SYM-AM-24-033

Excerpt from the Proceedings of the Twenty-First Annual Acquisition Research Symposium

Acquisition Research:
Creating Synergy for Informed Change

May 8–9, 2024

Published: April 30, 2024

The Value of an Agile Approach to Independent Verification and Validation (IV&V) for Acquisition

Justin Smith—is a member of the Agile Transformation Team at the Software Engineering Institute (SEI) of Carnegie Mellon University. Smith currently supports several customers within the Department of Defense for SEI and is focused on coaching leaders and teams through Agile adoption in areas both internal and external of software development. Prior to SEI, Smith spent 13 years as both a Contractor and Civil Servant for NASA. At NASA, Smith had various roles focused on leadership & development across the agency as well as supporting independent verification & validation (IV&V) projects. [jlsmith@sei.cmu.edu]

Abstract

IV&V is common in both the public and private sector and can range from a risk mitigation strategy to a requirement levied by a safety board or law. IV&V is performed by a team of analysts independent from the development team. The end goal of a second set of eyes on the system under development is to ensure the system satisfies its mission objectives and add assurance of mission success. This paper focuses on addressing the challenges that come with providing software IV&V for cyber-physical systems being developed for the federal government using an agile implementation. Another focus area this paper will address is how an Agile IV&V approach not only allows the IV&V team to be more in phase with the software development activity, providing more focused and timely assurance, and it reinforces the focus on rapid incremental delivery of working products by the enterprise. Observations from Agile IV&V implementations in the federal government and the DoD will be provided, demonstrating the value this Agile IV&V approach provides the program office and a reason to view IV&V as more of an informative partner rather than a burden on the program or as a necessary requirement of the acquisition process. This paper will also highlight how program offices can leverage the beneficial perspective of IV&V to influence the future in a proactive and timely manner rather than correct the past.

Key words: Agile Software Development, Independent Verification and Validation, Risk Mitigation, Technical Debt

Background

Software is often one of the most important pieces of the complex cyber-physical systems that the world is constantly designing and building. This paper will highlight the importance of software assurance on those systems and help the decision makers understand the ever-changing risks associated with software development.

Independent verification and validation (IV&V) is a form of software risk mitigation which has been around for decades in both the public and private sector. Projects in these sectors could either choose to perform IV&V of their software through their own volition or be told that IV&V will be performed on the project to satisfy a law or safety requirement. IV&V will provide valuable insight into the quality of the software and even insight into the development process, depending on what the program is looking for in their relationship with IV&V. No matter the reason for why IV&V is being performed, this paper aims to help the reader get the most out of the information that the IV&V team is providing throughout the interaction with the project or program.

IV&V has been rather commonplace in the federal government for several decades. Most of the instances in the Department of Defense (DoD) and National Aeronautics and Space Administration (NASA) are instituted from a safety perspective, usually mandatory by statutes dictating the use of IV&V either internally to the agency, or externally from a commercial provider. This "forced relationship" between IV&V and the developer usually sets the tone for how the experience will go. It doesn't mean that there is no strain on the relationship between IV&V and the developer if the program office voluntarily has chosen to do IV&V; it just provides an additional excuse for the relationship to sour if the developer can lean on the fact that they "were forced to participate" with IV&V.

This is where the role of the acquisition organization, the program office, can help set the tone for how the developer will interact with the IV&V agent. IV&V has an incredible role in acquisitions. The IV&V team will be able to provide the program office an unbiased view of what is happening at the software level. This insight the IV&V team provides will highlight where the software is doing what it is supposed to be doing well and identify issues that may be preventing it from doing so. IV&V can highlight risks in the development process or in the software itself and whether the software is satisfying the requirements the program office set in place. All this work boils down to one thing, information for the decision makers along the way to help them make the best decisions possible regarding resources and technical outcomes to make the development a success that will safely execute what it was designed to do.

One major change that has occurred impacting program offices as well as IV&V providers is the way software is developed. In the past decade, Agile software development has moved to become more of the norm when it comes to software development in the DoD and other federal agencies. As this approach to software development has been maturing within the federal space, many of the players involved with acquisition have had to tailor some of their own practices to allow the rules and regulations of federal acquisition to mesh with the iterative approach of Agile development. This is no different for IV&V; the work this team must perform has become slightly more complicated due to an Agile approach to software development.

It is up to the program office, teams that support the program office, and IV&V to understand how the developer is utilizing Agile in their development methodology and figure out the best way to integrate into that flow without more disruption than is required. There are many pros and cons when it comes to Agile software development in the public space, but there are a few positives that should be focused on from an acquisition standpoint.

If you reference the Manifesto for Agile software development (agilemanifesto.org), it highlights the values and principles that most Agile software development teams are trying to adhere to. There are a few themes of these values and principles that, from an acquisition perspective, should make the program office very happy. Those are enhanced collaboration, transparency and visibility in the work, working software delivered to the customer quicker, and flexibility. The Agile Manifesto is very focused on customer satisfaction, and that is great news from a program office's perspective.

One of the greatest strengths of an Agile development approach is the iterative nature, that provides numerous touch points with the developer, and opportunities to course correct along the way. This is very different to traditional waterfall development methodologies, where requirements and designs can be set in stone years before the project is complete. While it is true that the stone that those requirements and designs are etched in can be changed for a price, it is still an undue burden to the program office that they would like to avoid.

Challenge for IV&V

The challenges arise with the cadence at which the software is being developed and the IV&V team's ability to match that pace given the rigor that they are expected to provide when analyzing the software. A traditional IV&V effort would be able to review requirements, design, code, and test, and have naturally occurring checkpoints with the program office and developer to deliver findings.

Teams performing IV&V for a developer using an Agile approach likely have had to make some sort of adjustment compared to a more traditional waterfall development methodology. A waterfall methodology is a methodology described by Winton Royce in 1970 in which steps occur sequentially, one after the next, and when this is depicted visually, looks like a waterfall. In a waterfall methodology, the requirements would be written and then approved in some sort of review. After the requirements are approved, the developer would move to do the design of the system, which would also be approved in a formal review. After the design is complete and approved, code will be written to satisfy the design and ultimately tested as well. This methodology was the paradigm for several decades and is still prevalent in the federal government today.

From an IV&V perspective, it was relatively straightforward for the IV&V team to begin involvement on a project using a waterfall approach. Depending on what level of coverage the program wanted, IV&V could be added early in the requirements phase or brought in when the software was delivered, and the program was doing testing. No matter when IV&V began their work, the goal of adding assurance for the software was achieved. If IV&V was a requirement for that project, that requirement was satisfied. Yes, it makes sense to have IV&V involved as early as possible so that the team can build up their system understanding, and integrate their findings into the development, but sometimes that level of coverage isn’t in the budget and IV&V can only be brought on for code reviews.

The iterative approach of an Agile development is very different for an IV&V team that isn’t used to the speed at which the development can move through the traditional phases of development. Continually iterating on the requirements, the design, and the code. IV&V teams are used to having everything they need to perform the analysis they want to do and a date to deliver the findings by that typically coincides with a milestone review. This model for IV&V struggles to be successful unless the team is educated in Agile principles and understands how to best integrate with the program and developer while still maintaining independence, as will be documented later in the paper.

A program office who is going to perform IV&V utilizing an Agile development approach should look at it from the lens of getting to have more insight into the development as it is happening. This should come with more collaboration along the way, and opportunity, as the customer can share thoughts on where the development is headed. This collaboration may not always be easy given traditional relationships between program offices and their developers, but IV&V teams are in a unique position to help bridge any potential gaps that may be in place.

It does matter what level of independence the IV&V team has when it comes to execution and not impacting the work that the team is doing, but perhaps not so much from an insight perspective where the team is delivering insights to the program officer quicker. The Institute of Electrical and Electronics Engineers (IEEE; 2017) Standard for System, Software, and Hardware Verification and Validation (IEEE 1012-2016) does a great job highlighting the different levels of independence. These levels are so important to recognize and to understand by all parties involved when defining the scope and execution of the work. It matters if a program office wants to have a say in what is in scope for the IV&V team and what is not in scope. It matters where the funding comes from for the IV&V activities. It matters if the IV&V team shares developers with the team that did the development work. All these things impact that level of independence the team does or doesn’t have. This paper wants to focus on the integration of the information the IV&V team is providing to not only the developer, but also the program office. Using an IV&V provider that integrates into the agile development cadence is great for the developer, being able to add issues and risks into their backlog and weigh them with the business value of the other things they wanted to accomplish that increment. Allowing the developer to factor in the issues found prior to doing a thorough review when the software is delivered is reducing the amount of technical debt overall that the program will face. That same constant insight from an independent source to the program office is also extremely valuable. The early warning signs can help direct questions to the developer. The analysis done by the IV&V team can serve as the evidence needed to help decision makers with what they want to address and what they are comfortable accepting as risk.

This approach to IV&V could be different than what your program office is used to. The more traditional IV&V approach would have the IV&V team present findings at various milestone reviews, and decisions from the program could be handed to the developer to either fix the issues or accept the risk moving forward. The issue is that these milestone reviews could be years apart, an eternity in the scope of an Agile developed program. Adopting an Agile mindset and implementing Agile principles can build ceremonies and a cadence with a much greater frequency, shortening that timeline to months or weeks depending on your program's preferences. It also can help get inside the developer's OODA loop (Observe, Orient, Decide, and Act). Timing is very crucial for a program utilizing an Agile approach, so if you as the program office can help them out with your timing, it would be very beneficial for all.

Adopting Agile Values and Principles

The first thing the IV&V team needs to do to become Agile is to understand their process. This may seem straightforward, but many teams do not truly understand their IV&V process. The team sets out to simply just do IV&V. IV&V analysts are creatures of habit, and they fall into methods and techniques that are comfortable to them, where they have high confidence in their work. There is nothing wrong with that, but how does an IV&V analyst doing IV&V fit into the team's overall process. When does the analysis begin, when does it end, what is the analyst looking for, what will they do if they find what they are looking for, and so on. Thinking through this process is very important to understand what the team is doing when and for what reason. After the process is understood, it can more easily be determined where Agile principles make the most sense.

Once the process is understood better, the team can begin exploring better ways to get in phase with the developer. This is the overall goal of going Agile in the first place, for the team to be flexible and reactive in nature to things they may find along the way, enhancing their ability to communicate it to the developer and stakeholders as quickly and efficiently as possible.

The next step recommended to IV&V teams adopting Agile principles is to understand what work the IV&V team needs to perform to assure the software to the desired level of detail. This step tries to answer the questions posed earlier about when analysis begins and ends and what the analyst is looking for. This will likely involve some level of internal planning for the IV&V team to understand the entire scope of the task as well as to begin to develop a backlog of work that needs to be accomplished. This backlog, just like a developer backlog, will need to be groomed and prioritized based on the rationale set forth by the IV&V team. With an Agile project, the days of being able to get a code release and have 6 months to review it prior to a major milestone review while the developer and program office are doing their own testing are gone. As described above, the iterative nature of an Agile approach makes this approach a flawed one. So going into this level of planning and strategically deciding on what makes the most sense to work on when is critical.

Planning is foundational to a team utilizing Agile. It is an action that helps them stay grounded in what they are doing to achieve the vision and the goals of the development lifecycle. One of the key ceremonies that is observed by development teams, especially those at scale, is an increment planning meeting. These planning meetings are great opportunities to gain insight into the work the developer is planning to do. From an IV&V perspective, there should be a concentrated effort to attend these planning meetings to understand where the developer is prioritizing their efforts for the next 10 to 12 weeks. From a program office perspective, if there isn't already representation present at these planning meetings, it should be strongly considered for the strong insights they can provide.

Once the IV&V team has insight into what the developer plans to do at any given increment, they can use this information to prioritize the IV&V backlog, choosing the highest priority and relevant work, ultimately having greater impact and adding the most value as it pertains to the timing aspect of issue delivery. The program office will play a big role in this, but the IV&V team should try to get all issues identified and delivered for the given work they have chosen to do during an increment into the planning cadence of the developer. Whatever issue adjudication process is implemented by the program office should establish target due dates for IV&V issues. These due dates should provide enough time ahead of the next increment planning so that if accepted by the program office, the issues can be added to the developer's backlog and be prioritized with all the other work they would like to accomplish that increment. Doing this allows you as a program to get inside the developer's OODA loop. This will go a long way in strengthening the relationship between all parties involved. It helps the developer understand what the program office values and can then help the developer establish their business value off that. It will also drive conversations that will allow each side to understand the plan for that increment as well as the issues that the program office wants to see addressed at some point.

Observations

This Agile IV&V approach has been implemented across the federal government and has been used by NASA (Smith et al., 2019), and currently within the DoD. There are several observations through various implementations that should be pointed out and can help with the success of an Agile IV&V approach for any program. These observations also serve as warning signs for potential trouble spots normally revealed while performing IV&V for an Agile project.

The first set of observations revolves around relationships across the project. The levels of independence discussed earlier as defined by IEEE 1012 is something that needs to be clearly understood by all parties involved. It is very easy for program offices or leaders within the acquisition community to try and dictate where the IV&V team's effort should be focused based on the information shared. It is the good and the bad that comes with the IV&V team performing in phase analysis. If a more traditional IV&V was done after a milestone or at final software delivery, the conversation is more focused on what issues should be fixed. When the IV&V team is providing in phase analysis and communicating issues up through the program office to allow for adjudication, it is a very different approach and very easy to fall into the temptation of trying to drive the IV&V team to look in certain areas. If the team has the three levels of independence, this violates that independence in the managerial area. Now, if it were discussed ahead of time before the IV&V began their work that the program office would have the ability to steer or recommend where the IV&V team would focus their efforts based on findings, that that is a different story, but everyone is on the same page and understands that from the beginning.

Another observation is the "us vs. them" mentality that can develop in any IV&V relationship. It is very easy to forget that everyone is on the same team and ultimately wants the same outcome of mission success. Given that this approach calls for increased communication and integration of findings into the developer's backlog, there seem to be more opportunities for this mentality to arise. The program office will receive issues and risks that need to be communicated to the developer on a much more frequent basis, and given the early lifecycle integration, it can be easy to develop a pessimistic attitude towards the developer. This is where the insight and understanding of the developer's approach and plan can help tremendously in trusting that plan. There will be times where the developer chooses not to incorporate the IV&V findings into a specific increment's plan, and that is okay. The issue still needs to be captured in the backlog or an issue tracker for future reference, but if other things are higher priority that planning increment, that is an acceptable approach. The trap that teams can fall into in this dynamic is the developer is doing that intentionally, they are ignoring IV&V, or the program office. That is likely not the case; it is more of a matter of vision and knowing when it makes the most sense to implement fixes in their development process.

Ultimately having an IV&V team getting involved in software assurance is an exercise of trust. The relationship between IV&V, the program office, and the developer should be collaborative in nature. The IV&V team will know their lane and understand the boundaries that should not be crossed to maintain independence. It is up to everyone involved to make sure that the IV&V involvement does not lead to an adversarial relationship. It is very easy in development environments for IV&V to take the “blame” for slowing things down. IV&V will find issues that will create technical debt; they will identify systemic risks that no one else could possibly have seen. At the end of the day, it is highly likely that IV&V will create some very difficult decisions on what findings must be addressed and what risk will ultimately be accepted, which can lead to a delicate balance of the perception of IV&V. There is a great paper by James Dabney and James Arthur (1998) highlighting the professional challenges to IV&V which reiterates some of the observations mentioned related to the parties trusting and cooperating with each other as well as some other potential pit-falls that could occur when using IV&V.

Another delicate balance is the importance of sticking to the facts that are produced by the evidence that the IV&V provides. This is important for all parties involved, going back to the earlier mention of the shared vision of providing the best software product possible. Based on the evidence the IV&V team provides, if one becomes too emotional regarding the findings, it could lead down a trap that the developer wasn’t performing up to their maximum potential. This is just emotion leading people down an unproductive path. The facts help ground everyone in the reality of what has been found. If the issues that are found are in fact warranted, then the teams will need to work together to figure out how to move forward. The decision with what to do with the issues will likely come down to risk tolerance, but finding these issues earlier in the lifecycle is such a better outcome than finding them later or, worse yet, in operation.

Conclusion

In conclusion, there are many approaches IV&V can take for projects implementing Agile software development. There are many approaches for the program office to have insight into how the development of the software is progressing. This paper highlights the value in transparency and communication. Between the program office and the IV&V provider, as well as communication with the developer. What has been described is an approach that is tailorable to fit various settings in acquisition, and applications of IV&V. In the spirit of Agile, this approach to IV&V focuses on continuous improvement. How can the team, and team in these cases is everyone involved with the project, continually learn from each other to improve the overall development of whatever they are trying to do.

The key concepts that should be considered are consistent communication with the IV&V team and being an active attendee, and participant if able, during the developer’s increment planning, getting inside their OODA loop. These concepts paired with insight into the backlog of IV&V work through the consistent communication will help the program office be much more in phase with the developer. Depending on the level of independence, the program office may or may not have influence of the priorities of that backlog, but either way, they will have a clearer picture and the supporting detail of the work being performed by the IV&V team.

Ultimately, program offices acquiring a software product from a developer will have to make a final call on acceptance of that software. If IV&V is involved in some fashion, issues with that software will likely be uncovered along the way. It will be up to the program office leadership to decide what to do with that information, no one else. The IV&V team will have opinions on the findings, the developer will have opinions on the findings, and there will likely be cost and schedule impacts to the decisions that the leadership team must make regarding the findings. At the end of the day, though, hopefully the use of an IV&V team in an agile approach has informed the decision makers enough along the way that they feel confident enough leveraging the beneficial perspective of IV&V to influence the future in a proactive and timely manner rather than correct the past.

Hopefully, this paper has shown the value when it comes to performing IV&V in phase for an Agile project. These concepts do not have to be limited to IV&V, though; the same principles and approaches can be valuable to the internal verification and validation (V&V) team or even operational testing. All these teams contribute to the overall risk-mitigation of the software development for the program and can benefit from some of the concepts discussed in this paper. There is so much value in the increased communication across the program and the developer that it is worth looking at ways to implement different Agile ceremonies. If a program is struggling with technical debt, and consistently being surprised by issues coming up and impacting cost and schedule, Agile and lean concepts should strongly be considered by the program office to help get in sync with the developer.

References

Agile Manifesto. (2001). agilemanifesto.org

Dabney, J. B., & Arthur, J. D. (1998). *Anticipating and mitigating the professional challenge to independent verification & validation*. Department of Computer Science, Virginia Polytechnic Institute & State University.

Institute of Electrical and Electronics Engineers. (2017). *IEEE standard for system, software, and hardware verification and validation*. IEEE Standard 1012-2016.

Royce, W. (1970). *Managing the development of large software systems*. Proceedings of IEEE WESCON.

Smith, J., Bradbury, J., Hayes, W., & Deadrick, W. (2019). *Agile approach to assuring the safety-critical embedded software for NASA’s Orion spacecraft*. 2019 IEEE Aerospace Conference.