Contrary to what you may have heard, organizations around the world are applying agile and CMMI together, and some are doing it effectively. Sadly, some are not as successful as they would like to be. In this article, I start by sharing some important results from several surveys that I’ve run over the years that explore how people are applying agile and CMMI in practice. Second, I address some of the debilitating rhetoric still swirling around this topic. I then describe some prerequisites for getting agile and CMMI to work together and end by making the case for a disciplined agile CMMI strategy.

SOME INDUSTRY STATS

Let’s begin with a quick discussion of what’s actually happening out there. The Dr. Dobb’s Journal (DDJ) January 2008 Process Framework Survey explored whether people were combining agile and CMMI together. Of the respondents who said their organizations were doing agile:

- 10% indicated that they were exclusively doing CMMI-compliant agile projects.
- 45% said they were exclusively doing non-CMMI agile projects.
- 45% said that their organization had both CMMI-compliant and non-CMMI agile projects.

The same survey found that agile teams in CMMI-compliant environments statistically had the same success rates as those in non-CMMI environments. It also found that when given the choice, most people prefer repeatable results over repeatable processes, contrary to the mantra of some CMMI proponents.

To summarize, of the survey respondents whose organizations were doing agile, 55% had one or more teams doing so within the scope of CMMI compliance. CMMI was neither helping nor hindering agile project teams on average, and repeatable processes aren’t as attractive as some would lead us to believe.

I have to say, I was uncomfortable with the 55% figure. Although the 2008 survey revealed that many organizations were applying agile and CMMI together, it didn’t determine what percentage of teams were combining them. For example, think of a company that has 20 agile project teams and only one of them is doing CMMI too. Yes, the company is doing agile and CMMI together, but only with 5% of its agile teams.

About a year and a half later, I explored the agile and CMMI issue again in a new survey that I targeted to the agile community itself. The Ambyssoft July 2009 Agile Practices survey asked several questions, one of which was whether the person’s existing agile team was currently working in a CMMI-compliant manner. Of the respondents who indicated they were currently on an agile team, 9% said that they were working on projects that were CMMI-compliant. This result — that roughly 1 in 10 agile teams need to be CMMI compliant — seems a lot more realistic to me.

This prompts the question of how easy it is to combine agile and CMMI. In the DDJ Summer 2012 State of the IT Union Survey, I touched on this issue. The goal of the survey was to explore whether organizations were successful or unsuccessful at various levels of the scaling factors called out in the Agile Scaling Model (ASM). One of the ASM scaling factors is regulatory compliance, which includes both legal compliance, such as adhering to US Food and Drug Administration (FDA) regulations, and self-imposed compliance, such as conforming to CMMI or ISO 900X. The survey found that of the respondents whose organizations had achieved success applying agile techniques in practice, 44% said that one or more of their project teams had done so under self-imposed compliance requirements. Of the respondents whose organizations had experienced one or more failed agile projects, 30% indicated that one or more of their projects had self-imposed compliance requirements. Unfortunately, the survey did not ask about CMMI compliance directly, although CMMI was used as one example of self-imposed compliance.

The survey results lead me to three important observations. First and foremost, people are in fact successfully applying agile and CMMI together. Second, it can be a rocky road when doing so, and some organizations are running into problems. Three, there isn’t any blatantly
obvious evidence for or against applying the two together. Granted, this third observation is based on averages — your organization may have very good reasons to apply the two together. More on this shortly.

ADDRESSING THE RHETORIC

You don’t have to spend much time online to discover that when it comes to applying agile and CMMI together, there is some questionable rhetoric being bandied about. I feel it’s important to surface this rhetoric and describe the reality of the situation. Common misperceptions about agile CMMI include:

- **Agile and CMMI are incompatible.** This is clearly not the case, as we learn from the aforementioned surveys as well as from several publications about the topic. From what I’ve seen, most of this problem stems from agile protagonists not understanding CMMI and CMMI protagonists not understanding agile.

- **Scrum is CMMI Level 5.** Scrum is a very, very small part of what you need to do to succeed at agile. Scrum’s focus is on some aspects of project leadership and requirements management, and it relies on other methodologies such as Extreme Programming (XP), Agile Modeling (AM), the Unified Process (UP), and many others to fill in the blanks. Yes, Scrum can be used in CMMI environments, but Scrum on its own clearly doesn’t even address all CMMI Level 2 issues let alone those at the higher levels. By the same token, agile methods such as XP and AM can just as easily be applied in CMMI environments to address portions of one or more process areas (PAs) in an agile manner.

- **CMMI doesn’t add value.** Empirically you can demonstrate that this is not the case by simply hopping on a flight to Bangalore to see how the Indian IT service providers have leveraged CMMI into a multibillion-dollar industry. Furthermore, there are numerous studies showing that as organizations move up CMMI levels, their productivity improves.

- **CMMI equals needless bureaucracy.** The way organizations address CMMI compliance is up to them. They can choose to adopt a documentation-heavy strategy (which many unfortunately do), or they can choose to adopt a more streamlined agile approach. Many agilists have had very bad experiences in heavy CMMI shops, and in many cases that is their only CMMI experience — hence the bitterness toward CMMI.

**MAKING AGILE AND CMMI WORK TOGETHER**

For agile and CMMI to coexist together effectively, in my experience four things need to occur. Organizations must:

1. Take an enterprise view of IT
2. Focus on quantifiable business value
3. Adopt a full delivery lifecycle
4. Take a hybrid approach

Let’s examine each of these success factors in greater detail.

**Take an Enterprise View of IT**

Adopting an enterprise view of IT can require an open mind of both CMMI and agile practitioners. An important implication of enterprise awareness is that different teams are in different situations; therefore they need to tailor their strategies accordingly. Enterprise professionals who deal with many project teams need to be capable of working with and governing traditional teams in a traditional manner and agile teams in an agile manner. For agilists, being enterprise-aware can be very difficult at first due to the prevailing project mindset within that community. Agilists need to become disciplined enough to leverage and enhance the existing software and data infrastructure, to follow common guidelines, to work with enterprise professionals such as enterprise architects and operations professionals, and to be governed effectively. The need for enterprise awareness is a key philosophy of Disciplined Agile Delivery (DAD), which I describe below.

**Focus on Quantifiable Business Value**

The second success factor is for your organization to focus on delivering quantifiable business value to your stakeholders in all activities that you perform. In other words, focus on real process and organization improvement and squeeze out the needless bureaucracy that is all too prevalent in CMMI environments. I have yet to evaluate a CMMI organization, including those rated at Level 4 and Level 5, that didn’t have very large opportunities for improving their productivity by adopting more agile ways of working. For example, I often run into existing development processes in which the project team creates a requirements specification, writes test plans and test cases so that the solution may be validated, and then maintains traceability between these artifacts (and more) for good measure. Yes, business value is being delivered via this process, but we can achieve the same goals while working more effectively. For example, by adopting an acceptance test–driven
approach, the acceptance tests become both the detailed
tests and the detailed requirements specification, with
full requirements-to-test traceability between them with
no extra work.8 By working smarter, not harder, you
not only reduce the work required to provide the same
business value as before, but you do so with a shorter
feedback cycle between requirements elicitation and
implementation, thereby reducing project risk.

**Adopt a Full Delivery Lifecycle**

Next, you need to adopt a full delivery lifecycle.
Minimally this includes project initiation, construction,
and deployment activities, although you should also
consider enterprise activities such as portfolio manage-
ment, enterprise architecture, asset management, oper-
ations, support, and others. Figure 1 depicts the “agile”
version of the DAD lifecycle (there are also lean and
continuous delivery lifecycles to choose from). Notice
how the lifecycle starts with project identification and
selection, a portfolio management activity. It also
includes an explicit Inception phase, sometimes erro-
noeusly referred to as Sprint 0 or Iteration 0, where
fundamental project initiation activities occur. Even
global project teams need to get started somehow. There
is, of course, an explicit Construction phase as well as a
Transition phase9 focused on deploying your solution.
After deployment into production, the work continues
with operations and support, and [hopefully] the team
is allowed to begin work on the next release.

**Take a Hybrid Approach**

The final success factor is to take a hybrid approach to
development. Many agile methodologies — including
Scrum, XP, AM, Agile Data, Kanban, and more — focus
on a subset of the activities required to deliver a solution
from project initiation to delivery. Before DAD was
developed, you needed to cobble together your own
agile methodology to get the job done. As you can see in
Figure 1, DAD clearly adopts strategies from Scrum, AM,
and the UP, and you’ll need to take my word for it that it
also adopts ideas from other sources such as XP, Kanban,
lean software development, and many more. The bottom
line is that if you intend to address all of the CMMI
process areas, you will need to adopt a hybrid approach
such as DAD or do the work to invent your own.

**DISCIPLINED AGILE DELIVERY (DAD) IN DETAIL**

I have already covered a few aspects of the DAD
process decision framework — it is a hybrid agile
framework that supports a full delivery lifecycle and
enterprise awareness — but there’s more to it than that.
For CMMI practitioners, there are several interesting
aspects about DAD that I haven’t yet described. First,
it is very flexible. Although Figure 1 depicts a basic
agile version of the lifecycle, that is only one of many
options available to you. For example, you may choose
to follow a leaner lifecycle in which Construction and
Transition activities occur when appropriate, not within
the confines of iterations and phases. Or you may want
to follow a continuous delivery version of the lifecycle
in which you release into production on a very regular
basis, perhaps even several times a day. Because differ-
ent teams will find themselves in different situations,
an effective — or dare I say mature? — process frame-
work needs to support several types of lifecycles.

Second, when it comes to process tailoring, DAD starts
in the middle, not at the extremes, as do other methods

---

**Figure 1 — Agile DAD lifecycle.**
such as the Rational Unified Process (RUP) and Scrum. Many organizations that adopted the RUP ran into trouble because its tailoring strategy assumes that individual teams will have the process expertise to tailor the very large RUP library down into something manageable. At the other end of the spectrum, many organizations that have adopted Scrum have also run into problems because Scrum starts with something very small and assumes that your teams have the process expertise to tailor it up to what you need. These strategies are doubly flawed — teams typically don’t have the required process expertise and both of these methods start at the extremes and insist that you tailor them into the middle ground. DAD starts in the middle ground.

DAD is based on the assumption that project teams are unlikely to have the process expertise to tailor their process effectively even though they will need to. This leads to the third aspect of DAD that CMMI practitioners will find intriguing — it is goal driven, not prescriptive. Rather than prescribing certain practices or activities, the DAD process instead suggests that your team needs to address certain goals, that for each goal there are several issues to contemplate, that for each issue you have a number of techniques you can employ to address the issue, and that each technique has its strengths and weaknesses so you will need to choose the one that’s best suited for your situation. The goals of the DAD framework are depicted in Table 1. Notice how some are applicable for a given project phase and some are applicable throughout the entire project lifecycle.

What “Goal-Driven” Looks Like

To understand the difference between prescriptive and goal-driven approaches, let’s compare similar aspects of Scrum and DAD. Scrum prescribes that you manage your work via a queue called a product backlog whose items are ordered by business value. DAD, on the other hand, suggests that you address changing stakeholder needs in some manner and that one of the associated issues is choosing a strategy to prioritize your work. It further indicates that strategies could be driven by several factors — business value, risk, due date, dependencies — and that strategies may in fact be combined. DAD goes further yet to describe the tradeoffs associated with each strategy and provides advice for when each strategy is viable. The implication is that, in some situations, Scrum’s product backlog strategy is appropriate, but the RUP’s risk-value approach might be better for some of your teams, or perhaps a combination of prioritization strategies is best. Rather than obsessing over repeatable processes, organizations that want to achieve repeatable results — such as producing quality solutions that meet the needs of their stakeholders and spending their IT budget wisely — will want delivery teams that manage changing stakeholder needs effectively. Because different teams are in different situations, each team may do this differently.

Let’s dive a bit deeper into the workings of a goal-driven strategy. Figure 2 depicts the overview diagram for the Construction phase goal Move Closer to a Deployable Release. To fulfill this goal, you need to address several issues, such as adopting a deployment strategy and a validation strategy. For each strategy, there are several techniques that you may choose to adopt. In some cases the techniques may be combined, and in some cases it wouldn’t make sense to do so. For example, to validate your solution, you might apply both developer and acceptance test–driven development as well as continuous integration (CI) and parallel independent testing. For verification, you might

<table>
<thead>
<tr>
<th>Goals for Inception Phase</th>
<th>Goals for Construction Phase Iterations</th>
<th>Goals for Transition Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Form Initial Team</td>
<td>• Produce Potentially Consumable Solution</td>
<td>• Ensure Solution Is Consumable</td>
</tr>
<tr>
<td>• Develop Common Project Vision</td>
<td>• Address Changing Stakeholder Needs</td>
<td>• Deploy Solution</td>
</tr>
<tr>
<td>• Align with Enterprise Direction</td>
<td>• Move Closer to Deployable Release</td>
<td></td>
</tr>
<tr>
<td>• Explore Initial Scope</td>
<td>• Improve Quality</td>
<td></td>
</tr>
<tr>
<td>• Identify Initial Technical Strategy</td>
<td>• Prove Architecture Early</td>
<td></td>
</tr>
<tr>
<td>• Develop Initial Release Plan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Form Work Environment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Secure Funding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Identify Risks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Ongoing Goals | | |
|----------------|----------------|
| • Fulfill Project Mission | • Improve Team Process and Environment |
| • Grow Team Members | • Leverage/Enhance Existing Infrastructure |
| • Address Risk | | |
adopt non-solo techniques such as pair programming but not invest time in other types of reviews (either formal or informal) yet still adopt both static and dynamic analysis techniques. The point is that an individual team will need to address this goal and its issues, but it may do so using a unique combination of techniques depending on its situation.

Reconciling DAD and CMMI

So ... how do DAD and CMMI fit together? Some of the CMMI process areas\textsuperscript{10} are subsumed in a single DAD goal. For example, CMMI’s Configuration Management PA maps to the Asset Management issue of the Move Closer to a Deployable Release goal. The Requirements Development PA, on the other hand, is addressed by two DAD goals, in this case Explore Initial Scope and Address Changing Stakeholder Needs. And while the DAD goals don’t directly address the Organizational Training PA, they do address it indirectly through DAD’s People First aspect, which underlies the overall DAD framework.

One challenge with directly mapping CMMI to DAD is that the underlying assumptions between the two are not identical. For example, the CMMI’s Requirements Management (REQM) PA states that an aspect of REQM is bidirectional traceability. DAD, on the other hand, adopts the AM and XP philosophy that a particular effort, such as investing in bidirectional traceability, should only be undertaken when it adds value for one or more stakeholders and the stakeholders are aware of its ongoing maintenance costs and they are willing to pay those costs. In other words, maintain traceability because it makes sense to do so, not just because a self-imposed regulation tells you to.

It is valid to ask whether DAD’s goal-driven approach forms the basis of an agile maturity model. I would be reticent to go down this path. Consider the Move Closer to a Deployable Release goal overview diagram of Figure 2 again. Some of the techniques associated with some issues are in fact ordered according to something resembling “maturity.” For example, the Documentation Strategy issue has techniques such as Continuous Documentation (Same Iteration), Continuous Documentation (Following Iteration), Document Late, and None (at all). Ignoring the orthogonal technique of Active Stakeholder Participation, these techniques are

![Figure 2 — Goal overview diagram for Move Closer to a Deployable Release.](image-url)
presented in order of most mature to least mature. In contrast, consider the Validation issue, which has a variety of testing strategies that could be used in combination and that require various levels of maturity. Perhaps it would be possible to refactor this issue into two or more smaller issues for which a maturity ordering makes sense, but there are other issues in other goals where such refactoring wouldn’t be a viable option.

PARTING THOUGHTS

We can observe that organizations are succeeding at applying agile and CMMI in practice. We can also observe that there is significant motivation to do so for some organizations, regardless of the rhetoric of either the CMMI or agile camps. If we’re going to help make the adoption of “agile CMMI” smoother, we will need to focus on several things. First, we must start with respectful and meaningful conversations, and that isn’t always happening. Second, CMMI advocates need to open their minds to agile ways of working that still achieve the same goals, albeit in different, more effective ways. Third, agile advocates need to be open to the idea that CMMI can in fact add some real value in their environments. Fourth, we must evolve CMMI so that it truly focuses on specifying what a process should address without bringing process design issues into the conversation. Fifth, agile must evolve to reflect a more mature way of working, something that I feel has been achieved with Disciplined Agile Delivery.

I believe the evidence is very clear that agile approaches to working prove to be more effective in practice than traditional ones. I also believe that for some organizations there are compelling reasons to adopt CMMI. Having said that, let me leave you with this thought: If you’re going to be CMMI compliant, wouldn’t you rather take an agile approach — or better yet, a disciplined agile approach — when doing so?

ENDNOTES

9The Transition “phase” eventually evolves into an activity with the adoption of continuous delivery practices.

Scott W. Ambler is a Senior Consultant with Cutter Consortium’s Agile Product & Project Management and Business & Enterprise Architecture practices. Mr. Ambler works with organizations around the world to help them to improve their software processes. He provides training, coaching, and mentoring in Disciplined Agile and Lean strategies at both the project and organizational level. Mr. Ambler is the founder of the Agile Modeling (AM), Agile Data (AD), Disciplined Agile Delivery (DAD), and Enterprise Unified Process (EUP) methodologies. He is the author or coauthor of 20 books, including Disciplined Agile Delivery, Refactoring Databases, Agile Modeling, Agile Database Techniques, The Object Primer 3rd Edition, and The Enterprise Unified Process. Mr. Ambler is a Senior Contributing Editor with Dr. Dobbs’s Journal. He can be reached at sambler@cutter.com; website ScottWAmbler.com.