CMMI and Agile Development: A Binary Choice?

The Capability Maturity Model Integration (CMMI) for Development has a long and impressive history in industry for progressing the cause of process improvement in software and systems programs. Fueled with funding from the US government, Carnegie Mellon University (CMU) worked with US Air Force, NASA and other organizations to create a framework for quantifying a software development organization's capability maturity. This was first done specifically for software at a time when many industries were seeing dramatic software failures. The byproduct of this initial research was the Capability Maturity Model (CMM). The CMM later morphed into the CMMI which is a model for process improvement in the following areas of interest:

- Product and service development – CMMI for Development (CMMI-DEV)
- Service establishment and management – CMMI for Services (CMMI-SERV)
- Product and service acquisition – CMMI for Acquisition (CMMI-ACQ)

Agile development is a paradigm for software development processes that are characterized by highly collaborative, cross-functional teams who work closely with their customers to deliver regular increments of functional software capability that delight their customers and end users. Agile practices require flexibility and rapid response to change in requirements. The tenets of an agile development include (as stated in the *Agile Manifesto* [1]) :

- Valuing individuals and interactions over processes and tools
- Valuing working software over comprehensive documentation
- Valuing customer collaboration over contract negotiation
- Valuing responding to change over following a plan

Stating that while there is value in the things on the right they (the Manifesto authors) value the items on left more.

The CMMI focuses on having institutionalized processes that are understood and respected by the software development team and their management and that are pervasive throughout the organization. Agile development practices appear to encourage abandoning well document processes for more rigorous interactions and trade-offs. At a top level it appears that what the CMMI-DEV is trying to achieve flies directly in the face of the tenets of an agile software development project – it seems like it might be a binary choice. And yet some have been quite successful with an intelligent marriage of the two approaches.

This paper delves into the specifics of each of these approaches with respect to where and how they have and have not worked well together. The first section discusses the CMMI – what it is and how it came to be. The second section applies the same discussion to agile in a similar light. Following this is a comparison of the two concepts – where they gel nicely and where there are potential conflicts. After this is a discussion of several instances where the marriage was attempted – covering the roadblocks, successes and lessons learned.

CMMI

In the mid 1980's The US Department of Defense (DoD) issued a Request for Proposal (RFP) looking for ideas to help with the many seemingly massive and very public software failures linked to DoD programs. A group at the Carnegie Mellon University (CMU) that would eventually become the Software Engineering Institute (SEI) won the award. In 1988 Watts Humphrey introduced the first version of the Capability Maturity Model (CMM) and in 1989 he published his book *Managing the Software Process* which introduced and describe the CMM. The CMM was successful but definitely had its detractors. Many interpreted it as very prescriptive and inflexible and felt that the application of it to their projects would be a productivity detractor rather than an improvement. Yet many government contracts required an organization to be appraised to a certain CMM level (more on levels in following paragraphs) for them to be considered as a viable contractor. So organizations read the CMM and got appraised.

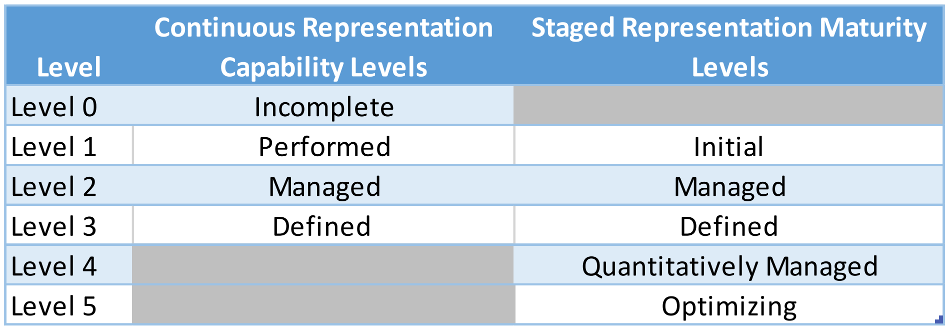
Through the process of mentoring and performing appraisals, the SEI and other wise software process professionals learned a lot about what the CMM got right and what it got wrong. The knee jerk reaction was to develop different – more customized versions – of the CMM to deal with situations where the CMM didn't quite fit. One of the important reasons for this is that the CMM was clearly biased towards the traditional waterfall method of developing software with language that made this seem the only way to go. The SEI recognized that the real lesson learned was that they needed to consolidate the various one-off situations and pull them all together into a model that would meet wide market needs while being more instructive and less prescriptive. At the same time they realized another important thing – while the CMM was originally intended to apply to software projects – there was significant overlap in many of the process areas that applied equally well to services and acquisition. They began to work towards a more comprehensive model that could accommodate all of these areas.

In 1998 the first version of the CMMI-DEV was introduced to industry. While there were of course early adopters, there were many who were content to stay with the CMM and many who did not really want to think about CMMI because of bad experiences with the CMM. This was in a period of time when agile practices were getting noticed and coming to the forefront – especially on small to medium size projects. CMM/CMMI users were starting to wonder how agile might be integrated into their development processes while still maintaining their appraisals. In 2005 the CMMI Tech Conference added an agile and lean track and in 2006 the Software Engineering Process Group Conference (SEPG – run by the SEI) added an agile track as well. In 2006 CMMI-DEV 1.2 was released, in 2007 CMMI-ACQ was release and a new version of the CMMI-DEV 1.3 was released in 2010.

So what exactly is the CMMI? Starting out, this author had a fairly comprehensive understanding of the CMM and agile and thought that the focus of this research effort would mostly be the intersections and contradictions of the two areas of thought. This was only partially true. The first thing that struck this author was that some of the CMMI terminology was different enough from the CMM to cause pause. First step therefore was to read the CMMI-DEV (sigh – this was not an easy read). Four hundred pages later… the amount of narrative is exhausting but the completeness and thoroughness of coverage is impressive and instructive.

The CMMI is a model, not a standard. The CMMI tells organizations what goals they should master to reach a desired level of process maturity; in no way does it attempt to tell them how to do it. The CMMI-DEV provides a road map for an organization to progress from lower levels of process maturity to higher levels. It also recognizes that some process areas are not relevant to an organization or do not align to their business objectives. It provides two paths towards maturity improvements – an organization can choose to aspire to maturity levels by conquering all the process areas to attain a certain level (staged representation) or they can choose to acquire capability levels to attain a certain level of maturity in process areas that are important to their business objectives (continuous representation).

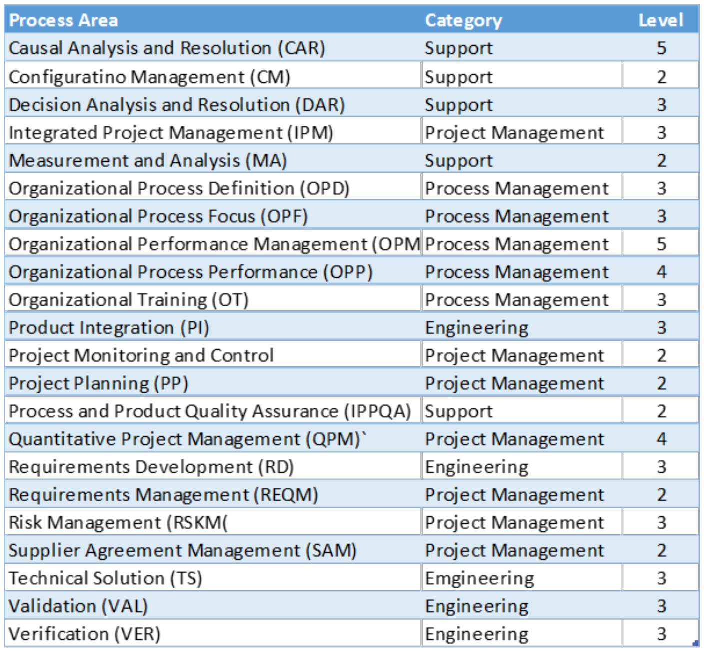
So what does all this mean? There are twenty two process areas, five maturity levels and four capability levels. Figure 1 shows the continuous representation capability levels aligned with the staged representation maturity levels [2]



Level	Continuous Representation Capability Levels	Staged Representation Maturity Levels
Level 0	Incomplete	
Level 1	Performed	Initial
Level 2	Managed	Managed
Level 3	Defined	Defined
Level 4		Quantitatively Managed
Level 5		Optimizing

Figure 1: Capability and Maturity Levels

There are twenty two process areas in the CMMI-DEV. Each of these process areas has a level associated with it. Figure 2 outlines the Process areas and the maturity/capability level associated with those process areas (Each process area is also assigned a category – for more information on this refer to the CMMI-DEVC [2], Appendix A contains more information on the purpose of each process area – though many are self-evident).



Process Area	Category	Level
Causal Analysis and Resolution (CAR)	Support	5
Configuratino Management (CM)	Support	2
Decision Analysis and Resolution (DAR)	Support	3
Integrated Project Management (IPM)	Project Management	3
Measurement and Analysis (MA)	Support	2
Organizational Process Definition (OPD)	Process Management	3
Organizational Process Focus (OPF)	Process Management	3
Organizational Performance Management (OPM	Process Management	5
Organizational Process Performance (OPP)	Process Management	4
Organizational Training (OT)	Process Management	3
Product Integration (PI)	Engineering	3
Project Monitoring and Control	Project Management	2
Project Planning (PP)	Project Management	2
Process and Product Quality Assurance (IPPQA)	Support	2
Quantitative Project Management (QPM)`	Project Management	4
Requirements Development (RD)	Engineering	3
Requirements Management (REQM)	Project Management	2
Risk Management (RSKM(	Project Management	3
Supplier Agreement Management (SAM)	Project Management	2
Technical Solution (TS)	Emgineering	3
Validation (VAL)	Engineering	3
Verification (VER)	Engineering	3

Figure 2 CMMI-DEV process areas and associated levels

To reach a maturity level an organization has to master all of the process areas associated with that maturity level as well as all the process areas associated with lower maturity levels. An organization that is only interested in certain process areas will, upon mastering a specific process area will have achieved a capability level for that process area based on the capability level of that process area. To make this a bit clearer:

- An organization wishing to reach a maturity level of 3 via a staged representation would need to master the following process areas
  - Configuration Management (2)
  - Measurement and Analysis (2)
  - Project Monitoring and Control (2)
  - Project Planning (2)
  - Process and Product Quality Assurance (2)
  - Requirements Management (2)
  - Decision Analysis and Resolution (3)
  - Integrated Project Management (3)
  - Organizational Process Definition (3)
  - Organizational Process Focus (3)
  - Organizational Training (3)
  - Product Integration (3)
  - Requirements Development (3)
  - Risk Management (3)
  - Supplier Agreement Management (3)
  - Technical Solutions (3)
  - Validation (3)
  - Verification (3)
- An organization that was not concerned about appraisals but wanted to demonstrate improved capability in Requirements Development, Technical Solutions and Risk Management would upon mastering these process areas would be:
  - Capability Level 3 for Requirements Development
  - Capability Level 3 for Technical Solutions
  - Capability Level 3 for Risk Management

The way to master a specific process area is to master both the Generic goals of all process areas and the specific goals of the process area of focus. Generic goals are goals associated with all process areas. Specific goals are specific to the process area in question. For each generic and specific goal the CMMI has generic and specific practices, respectively, that are intended as recommendations on what needs to be done to achieve that goal if the organization does not have other processes or practices in place. Briefly the generic goals are (consult the CMMI-DEV for more information)

- GG1 – The specific goals of the process area are supported by the process transforming identifiable input work products into identifiable output work products
- GG2 – The process is institutionalized as a managed process - the process is managed and performed per policy (Appendix B contains an elaboration of this generic goal for each of the specific processes)

- The process is institutionalized as a defined process - the process is defined at an organizational level and is used by projects tailored to the project based on organizational tailoring guidelines

Each process area then has a set of specific processes that must be achieved in addition to the generic processes in order to master that process area. Examples of specific goals include:

- For Configuration Management
    - SG1 – Baselines of identified work products are established
    - SG2 – Changes to the work products under configuration management are tracked and controlled
    - SG 3 – Integrity of baselines are established and maintained
- For Measurement and Analysis
    - SG1 – Measurement objectives and activities are aligned with identified information needs and objectives
    - SG2 – Measurement results which address identified information needs and objectives are provided

As with the generic goals, the specific goals are accompanied by specific practices which are intended as guidance not mandate as to how to master that specific goal. The only thing mandatory to achieve a maturity or capability level in a process area is to demonstrate that the generic and specific goals have been mastered.

Agile

The notion that it might be smart to build software in iterative or incremental chunks did not emerge with the *Agile Manifesto*. Iterative and Incremental Design and Development (IIDD) has been around for close to 80 years. The practice of time boxed product development cycles has been practiced by smart engineers from the DoD, NASA, the US Air Force and elsewhere. In 1976, Tom Gilb in his book, *Software Metrics* argued that evolutionary development resulted in superior software [3].

The software industry, being much less mature than the hardware industry, suffered significant starts and stops through its early years and into the present. At its inception, software development was not treated like an engineering discipline but rather some combination of art and science with a bit of black magic blended in. Early coders had to solve problems programmatically but also had to deal with many less academic issues associated with the logistics of software development. The advent of the personal computer and the improvement in software development environments made it possible for more and more complex problems to be addressed with software. The problem was that few stopped to take a breath and think about software as a discipline. Increasingly complex software projects led to increasing instances of software project failures. The industry countered with ‘silver bullet’ solutions such as structured programming, Computer Aided Software Engineering tools, defined processes, formal methods, CMM, and many others. Then along comes the internet, significantly increasing the complexity of things that could be accomplished with software. In light of emerging technologies and growing complexity, the formal processes, methods and standards were viewed by many as an impediment to progress rather than support for projects’ productivity and quality goals. This started a focus on lightweight technologies, or what we now refer to as agile development.

In February of 2001, a group of these software development professionals sat down to talk about lightweight methodologies. The result of these discussions was the Agile Manifesto whose tenets are referred to in the introduction of this paper. Agile development processes rely on experienced, highly skilled people communicating with customers, end users, and each other to deliver software that provides the customers and end users with value for their money. Working software is delivered in short iterations that customers can review, play with and provide near real time feedback and reprioritization. This requires that both developers and consumers of software accept the reality that things will change over the course of a project and that the software that is eventually delivered most likely will not be the same as what was envisioned when the project began.

When discussing agile development it is important to understand that agile is a set of tenets. Any project that is adhering to these tenets can be considered an agile project. There are no rules or standards emerging from these tenets – they propose an overarching philosophy with which a team should approach a software development project.

A traditional software development project begins with a set of customer or end user requirements. These requirements are analyzed, an architecture and design is created, and the set of requirements are implemented, tested and delivered to the customer. With this approach it could take months or years to get to the point where there is usable software for the customer to use and evaluate. If there was misunderstanding, confusion or poor communication about the requirements it may not come to light until much software has been developed. This has led to more than a few software failures in industry.

Agile development projects develop small usable chunks of software in short periods of time, evolving the software incrementally over time without ever being too far away from releasing a usable piece of software. The end user works with the agile team so that misunderstandings and confusion around requirements can be addressed quickly and proactively. Agile teams are composed of self-organizing, cross functional software developers that expect and embrace change.

The reader should be aware that, as stated earlier, agile is a philosophy not a specific development process or set of practices. There are many forms of agile and many different ways to incorporate agile tenets into a software project. Different agile teams have different ways of applying these principles. One should think of agile as an umbrella, not a methodology. Various methodologies have been developed proposing various sets of practices to realize agile tenets. Scrum, Extreme Programming (XP), Lean and Crystal are examples of such collections of practices. Some of the popular agile practices are:

- Pair programming or other forms of peer review
- Continuous integration with automated testing
- Test Driven Development
- Daily stand up meetings or other forms of regular on-going team communication
- Co-located teams
- Small releases
- Refactoring as part of each iteration
- Customer participation with the team
- Simple design

Each of the methodologies available incorporates some or all of these practices, though terminology and specifics of practice varies from method to method and team to team. It is important to note that many organizations do not apply all of the practices but rather pick and choose those that make sense for the company culture, their customers and end users, and the type of software development projects they normally engage in.

Agile and CMMI – Can they Coexist?

To recap, CMMI-DEV is a model to help organizations improve their development process toward improving project productivity and product quality. The ultimate end goal of CMMI is to reach a level of optimization where processes are institutionalized across the organization and projects tailor these processes based on tailoring guidelines to meet their project needs. While optimization across an organization may be the ultimate goal of CMMI, it is certainly not the ultimate goal of everyone who considers using the CMMI. Many organizations and projects use parts of the CMMI to make improvements in specific areas that are important to business objective. Organizations reach a certain level of capability or maturity for a process area by satisfying all of the goals for that process area. For each process goal recommended practices are offered, though the organization is free to use any practices that sufficiently leads them to satisfy the goals. The CMMI only requires that goals associated with each process areas are achieved; the "how to" of achievement is entirely the decision of the organization.

Agile is a paradigm for software development that encourages small deliveries of usable software, customer focused interactions, flexibility combined with constant and effective communication. There is no prescribed way for an organization to 'do agile' but rather some top level tenets and some recommended practices. Organizations who follow agile tenets are being agile whether they intend to be or even recognize that they are.

Clearly – there are no inherent impediments for an agile shop to be successful with CMMI nor for an organization attempting to maintain a capability or maturity level to deploy agile projects. CMMI tells an organization "what" to do while agile prescribes a specific (per project) set of practices associated with "how" to get it done. So why is there so much skepticism about the two playing nicely together? In a large part the problems are in perception, implementation, misunderstanding and cultural issues.

There appears to be a great deal of misunderstanding in both the agile communities and the CMM/CMMI communities as to what it means to have a CMMI level or what it means to be agile. Many in the CMMI community believe that agile is an excuse to abandon discipline and process. Many in the agile community believe that a CMMI exercise will burden development teams with an unnecessary amount of paperwork. In theory both of these things are not true but not only was that not always the case – it still is the case in many places – mostly where either CMMI or agile or both have been misinterpreted. These misperceptions within both the agile and CMMI communities fuel the notion that the two are incompatible. In order for the marriage of CMMI and agile to be successful, it is very important that the agile team is educated as to specifically what CMMI is and the CMMI implementation team needs to understand agile. Understanding will enable cooperation and successful collaborations toward process goals.

CMMI can be implemented in such a way that it exploits existing agile practices to reach process area goals. Project planning does not need to be done in one big phase, but rather it can be done as the project progresses. It's possible that adding a small amount of formality to the planning poker sessions is all that is required to meet certain Project Planning process area goals. Project Management and Control goals may be accomplished through reports at the daily standup meeting or possibly the retrospective at the end of an iteration. Requirements Management in agile teams is often accomplished using a tool that keeps track of user stories and the backlog. Agile teams generally tend to collect data around each iteration – number of story points completed, team velocity, burn down, etc. These measurements can be used to support goals relating to measurement, estimation or causal analysis. This is not to say that an agile team will not have to compromise in some areas, there will most likely be the need to make some modifications to ensure existing practices are considered communicated, defined and eventually institutionalized.

Remember that CMMI is focused on top down process initiatives aimed at making improvements that benefit the entire organization. Agile on the other hand has a very project focused set of practices that is very bottoms up and applies to what the team is doing day to day. In many organizations agile is applied completely different in different teams across the organization. An organization that seeks to elevate their agile development projects to a specific capability or maturity level needs to recognize that agile practices need to be institutionalized to some level. While this sounds like it will send all the real 'agelists' running for the door – this is not necessarily so. The ultimate goal of CMMI is for an organization to have development processes, practices and methodologies that can be used, with tailoring, across the organization. Institutionalization done correctly means that every agile team in the organization stands to benefit from the best practices developed and enforced by all the other agile teams in the organization. Not every agile implementation needs to be the same, they just need to follow some basic set of rules.

It is possible for an organization to implement CMMI-Dev in a way that would make it very hard for projects to be agile. An organization that interprets the Generic and Specific practices to be gospel rather than guidance is likely to implement a documentation heavy, rigid implementation of CMMI. In this case integrating agile practices that prefer communication over documentation and flexibility over rigid requirements would be problematic. But an organization who incorporates agile friendly practices to achieve CMMI process area goals could be quite successful. Organizations wishing to marry agile with CMMI should make sure that the practices they use to ensure achieving a process area goal are in line with practices they use in their daily work day and are referred to using terminology the organization is familiar with.

Agile & CMMI in the real world

There's two ways to think about this. How can we achieve CMMI levels when we are agile and how can we use what agile teaches us to achieve CMMI maturity. Both are interesting questions. Let's take the case of DTE Energy [5],[6] a diversified energy company involved in energy related businesses and services worldwide. Because there are different regulatory and environmental issues associated with the different arms of the business the delivery of IT-based products and services is complicated for DTE Energy. This organization started down a long road towards process maturity in the early 2000's. At this time, while many of the organization's project teams used a traditional waterfall based "V" approach to attack their projects, there was a small but growing group of project teams that had embraced agile practices quite successfully. The organization decided to seek CMM level maturity 2 (CMMI was not quite there yet). They put together a process improvement team – authoring standards and templates and providing training on these. After three years there was significant disappointment with the progress. They eventually came to realize that their interpretation of the CMM was taken too literally. Rather than looking to see how existing organizational processes, practices and methodologies could be tweaked to align with Key Process Area Goals, they sat down with the CMM model documentation and sought to enforce it. The effects of this common misinterpretation of 'how' to do CMM was exacerbated in situations where it was being applied to the agile groups.

Over time as the industry grew to increase the number of unregulated entities supported, the gap between the 'agilests' and the process improvement people widened. The organization took a step back from the day to day issues and decided to form a Software Engineering Process Group (SEPG) which brought all the stakeholders to the table to determine how they could work and play nicely. CMM had taken a back seat during this period of time but with the increasing successes of the SEPG and with a new updated CMMI, the organization decided to seek a Level 3 assessment. Interestingly, the organization, through the SEPG, determined that the best approach to achieving Level 3 assessment was to attack it in an agile fashion. One of their findings was that agile methodologies –proven to deliver successful software projects – also created a quicker "fail first" and "just enough" process mindset.

A CMMI assessment can be applied to a subset of an organization (known in CMMI terminology as an "Organizational Unit"). In this case they decided to define their Organizational Unit as projects that exhibited specific behaviors – one of which was adherence to an agile methodology. The process improvement team put together a plan for deployment of the CMMI using the following releases:

- Project planning and timing
- Release 1 – focus on ten Level 3 process areas, test with an informal Class B appraisal
- Release 2 – focus on six level 2 process areas, plus priority Release 1 defects , test with an informal class B appraisal
- Release 3 – focus on 16 Level 2 and 3 process areas, plus priority Release 2 defects, test with 2 formal Class A appraisals

Like any good agile project, releases were broken into iterations and were followed with a retrospective to discuss what went wrong and what went right (to improve process improvement processes). Release 1 went well with informal appraisal results well within what the team expected. With Release 2 came a snag. External influences caused some of the projects targeted for the informal class B appraisal to start late or not start at all – leaving the process group with not enough projects far enough along to do a reasonable informal appraisal. They either had to postpone the appraisal or extend the Organizational Unit to include more traditional projects as well. They decided to expand the Organizational Unit. This, of course, required revisiting some of the process assets to ensure that they could be interpreted correctly in non-agile projects. They found that by expanding their CM framework to include assets beyond traditional agile work products, they could support tailoring to account for other methodologies.

Some lessons learned from this transition

- Understand why you are pursuing process improvement, share that vision with the entire organization

- Based on that vision determine who is best to perform improvement initiatives – experts from the outside or home grown teams that will go the distance
- Create clarity as to how you will develop, deploy and maintain process improvements

Another case where there is synergy in the quest to be agile and the quest towards capability maturity levels can be found In [4], an organization that had achieved a CMMI maturity level 5 discusses the challenges of applying the process areas to a SCRUM development project. Because with agile there are no clear and distinct boundaries between lifecycle phases it was impossible to model against lifecycle phases, as they had done with more traditional projects. They chose to use Discreet Event Simulation to model the entire lifecycle within each sprint. Over time, they collected data within each sprint and were able to perform Causal Analysis and Resolution processes whose feedback was incorporated into the end of sprint retrospectives. This is just one example of how they chose to use and enhance existing agile practices to achieve CMMI process area maturity goals. By being flexible and sensible about the CMMI process area goals, this team was able to exceed sprint velocity goals, plan sprints and releases more effectively, maintain quality, increase productivity and create a high level of team satisfaction. The team also found that because agile provides for data collection with each sprint, they were able to see results of the level 5 analysis sooner than with more traditional processes.

Conclusions

CMMI is a model designed to help organizations achieve and institutionalize process maturity. CMMI specifies the goals that need to be achieved to reach a specific maturity or capability level – it is neither rigid nor prescriptive with how exactly these goals are to be met. Agile is a philosophy and set of tenets for software projects that are characterized by highly collaborative, cross-functional teams who work closely with their customers to deliver regular increments of functional software capability that the customers and end users are happy with. Neither the agile philosophy nor the CMMI is prescriptive as to how the software is developed, though a specific implementation of agile will come with a set of prescribed practices.

There is no reason the two cannot be successfully applied in tandem if this is done with careful thought a deliberation. Failures have been noted but most of these are associated with one of the following situations:

- CMMI implementation is an exact interpretation of the generic and specific practices guidelines from the CMMI manual – creating a document heavy, rigid interpretation of CMMI
- Agile implementation is not so much agile as it is a wild west interpretation of agile where there is no planning, process or oversight
- CMMI team has had a bad experience with agile
- Agile team has had a bad experience with CMMI (or CMM)
- Organizational culture creates barrios to prevent either the adoption of agile or adoption of CMMI

Agile organizations thinking of achieving CMMI Maturity or Capability levels should understand the process area goals and give serious thought to the agile practices in place which might (with or without some modification) accomplish those goals. There is nothing in the CMMI that says the product backlog currently being tracked doesn't suitably accomplish one or more of the Requirements Management

Process Area Goals. Agile organizations should also think about how the agile approach could be applied to achieving process goals.

Organizations with a CMMI level that are thinking of launching agile projects should similarly think about where the synergies might lie. As an organization with institutionalized processes – this thought process should revolve around what new and different artifacts an agile process might result in and how the existing process tailoring guidance should be modified to accommodate for new artifacts and procedures.

When agile and CMMI are implemented sensibly and with a favorable nod to the practices and intent of each other, the results can be quite successful. While on the face they seem like competing notions, at the heart the agile project teams and the CMMI process teams are both on a quest to improve the productivity of their software projects and the quality of the products they deliver. When examined at a deeper level it is easy to see that there is a great deal of synergy between the two and when this synergy is realized and exploited organizations will be able to optimize their software development processes.

References

[1] http://agilemanifesto.org/ (retrieved 3/2017)
[2] Software Engineering Institute – CMMI Product Team, "CMMI® for Development Version 1.3", November 2010, CMU/SEI-2010-TR-033, available at https://resources.sei.cmu.edu/asset_files/TechnicalReport/2010_005_001_15287.pdf (retrieved 3/2017)
[3] Hillel, G., et.al., "CMMI® or Agile: Why Not Embrace Both", November 2008, CMU/SEI-2008-TN-003,available at http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=8533, (retrieved 3/2017)
[4] McMahon, P., "Case Studies Using Agile Methods and the CMMI Framework Together", Proceedings IEEE Software Technology Conference 2007, Long Beach, CA, available at: http://www.ieee-stc.org/proceedings/2007/pdfs/pem1633.pdf (retrieved 3/2017)
[5] Baker, S., "Formalizing Agility: An Agile Organization's Journey toward CMMI® Accreditation", Proceedings of the Agile Development Conference 2005 (IEEE), 2005
[6] Baker, S., "Formalizing Agility, Part 2: How an Agile Organizations Embraced the CMMI®, Proceedings of AGILE 2006 Conference (IEEE), 2006

Good References for aligning agile practices to CMMI goals

http://cmmiinstitute.com/sites/default/files/resource_asset/Effective%20CMMI%20Implementation%20in%20Agile%20Environment_Atos_SEPGNA14.pdf
https://www.slideshare.net/systematicsoftware/mature-agile-to-manage-complex-projects-or-products-v1-8
http://www.entinex.com/AgileLifeCycleCMMIClues.pdf

Appendix A – Purpose of each Process Area from [2]

Causal Analysis and Resolution (Level 5)

- The purpose of CAR is to identify causes of selected outcomes and take action to improve process performance.

Configuration Management (Level 2)

- The purpose of CM is to establish and maintain the integrity of work products using configuration identification, configuration control and configuration status accounting and configuration audits. Work products placed under CM include the products that are delivered to the customers, designated internal work products, acquired products, tools and other items used in creating and describing these work products. This process area applies not only to CM on projects but also to CM of organizational work products such as standards and procedures.

Decision Analysis and Resolution (Level 3)

- The purpose of DAR is to analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria. While the primary application of this process area is to technical concerts, formal evaluation processes can be applied to many nontechnical issues as well – especially when a project is being planned.

Integrated Project Management (Level 3)

- The purpose of IPM is to establish and manage the project and the involvement of relevant stakeholders according to an integrated and defined process that is tailored from the organization's set of standards.

Measurement and Analysis (Level 2)

- The purpose of MA is to develop and sustain a measurement capability used to support management information needs.

Organizational Process Definition (Level 3)

- The purpose of OPD is to establish and maintain a usable set of organizational process assets, work environment standards and rules and guidelines for teams.

Organizational Process Focus (Level 3)

- The purpose of OPF is to plan, implement and deploy organizational process improvements based on a thorough understanding of current strengths and weaknesses of the organizations processes and process assets.

Organizational Performance Management (Level 5)

- The purpose of OPM is to proactively manage the organizations performance to meet its business objectives. Improvement refers to all ideas that would change the organizations processes, technologies, and performance to better meet the organizations business objective and associated quality and performance objectives. This process area extends OPF practices by focusing on process improvement based on a quantitative understanding of the organization set of standard processes and technologies and their expected quality and process performance.

Organizational Process Performance (Level 4)

- The purpose of OPP is to establish and maintain a quantitative understanding of the performance of selected processes in the organizations set of standard processes in support of achieving quality and process performance objectives. This process area provides performance data, baselines, and models to quantitatively manage the organization projects.

Organizational Training (Level 3)

- The purpose of OT is to develop skills and knowledge of people so that they can perform their roles effectively and efficiently.

Product Integration (Level 3)

- The purpose of PI is to assemble the product from the product components, ensure that the product, as integrated, behaves properly (i.e. possesses the required functionality and quality attributes) and deliver the product.

Project Monitoring and Control (Level 2)

- The purpose of PMC is to provide an understanding of the projects progress so that appropriate corrective actions can be taken when the projects performance deviates significantly from the plan.

Project Planning

- The purpose of PP is to establish and maintain plans that define project activities.

Process and Product Quality Assurance (Level 2)

- The purpose of PPQA is to provide staff and management with objective insight into processes and associated work products. The practices in this area ensure that planned processes are implemented while the practices in the Verification process area ensure that specified requirements are satisfied.

Quantitative Project Management (Level 4)

- The purpose of QPM is to quantitatively manage the project to achieve the projects' established quality and process performance objectives. This process area, unlike Project Management practices, helps to develop a quantitative understanding of the expected performance of processes or sub processes.

Requirements Development (Level 3)

- The purpose of RD is to elicit, analyze, and establish customer, product and product component requirements.

Requirements Management (Level 2)

- The purpose of REQM is to manage requirements of the projects products and product components and to ensure alignment between those requirements and the projects plans and work products.

Risk Management (Level 3)

- The purpose of RISM is to identify potential problems before they occur so that risk handling activities can be planned and invoked as needed across the life of the product or project to mitigate the adverse impacts on achieving objectives.

Supplier Agreement Management (Level 2)

- The purpose of SAM is to manage the acquisition of products and services from suppliers.

Technical Solution (Level 3)

- The purpose of TS is to select, design, and implement solutions to requirements.

Validation (Level 3)

- The purpose of VAL is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment. Validation ensures that you built the right thing while verification ensures that you built it right

Verification (Level 3)

- The purpose of VER is to ensure that selected work products meet their specified requirements.

Appendix B – Elaboration as to what Generic Goal 2 means for each process area [2]

CAR elaboration – policy establishes organizational expectations for identifying and systematically addressing causal analysis of selected outcomes

CM Elaboration – policy establishes organizational expectations for establishing and maintaining baselines, tracking and controlling changes to work products, and establishing and maintaining integrity of baselines

DAR Elaboration – policy establishes organizational expectations for selectively analyzing possible decisions using a formal evaluation process that evaluated identified alternatives against established criteria – policy should also provide guidance on which decisions require formal evaluation process

IPM Elaboration – policy establishes organizational expectations for establishing and maintaining the projects defined processes from project startup through the life of the project, using the projects defined process in managing the project and coordinating and collaborating with relevant stakeholders

MA Elaboration – policy establishes organizational expectations for aligning measurement objectives and activities with identified information needs and project, organizational, or business objectives and for providing measurement results

OPD Elaboration – policy establishes org expectations for establishing and maintaining a set of standard processes for use by the organization, making org process assets available across the org and establishing rules and guidelines for using them

OPF Elaboration – policy establishes or expectations for determining process improvement opportunities for the processes being used and for planning, implementing and deploying process improvements across the organization

OPM Elaboration – process establishes org expectations for analyzing the orgs business performance using statistical and other quantitative techniques to determine performance shortfalls, and identifying and deploying process and technology improvements that contribute to meeting quality and performance objectives

OPP Elaboration - process establishes org expectations for establishing and maintaining process performance models for the organization's set of standard processes

OT Elaboration – policy establishes org expectations for identifying the strategic training needs of the organization and providing that training

PI Elaboration – policy establishes org expectation for developing product integration strategies, procedures, and an environment; ensuring interface compatibility among product components; assembling the product components and deliver the product and product components

PMC Elaboration – expectations for monitoring project progress and performance against the project plan and managing correction action to closure when actual results deviate significantly from the plan

PP Elaboration – expectations for estimating the planning parameters, making internal and external commitments, and developing the plan for managing the projects

PPQA Elaboration – expectations for objectively evaluating whether processes and associated work products adhere to applicable process descriptions, standards, and procedures; ensuring that noncompliance is addressed

QPM Elaboration – expectations for using statistical and other quantitative techniques and historical data when; establishing quality and process performance objectives, composing the projects defined process, selecting sub process attributes critical to understanding process performance, monitoring sub process and project performance and performing root cause analysis to address process performance deficiencies

RD Elaboration - expectations for collecting stakeholder needs, formulating product and product components requirements and analyzing and validating those requirements.

REQM Elaboration – expectations for managing requirements and identifying inconsistencies between the requirements and the project plans and work products

RSKM Elaboration – expectations for defining a risk management strategy and identifying, analyzing and mitigating risks

SAM Elaboration – expectations for establishing, maintaining and satisfying supplier agreements

TS Elaboration – expectations for addressing the iterative cycle in which product o product component solutions are selected, designs are developed and implemented

VAL Elaboration – expectations for selecting products and product components for validation; for selecting validation methods, and for establishing and maintaining validation procedures, criteria, and environments that ensure the products and product components satisfy end user needs in their intended operating environment

VER Elaboration – expectations for establishing and maintaining verification methods, procedures, criteria and the verification environment, as well as performing peer reviews and verifying selected work products