about the team itself.

Team Agreements are typically developed at the beginning of a release and can be refactored after each Sprint Retrospective or Sprint Demo.

USING CMMI TO STRENGTHEN TEAM AGREEMENTS / INTEGRATED PROJECT MANAGEMENT

Self-organizing teams are an important goal of implementing agile practices, and a self-organized team is a key success factor in making agile work in any organization. No one (not even the Scrum Master) tells the Development Team how to turn product backlog into increments of potentially releasable functionality. Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the team as a whole. The CMMI Integrated Project Management process area captures ways to help a team self-organize, and the information is useful for teams using agile. Some examples are:

- Creating a shared vision that helps the team have an identity and a common purpose. Interfaces between the Development Team and external project stakeholders should provide clarity. A shared vision should be visible, compelling, and understandable by any stakeholder who sees it.
- Making sure the resources needed by the team to achieve the shared vision are available when needed. A simple yet compelling team vision is an excellent way to get upper-level management behind a project.
- Setting the expectation that the shared vision is a living document that should be updated and improved as the team learns through the product development. This is a good fit for the agile goal of short feedback loops and continuous learning.

Many more examples and suggestions are found in the informative components of the Integrated Project Management process area.

CMMI LEAD APPRAISER PERSPECTIVE: TEAM AGREEMENTS

Team Agreements can be improved through the application of practices in both OPD and IPM, and, if they are selected for improvement by an agile team, can benefit from the use of Generic Practice (GP) 3.2.

Integrated Project Management (IPM)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH AND MAINTAIN TEAMS	The practice gives guidance for creating a shared vision for the team or teams involved in developing the product	The team agreements contain sufficient information to define team norms and team information	Team Charter Project WIKI Information Radiator (white board, poster, etc.)
IDENTIFY AND INVOLVE RELEVANT STAKEHOLDERS	The practice makes sure that the relevant stakeholders responsible for the development are included in the team	The team agreement identified local and extended team members, and engagement was verified	Team Charter Project WIKI Information Radiator (white board, poster, etc.)

USING CMMI TO STRENGTHEN TEAM AGREEMENTS / PROJECT PLANNING

Self-organizing teams are an important goal of implementing agile practices, and a self-organized team is a key success factor in making agile work in any organization. Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the Development Team as a whole. When creating a team agreement, consideration must be given not only to the Development Team, but to all external stakeholders that influence or are influenced by the Development Team. The CMMI Project Planning process area captures ways to help make sure important stakeholders are identified and involved. Some examples are:

- Making sure thought is applied to the decision making used to identify relevant stakeholders. Missing a key stakeholder can have a negative future impact to the project, and minimizing this risk should be a high priority.
- Identifying any training that may be needed for external stakeholders to be effectively involved in the project. Support functions may not understand the agile practices used by the team, and this knowledge gap can be a negative influence in the organization.
- Clarifying when stakeholders need to be involved. A team that uses agile practices and works in sprints should not have team members that come and go, but external stakeholders may need to be involved at different times in the development. Good planning and communication is needed.

Having a clear picture of stakeholders and their responsibilities is a key principle for any type of development from traditional models to agile models. The informative components of the Project Planning process area provide additional information that can be useful to a team that is on the path to the self-organized goal.

Project Planning (PP)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
PLAN STAKEHOLDER INVOLVEMENT	The practice gives guidance to make sure that all relevant stakeholders are identified, and that their responsibilities are clear. This concept is also important for a team using agile development practices.	The team agreement identified local and extended team members, and engagement was verified	Team Charter Project WIKI Information Radiator (white board, poster, etc.)

Technical Debt

Technical Debt is incurred when the agile team proactively determines that a less optimum, less efficient, or less elegant solution is appropriate given constraints of time, budget, or resources.

As technical debt increases, the costs and effort to continue development, or maintain an existing system, become too high, and a "technical debt sprint" can be scheduled within a release to improve code quality.

Technical Debt is a design approach that is:

- Expedient in the short term
- Costly in the long term because a technical context is created in which the same work takes longer or costs more to do in the future than it costs to do in the present

The challenge to any Development Team is:

- The business community loves technical debt (short term gain)
- The technical community hates technical debt (long term pain)

To improve development over time, we must improve our ability to:

- Quantify the technical debt of our decisions
- Balance the business needs with the technical needs of our projects

USING CMMI TO STRENGTHEN THE MANAGEMENT OF TECHNICAL DEBT

The decision to incur technical debt can be strengthened through the use of Specific Goal 1: Select Product Component Solutions in the Process Area Technical Solution. The use of criteria and structured decision making, with its roots in Decision Analysis and Resolution (DAR), will help ensure that consistent and appropriate decisions are made as to the proactive degradation of code quality.

Agile teams can also leverage the practices in Project Planning to plan, prioritize and sequence the elimination of technical debt by allocating them as user stories to future sprints and releases.

CMMI LEAD APPRAISER PERSPECTIVE: TECHNICAL DEBT

Lead Appraisers, and Appraisal Team Members, should remain aware that the decision to incur technical debt should be proactive, and it is not the result of defects or coding errors. Technical Debt should be collected and allocated to user stories and sprints, and prioritized appropriately by Product Owners. As such, they represent execution of Project Management, Requirements Management, and Technical Solution.

Technical Solution

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
DEVELOP ALTERNATIVE SOLUTIONS AND SELECTION CRITERIA	Applying criteria and structure to the decision regarding what should be selected as a technical debt candidate	Criteria are used, and rational is documents (see Decision Analysis and Resolution)	Record of Decision that includes impact analysis and rationale

Project Planning

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTIMATE EFFORT AND COST	Helping to understand the impact of incurring technical debt so that a sound decision is made	An estimate and impact analysis is performed	Basis of Estimate for the cost of incurring technical debt
IDENTIFY PROJECT RISK (ALSO SEE RISK MANAGEMENT)	Helping the team to fully understand the risk to the organization before committing to technical debt	A risk analysis is performed	A record or risk and mitigation plans are both available.
OBTAIN PLAN COMMITMENT	Ensuring that the appropriate stakeholders are committed to incurring the cost of technical debt	Commitment is demonstrated by Product Owner of other authority	A documented change in the backlog that includes the technical debt user story

Test Driven Development

SUMMARY

Test Driven Development, or TDD, is an agile technique most commonly used as part of an Extreme Programming (XP) project.

Used as both an early Validation and Verification technique, an individual developer will write a basic test case to verify the desired functionality, knowing that it will fail, and then writes the minimum amount of code to pass the test. The developer will then clean up the code to ensure that it meets acceptable performance and coding standards and principles ("refactoring").

TDD brings the most value when paired with an iterative and incremental life approach with the short sprints or iterations that are usually found with agile projects.

USING CMMI TO STRENGTHEN TEST DRIVEN DEVELOPMENT / REQUIREMENTS DEVELOPMENT

Test Driven Development (TDD) allows a developer to incrementally create test cases and software that meets the test cases. An automated test framework is normally used to ensure that TDD practices are simple and efficient. Later in the development when the software is impacted by a change, the proven test cases can be used to make sure the change causes no unintended consequences. The practice of doing TDD helps the team create and mature the requirements over time, and also fits well with the CMMI Requirements Development process area. Some examples are:

- Making sure that thought is given to the operational concepts and scenarios in which the final product is used. In many cases, requirements exist for user interaction, operation environment, maintenance, and end of life. Operational concept and scenario development is an iterative process, and TDD is a good method to use.
- Identifying as soon as possible any requirements that cause issues or risks in the larger product context. Time constraints, system attributes, and quality attributes are possible sources of requirements conflict, increase risk, or increased cost. Considering these items while using TDD can help achieve an optimal balance.
- Making sure the intended environment is considered when maturing the requirements via TDD practices. Simulated environments are good in early stages of development, but the test cases created should translate over to validation of the requirements on the final product.

The informative components of Requirements Development are excellent sources for learning about ways to get the most out of Test Driven Development practices.

CMMI LEAD APPRAISER PERSPECTIVE: TEST DRIVEN DEVELOPMENT

TDD is a powerful technique that can improve the quality of code and requirements, so therefore has a strong relationship to Validation, Verification, and Requirements Development.

Requirements Development (RD)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH OPERATIONAL CONCEPTS AND SCENARIOS	The practice sets the expectation that the requirements cover all possible use cases. TDD practices can help flesh out the requirements.	Tests are written in sufficient detail, and used throughout the process as requirements are fleshed out	Test cases
ANALYZE REQUIREMENTS TO ACHIEVE BALANCE	The practice sets the expectation that the requirements mature over time, make sense in the larger context, and are sufficient. TDD practices meet this expectation.	Tests evolve in sufficient depth and breadth to cover all aspects of sufficiency	Evolved test cases
VALIDATE REQUIREMENTS	The practice sets the expectation that requirements are validated by the end user or end user surrogate. Automation can be used to facilitate this practice.	Tests continue to evolve, and automation is used to ensure that the requirement is appropriately validated	Automated test cases

USING CMMI TO STRENGTHEN TEST DRIVEN DEVELOPMENT / VALIDATION

Test Driven Development (TDD) allows a developer to incrementally create test cases and software that meets the test cases at the earliest point of development. As the test cases and software are created, requirements are created, updated, corrected, and matured in an efficient closed-loop process. Learning is immediate, and cost avoidance naturally occurs because requirements defects are discovered at the point of development, and do not escape to later parts of the development lifecycle. But how can TDD be strengthened by the CMMI Validation process area? As in many of the earlier sections, the CMMI contains useful practices and information that helps fill in some of the detail left out by agile frameworks. In validation, some examples are:

- Making sure the Development Team is considering testing the product in its intended environment as it uses TDD practices to create and mature test cases that meet validation requirements.
- Setting the expectation that results of validation are fed back to the Development Team so it can learn and make improvements to both requirements and the test cases that validate the requirements. This expectation is easily overlooked in many projects.

Having validation in mind early in the development and using TDD to support validation is a proven way to avoid costly requirements defects. The informative components of Validation contain additional information that teams can use to strengthen TDD practices.

Validation (VAL)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
PERFORM VALIDATION	The practice sets the expectation that the product is tested in its intended environment ideally by the user or user surrogate. The TDD test case inventory can help facilitate this activity.	Test case contains sufficient depth to account for appropriate validation	Test cases
ANALYZE VALIDATION RESULTS	The practice makes sure that validation results are recorded and fed back to the appropriate stakeholders. The expectation reinforces the closed-loop learning goal of agile.	TDD test results are analyzed, and code or user story corrections are made	Test results

USING CMMI TO STRENGTHEN TEST DRIVEN DEVELOPMENT / VERIFICATION

Test Driven Development (TDD) allows a developer to incrementally create test cases and software that meets the test cases. The practice of doing TDD helps the team to create and mature the requirements over time, and provides immediate verification of the requirements. As the test case inventory is built up, requirements changes can be easily evaluated and verified through regression tests. Unintended consequences are more likely to be found using the test case inventory, and correcting them at the point of introduction is excellent cost avoidance.

A product that works right the first time when a change is introduced and delivered is an excellent way to build confidence with higher-level management and external customers. So how can the CMMI Verification process area improve the practice of TDD? Some examples are:

- Setting the expectation that verification is done early and often in the development of a product. The results of verification save the considerable cost of fault isolation and rework associated with troubleshooting problems.
- Setting the expectation that results of verification are used by the Development Team to learn and make improvements to both requirements and the test cases that verify the requirements. This expectation is easily overlooked in many projects, but is a core principle of both the CMMI and agile.

Eliminating defects at the point they are introduced is a proven way to deliver a successful project. The informative components of Verification contain additional information that teams can use to strengthen TDD practices.

Verification (VER)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
PERFORM VERIFICATION	The practice sets the expectation that the product is tested to make sure it meets the requirements. The TDD practices and the test case inventory can help facilitate this activity.	Test case, user story, and DOD are traceable to ensure all user stories are subject to testing	Scrum board depicting traceability; sufficient test cases based on user story functionality
ANALYZE VERIFICATION RESULTS	The practice makes sure that verification results are recorded and fed back to the appropriate stakeholders. The expectation reinforces the closed-loop learning goal of agile.	TDD test results are analyzed, and code or user story corrections are made	Analysis of test results

User Stories

SUMMARY

A user story is a simple-language description of a discreet piece of functionality, described from the user's standpoint. Typically broken down from a larger "epic," or simply a larger user story, it is a self-contained piece of functionality that can be delivered in a single sprint.

User stories are the basis of estimation for a sprint, and are usually estimated in story points, leading to a project velocity for a given agile team.

Originally depicted on a single "card" and placed on a Scrum board, user stories can also be depicted in any appropriate Requirements Management tool.

User stories are typically owned by the product owner, who serves as the customer or customer advocate.

USING CMMI TO STRENGTHEN USER STORIES / REQUIREMENTS DEVELOPMENT

User stories are the requirements in an agile project. The user story is a translation of the user needs, including requirements, into a language that describes the outcome from the user perspective. The term user is very broadly interpreted, and can be the final customer, an organizational customer, the customer of the next stage, and so on. Clear user stories minimize the need for the product owner to spend time clarifying product backlog items, and improves the ability of the team to "get it right the first time" during a sprint. The CMMI specific goals and practices in the Requirements Development process area can give clarity to the process of creating user stories, and can help bring robustness to the product backlog. Some examples are:

- Making sure that all possible sources of needs for the product, not just needs from an external customer, are considered in the creation of the product backlog. Missing an important need early in a project can have a large negative future impact.
- Setting the expectation that the requirements, as captured in a user story, have a priority assigned to them that meets the needs of the business and the customer of the product. This is input to backlog grooming and sprint planning, both of which must be done well.
- Setting the expectation that the user stories are allocated in the best manner to feed the development and minimize external dependencies.
- Making sure that interface constraints are considered and represented in the user stories. As the project progresses and more is learned, the interfaces should be updated as needed.
- Setting the expectation that the user stories must be necessary and sufficient to drive development in sprints.

Interaction between the product owner and the Development Team in planning sessions should fulfill this practice.

- Making sure that the user stories satisfy the end user requirements in the intended environment. This should be done early and often in the development to confirm and reaffirm product feasibility.

Agile teams thrive when feedback loops are short and learning occurs. The informative components of Requirements Development are excellent sources for learning about ways to make sure the user stories in the product backlog are complete and robust.

CMMI LEAD APPRAISER PERSPECTIVE: USER STORIES

In environments where Test Driven Development (TDD) is employed, and there is a robust Definition of Done (DOD), development of user stories can be considered a robust form of Requirements Development. More informal use of user stories, without the requisite TDD or DOD, will simply constitute informal Requirements Development.

The use of short sprints or iterations (7-30 days) is critical to the success of TDD for ensuring that requirements are complete and sufficient.

Requirements Development (RD)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ELICIT NEEDS	The practice ensures that epics and user stories are captured early in the product lifecycle (e.g., during the creation of the product roadmap, during release planning, from focus groups, teams) in concert with all key stakeholders.	A product owner or customer advocate provides initial epics or user stories	Story cards; product backlog in a tool (e.g., Jira)
TRANSFORM STAKEHOLDER NEEDS INTO CUSTOMER REQUIREMENTS	The practice fully supports creation of epics and user stories. The practice includes prioritization of the backlog, which defines constraints on implementation.	Epics or large-user stories are broken down	Story cards; product backlog in a tool (e.g., Jira)
ALLOCATE PRODUCT COMPONENT REQUIREMENTS	The practice makes sure the requirements are allocated to a delivery increment to feed sprint planning	User stories are broken into tasks that define the product's functionality; DOD is defined for each user story	Story cards; product backlog in a tool (e.g., Jira); tasks on a task board
IDENTIFY INTERFACE REQUIREMENTS	The practice supports the need to capture interface requirements as user stories	The breakdown of user stories may identify additional interface requirements	Story cards; product backlog in a tool (e.g., Jira); tasks on a task board
ANALYZE REQUIREMENTS	The practice focuses on ensuring all epics and user stories are necessary and support higher-level requirements	TDD is employed to validate requirement; DOD is defined in sufficient detail	Initial test cases; Definition of Done
VALIDATE REQUIREMENTS	The practice supports ensuring epics and user stories meet the end user requirements.	TDD is used to evolve the code, and short iterations are used	Completed test cases; version of test cases

Velocity

SUMMARY

"Velocity" is a tool used by agile teams for capacity planning and tracking of ongoing work during sprints and releases. Based on an agreed-upon unit of work, usually a story point, velocity is an indicator of how much value a given Scrum team can be expected to deliver in an upcoming sprint, or how much value they have actually delivered.

Velocity is team and duration specific, meaning that as a team changes, we cannot reasonably expect its velocity (value delivered) to remain constant. Any change to the team or sprint duration will affect the team's ability to deliver on a given velocity.

During sprint planning, an agile team will set a velocity objective for the sprint. A Burn-Down Chart (BDC) or "Burn-Up Chart" is used as an "information radiator" that depicts a team's actual velocity in real time.

Agile teams should be aware that velocity is but ONE example among many of a metric that is valuable to the organization. In and by itself, it is insufficient for management of a software project.

USING CMMI TO STRENGTHEN VELOCITY / MEASUREMENT AND ANALYSIS

The velocity of an agile team, the capacity to get work done, is a foundational measure used to improve capacity planning and tracking of ongoing work during sprints and releases. As with many measures, the data can be used to create many metrics, some of which add value, and some of which don't add value. Sprint Burn-Down Charts and Burn-Up Charts can be based on story points or effort, but using both gives a better picture of the actual progress in the sprint. These types of considerations should be well thought out, and the CMMI Measurement and Analysis process area can be used to make measures used by agile teams more robust. Some examples are:

- Making sure that the measures generated by the team align with one or more goals of the business. This is a hurdle that can help eliminate unnecessary measures up-front before time is invested into collecting the data.
- Setting the expectation that the data needed for the measure is specified along with how to collect it, when to collect it, and what the final result means toward the objective it supports.

- Making sure that the results are communicated to all the stakeholders who are impacted. Support functions and higher-level managers want to know how this thing called “agile” is impacting them for the better.
- Making learning and improving the way work is done a key characteristic of the team. Closed-loop learning is fundamental to an agile team, and well defined measures strengthen the practice.

All of the examples listed above are common sense, but many times projects and organizations create metrics for the sake of having metrics. More is not always better. The informative material in the Measurement and Analysis process area contains knowledge that can make any measurement program more robust.

CMMI LEAD APPRAISER PERSPECTIVE: VELOCITY

Velocity is a core metric for any agile team, but it is also the foundation for estimating and tracking of project work. Because agile teams work in a “fixed-time-box,” “fixed-team-size” environment, measurements such as CPI and SPI are superfluous, making velocity an important measure of project success.

Because velocity is unique to a given agile team, velocity predictions at an organizational level are not possible unless all teams stay intact throughout an entire release.

Measurement and Analysis (MA)

Practice	Strengthens agile implementation by:	Could satisfy the practice if:	Could be demonstrated by:
ESTABLISH MEASUREMENT OBJECTIVES	The practice expects that metrics meet a business need, and provide useful information about that need	Velocity is considered a core metric to determine project success by management, and objectives for velocity are set at each Sprint Planning Meeting.	Sprint Plan with velocity targets tied to user stories
SPECIFY MEASURES	The practice expects that information exists that describes all important attributes of a metric including how to interpret the data in the business context	A specification for velocity exists that describes how it is capture, analyzed, displayed, and communicated	A project or organizational WIKI Policy document Training material
SPECIFY DATA COLLECTION AND STORAGE PROCEDURES	The practice expects that information exists for how to gather the data needed and where the data is stored for future reference	A specification exists, and information radiators are established	Information radiators such as BDC/BUC, or virtual representations in tools
SPECIFY ANALYSIS PROCEDURES	The practice expects that information exists for how to do any calculations with the gathered data	A burn-down chart or other tracking system is established for the team; a tool that provides a virtual BDC is use	Visual BDC Virtual Tool Status Reports
OBTAIN MEASUREMENT DATA	The practice expects that measurement data is gathered and analyzed as defined	A burn-down chart or other tracking system is displaying actual project data; a tool that provides a virtual BDC is use	Any information radiator that depicts velocity in real time
COMMUNICATE RESULTS	The practice expects that the results are used by the team to learn and improve. The results should also be communication to all the project stakeholders and the upper level managers.	A burn-down chart or other tracking system is employed; a tool that provides a virtual BDC is use	Any information radiator that depicts velocity in real time

References, White Papers, Presentations

CMMI or Agile: Why Not Embrace Both! (White Paper)
http://cmmiinstitute.com/resources/cmmi-or-agile-why-not-embrace-both

CMMI vs. Scrum. No! CMMI + Scrum! (Article)
https://www.linkedin.com/pulse/20140917030508-798283-cmmi-vs-scrum-no-cmmi-scrum

The Elements of Scrum (Book)
Chris Sims, Hillary Louise Johnson, Dymaxicon, 2011

Agile Project Management: Creating Innovative Products (Book)
Jim Highsmith, Addison-Wesley Professional, 2009

Implementing Scrum and Agile Together (White Paper)
https://www.scrumalliance.org/community/articles/2011/february/implementing-scrum-(agile)-and-cmmi®-together

Ask The CMMI Appraiser : An Agile Lead Appraiser's Take on the CMMI and Agile (Blog)
http://askthecmmiappraiser.blogspot.com

Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement (Book
Paul McMahon
Addison-Wesley Professional, 2010

Agile Resiliency: How CMMI will make Agile Thrive and Survive (Presentation)
Jeff Dalton
http://cmmiinstitute.com/resources/agile-resiliency

Manifesto for Agile Software Development
agilemanifesto.org

Agile CMMI Blog
http://agilecmmi.com

Agile/Scrum Developments using the CMMI - NASA (Presentation)
Kent Aaron Johnson
http://www.nasa.gov/ppt/482611main_2010_Tuesday_4_cmmi_Johnson_Kent_Agile%20Scrum%20Development%20Using%20CMMI%20v1.4a.ppt

Adding Practices to Scrum to Achieve your Goals (White Paper):
http://www.processgroup.com/pgpostapr2013.pdf

For many of these and more resources, visit
http://cmmiinstitute.com/resources

Development Team

PRINCIPAL AUTHOR

- Jeff Dalton

CONTRIBUTING AUTHORS

- John Voss
- Marion Taros
- Winifred Menezes
- Timothy Zeller
- Ross Timmerman
- Laura Adkins
- Kirk Botula

REVIEWERS

- Katherine White
- Paul McMahon
- Winifred Menezes
- Jeff Dalton
- Neil Potter
- Dan Torrens
- Prabhakar S.

MARKETING

- Chavonne Hoyle
- Katie Tarara

CMMI® Institute - Home of the Capability Maturity Model Integration (CMMI®)

As the organization behind the Capability Maturity Model Integration (CMMI), a capability and process improvement model that guides organizations in high-performance operations, the CMMI Institute is working to build upon CMMI's success, advance the state of the practice, accelerate the development and adoption of best practices, and provide new and evolved solutions to meet the emerging needs of businesses around the world. The institute also leads the People Capability Maturity Model and the Data Management Maturity Model.

The CMMI Institute supports the worldwide adoption of its solutions in small and large organizations alike in a variety of industries, including aerospace, finance, health services, software, defense, transportation, and telecommunications.

The CMMI Institute licenses its products and services through a network of delivery partners; conducts training and certification; sponsors conferences, workshops, and events; and promotes the benefits of process improvement models and appraisal methods.

A Rich Heritage

CMMI Institute is a part of the ISACA family, the global non-profit association helping professionals to realize the positive potential of technology.

The CMMI is the result of more than 20 years of ongoing work at Carnegie Mellon University by members of industry, government, and the Software Engineering Institute, a federally funded research and development center.