Implementation of Agile Methods within Instructional Systems Design: A Case Study

Lisa Cooney
Addx Corporation
Alexandria, VA
lcooney@addxcorp.com

Anne Little, PhD
Addx Corporation
Alexandria, VA
alittle@addxcorp.com

ABSTRACT

Today's economic environment of shrinking budgets demands training that aligns with the needs of the workforce by improving specific employee behaviors. When organizations identify a workforce deficiency there often are critical implications for their operations, so they typically respond with training solutions to correct the deficiency.

During the fall of 2014, the authors were tasked with a course design and development effort for a two-week instructor-led program management course. The traditional instructional design methods used for previous versions of this course relied on a locked-down front-end design, classic linear processes, and evaluation methods that provided feedback late in the development cycle. The previous version of the course was heavily based on another Federal agency's models and did not meet the needs of the students. Additionally, policies needed for inclusion in the class were in flux. The time available for development was tight; two full-time instructional designers and three part-time subject matter experts needed to create a 10-day instructor-led course in time to deliver a class offering in less than seven months.

To create the new course, the development team incorporated the principles of Agile software development. This paper will review Agile software development, the ADDIE (Analysis, Design, Development, Implementation, Evaluation) instructional design method, and explain how the team applied the principles of Agile development to instructional systems design. We will discuss the implementation process, organizational tools, team dynamics, and customer involvement. Finally, we will illustrate the potential cost savings of this method by comparing a summary of the resources utilized to industry training development metrics.

ABOUT THE AUTHORS

Ms. Lisa Cooney is an Instructional Systems Designer for Addx Corporation. Ms. Cooney is certified by the Scrum Alliance as a ScrumMaster, and by the Project Management Institute as an Agile Certified Practitioner (PMI-ACP). She designed and wrote two virtual instructor-led courses called Agile for the Federal Government and Agile for the Product Owner for the Department of Veterans Affairs in 2014. She has expertise in the area of Section 508 compliance, and she has experience working for several federal clients including the Department of Homeland Security Acquisition Institute, the Veterans Affairs Acquisition Institute, the Federal Deposit Insurance Corporation, the Federal Acquisition Institute, and the Central Intelligence Agency's Open Source Center. Ms. Cooney earned her Master's degree in Instructional Design from the University of Virginia and holds a Bachelor of Arts from Wellesley College.

Anne Little, PhD is the Director of Training Services for Addx Corporation. Dr. Little provides oversight and management of all aspects of Addx's career development training programs. She has extensive experience designing training programs for many federal clients including the Department of Homeland Security Acquisition Institute, the Department of Energy/Energy Information Association, Defense Acquisition University, and the Federal Aviation Administration. She is recognized for developing innovative training solutions, especially in the area of leadership development and performance support. Dr. Little earned her PhD in Instructional Technology from George Mason University and holds a Bachelor of Science in Mathematics from Purdue University.

Implementation of Agile Methods within Instructional Systems Design: A Case Study

Lisa Cooney
Addx Corporation
Alexandria, VA
lcooney@addxcorp.com

Anne Little, PhD
Addx Corporation
Alexandria, VA
alittle@addxcorp.com

INTRODUCTION

In the fall of 2014, our team was worried. We worked onsite at a government agency on a long-term contract, and we had just been tasked to create a two-week live instructor-led Program Management capstone course for senior program managers. Program and project managers within the agency typically progress through a series of prerequisite courses as part of achieving certification. However, in order to be admitted to this capstone course, each participant's experience level was screened to ensure the training audience met minimum thresholds for work experience and complexity of program to which they are assigned. Therefore, the training audience was typically comprised of GS-14-15 and O-4-6 members. This course culminates their training in project and program management, and is required for Level III certification.

The previous version of the capstone course was lacking in several areas; fundamentally, it did not teach the critical thinking skills essential to program management, and it did not include real-life examples. Our client had several criteria for the new course. It had to:

- Be innovative and creative
- Be in-depth
- Align with the new management core competencies of the government agency
- Align with new and emerging agency guidance on acquisition practices
- Be sufficiently demanding to justify a Level III program management certification upon successful completion
- Contain all new content
- Be highly interactive with many complex exercises based on actual live programs
- Teach critical thinking and analytical skills

Using industry standard development metrics provided by the Association for Talent Development (formerly ASTD) (2009), the authors estimated traditional instructional design would require approximately 60 hours of development time for each hour of training, or approximately 4800 hours to complete the two-week course. Our staff consisted of two instructional systems designers (ISDs) and three part-time subject matter experts (SMEs). In total, the available staff resource level was approximately 3.5 FTEs (full-time equivalents). Assuming each worked 1900 hours per year, it would take at least eight months to complete this project. External stakeholders' approval requirements increased the development time to an estimated 80 hours, which would have required close to 12 months. Using more aggressive estimating metrics didn't help; we had only seven months before the first scheduled delivery; thus our worry.

Faced with a seemingly insurmountable task, the program manager decided to try a new instructional design approach based on computer software development, called Agile instructional systems design[1], or Agile ISD, and she hired a senior instructional designer experienced in this method. We successfully delivered the course less than six months later. To understand what happened on our project, it is important to understand three things: traditional software development, Agile software development, and traditional instructional design.

[1] Agile ISD is based on the Agile software development. Do not confuse Agile instructional systems design, or Agile ISD, with A.G.I.L.E. instructional systems design. (http://www.learningsolutionsmag.com/articles/1346/effective-performance-with-agile-instructional-design) The latter is an instructional design model partly based on Agile software development, but it is quite distinct from Agile ISD as described in this paper.

TRADITIONAL SOFTWARE DEVELOPMENT (WATERFALL)

Agile software development (often referred to as Agile) uses the term "waterfall" to describe traditional approaches to software development. Agile defines itself against this traditional model. On a waterfall project, major phases are completed in a strict sequence, each one feeding into the next, in a process that resembles a waterfall:

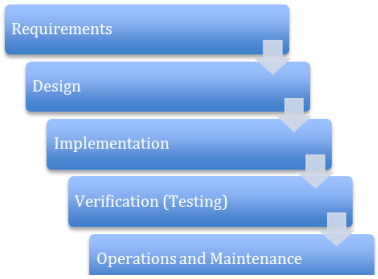

Figure 1. Traditional Waterfall Development Process

While Agile software development continues to increase in popularity, waterfall methods are still common, especially in the Federal Government. This is likely due to contracting issues; Agile handles and even welcomes changing requirements late in the process which is a stark contrast to government contracting processes. Nonetheless, the White House Chief Information Officer (CIO) began promoting Agile in 2010 (Kundra, 2010), and Federal agencies are responding. A GAO report from 2012 (GAO-12-681) details the issues and success factors relevant to Agile in the Federal Government, and is among a bevy of papers and studies promoting Agile software development (American Council for Technology - Industry Advisory Council, 2013, 2014; Kundra, 2010; Northern, 2010; Gorans, 2014).

AGILE SOFTWARE DEVELOPMENT

Agile began after the dot-com bust in 2001 when a group of prominent software engineers got together to discuss how to fix a problem plaguing the industry: 80% of their work was unneeded and discarded, and projects routinely were canceled or software rollouts failed. In response, they developed a radical new way to manage software development projects to produce high-value, high-quality software quickly and sustainably. Being able to sustain a steady pace indefinitely was important to them, because previously software coders worked around the clock and burnout was high, yet they still could not deliver working software on time. The founders of Agile wrote up their ideas into a document called the Agile Manifesto (Beck, et al., 2001).

When using Agile, people work closely in small teams, collaborating daily to prevent miscommunications. Teams communicate frequently with the client and other stakeholders, including users, and make adjustments to the product as they go along based on feedback. Agile is intuitive; when pressed for time and resources, teams often begin to work in an agile manner simply to finish the project, by using methods such as communicating much more frequently, breaking up the work into chunks with iterim deadlines, and checking in often with their client to be sure they are on track.

Agile is flexible, and lets teams incorporate the best methods for their situation. "Scrum"[2] is one of the most popular Agile methods, and it includes a set of meetings that enable the work to flow smoothly. Work is divided up into two to four week segments or time-boxes, called "sprints," with a clear set of deliverables due by the end of the sprint. Tasks are prioritized into a task list called the "product backlog," and the level of effort required to complete each is estimated and recorded on each task. The team reviews the completion criteria to ensure everyone understands the work, and reviews dependencies between tasks. At the end of each sprint, the team shows the work to the customer, also known as the product owner.

[2] The term "scrum" was borrowed from the game of rugby to describe how the players gather in a group, called a scrum.

On an Agile software development project, all the activities necessary for software development still occur, just in an iterative fashion. A product is "done" when the customer decides that the product offers sufficient high-value features, and no additional features are needed.

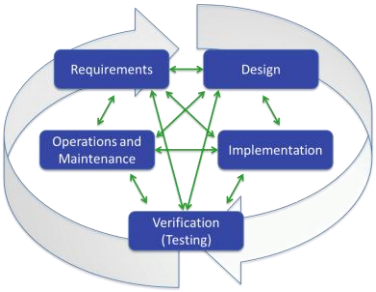

Figure 2. Iterative software development

In addition to scrum, many other management techniques such as Lean and Kanban have been absorbed under the Agile umbrella, as well as software development techniques such as eXtreme Programming, continuous integration, and paired programming. While many spin-off approaches have sprung up, Agile itself remains flexible because at its core, it is a set of values and principles, and does not dictate any one set of practices. Software companies using Agile enjoy tremendous success -- think of Google, Facebook, Apple, Spotify, and Uber to name a few. Many others have tried their approach with varying success. Agile is not necessarily easy, because it requires people to change not only their way of working but also their fundamental mindset about it. Doing Agile well is more of an art than a science. Organizations that adopt the methods without understanding or applying the values and principles typically run into problems. This has led to whole industry of "Agile coaches" who assist organizations on their journey towards becoming Agile.

ADDIE

Analysis, Design, Development, Implementation, and Evaluation (ADDIE) has been the most common instructional design model for several decades, and has worked well for many projects. It is a waterfall approach to training development, and when time and resources permit, can be the best method (Branch, 2009).

Figure 3. ADDIE process

As Figure 3 illustrates, the concepts behind ADDIE are similar to traditional waterfall project management. After conducting a needs analysis and designing the training solution, changes to the original design are minimal and often discouraged. All parties agree on the design, and then development work begins. Once the piece of training material is complete, it is implemented, often with a pilot (trial run) first, and the results evaluated to determine the effectiveness of the training solution. The activities involved with ADDIE are important and need to happen for any training effort. Good design is essential, and it is important to evaluate the effectiveness of the learning solution. Needs analyses provide the data fundamental to getting the content right, and to adjusting the course material to the type of learner.

While each phase of ADDIE is important, doing them strictly in order can be problematic for several reasons. First, ADDIE assumes that the best ideas come at the beginning of a project. In practice, on a complex project involving many stakeholders, training requirements emerge throughout the development period. Inevitably, new material must be added or existing material changed or deleted. This has consequences for course material that has already been developed, often resulting in extensive rework. Traditionally, the upshot is renegotiating the contract due to "change

orders" or deferral to a future project which may consist entirely of updating the course. The inflexibility of the design can be an impediment not only to completing the work on time, but also to providing a course that best meets the training need. Essentially, it is too costly to admit you were wrong. A second problem with ADDIE is that evaluation takes places late in the project, when it is harder and more expensive to make changes to the course. The sooner the ISDs know what problems exist in the material and design, the sooner they can address and correct them and this new information will guide future development. Agile ISD involves showing rough drafts to clients and stakeholders (including sample learners) early in the process, in order to make sure that the developers are on the right track and to obtain direction not only for reworking this small piece of training material, but also to learn about the learner needs going forward in order to implement their requirements effectively into the training. In effect, the needs analysis is ongoing during an Agile ISD project. Agile ISD, with its focus on early delivery and regular feedback, results in a much higher quality product from the perspective of accuracy and relevance of training content.

AGILE ISD IS ITERATIVE AND INCREMENTAL

Incremental projects deliver work products in phases, with each phase having all the parts and pieces necessary to be complete, for that phase. Iterative projects deliver work in drafts, with each draft becoming progressively more detailed until all the work is complete. Figure 4 illustrates these two processes.

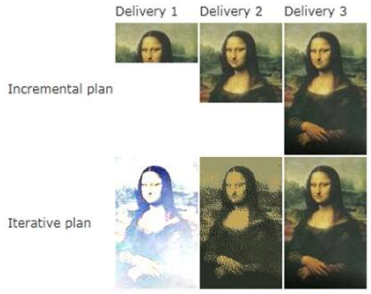

Figure 4. Incremental and Iterative development approaches[3]

Agile is both iterative and incremental, combining elements of both approaches, as shown in Figure 5.

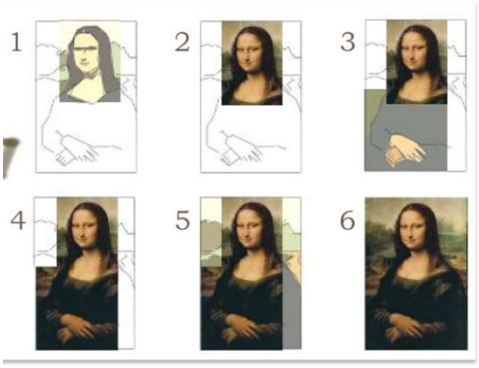

Figure 5. Combined Incremental and Iterative Development Approach[4]

Like Agile software development, and in contrast to ADDIE, Agile ISD is iterative and incremental in nature. All the same types of work occur, but in a different sequence. High-level up-front planning must take place of course, because someone must decide that a problem exists and that a training solution will fix it. However, Agile ISD does not expect a locked-down design at the beginning of the project. Instead, Agile ISDs begin to gather some preliminary information, create a preliminary design, and give this to the customer immediately (in draft form) for

[3] http://itsadeliverything.com/iterative-incremental-process-illustrated-with-mona-lisa/iterative-incremental-mona-lisa
[4] Ibid.

feedback. If the customer likes the idea, then development work begins right away, before the entire course has been mapped out. The results of this development work – say a draft lesson, or a prototype of an e-learning module – again go before the customer right away, within a week or two, for feedback. During this period, the constant – albeit sometimes very brief – communication between the client and the developers helps everyone to clarify the needs, identify the issues, and further refine the design for the training solution.

THE FUTURE OF AGILE ISD

Agile practices, which result in processes, are very important, but the values and principles are even more important because they meet the needs of today's knowledge workers. Knowledge workers, who earn a living using their minds rather than from manual labor, desire purpose and meaning in their work (Drucker, 2001). They value a healthy, collaborative, and especially productive work environment over increased compensation, and organizations that can meet these needs will be able to attract the top talent and foster high-performing teams (Denning, 2010).

The authors believe eventually the basic work practices of instructional design will be fundamentally altered by the widespread adoption of Agile ISD, both in the private and the public sector. Government customers, along with their private sector counterparts, will seek out Agile instructional design teams who can provide high-value training material quickly, and who have the agility to change content even very late in the process. This approach to training development will be transformative not only for the field of instructional design, but also for the customers they serve who will be delighted with the results Agile ISD teams can produce.

CASE STUDY

Overview

Our team was tasked to develop a two-week instructor-lead acquisition Program Management course. We began with a high-level outline for the entire course, based on terminal and enabling learning objectives derived from FAI (Federal Acquisition Institute) program management competencies. The objectives detailed the purpose of the course and each lesson. In order to ensure that we were on the right track, we began drafting lessons before we had finalized the detailed outlines for each lesson. This way, our customer could see a notional intended product and provide early feedback. The customer reviewed the material (ignoring typos and other errors in form) in order to confirm or adjust the direction the course needed to take. Sometimes this feedback necessitated changes to the overall design – perhaps the rearrangement of the sequence of lessons, or moving content from one lesson to another, or replacing entire sections of intended content.

We decided to use Agile ISD, which incorporates Scrum. During each sprint, the iterative process of redesign occurred simultaneously with the development of new lessons, and in some cases we added, deleted, or rewrote objectives as it became clear to the team and our client that it was necessary to do so. Not being locked down to a tight design allowed us to have great flexibility, enabling us to put in just the right amount of content in just the right spot in the course.

We had several stakeholders with a vested interest in the course from across our Federal client's organization including the certification requirements and policy management offices, so we sent drafts to these stakeholders as well. Our process involved completing a lesson within the team, obtaining government PM feedback and implementing it, then sending the material out to our wider stakeholder base for review. In some cases, where the course wrestled with particularly thorny content, we called a meeting of contractor and government subject matter experts to resolve guidance conflicts on what to put in the course.

Time-boxing

Each two-week sprint began with a collaborative planning session, during which we reviewed all the work in the backlog. We prioritized the work, assigned points (or level of effort) to each task, and decided which tasks to address in the next sprint. After the first few sprints, we could measure our team velocity (the number of points completed each sprint) and determine our team capacity (the number of tasks we think we can do in an upcoming sprint). The product backlog was "refined" towards the end of each sprint; we prioritized the work according to level of risk, putting high-risk items first, and considering customer needs and priorities as well as dependencies between tasks.

At the end of each sprint, we held a sprint review meeting, during which we presented all completed work to our client for feedback. Usually, we gave the client drafts of lessons during the sprint for them to read ahead. We implemented all the feedback the same day, but if it would take longer, we would document a new task to track the completion of this additional work. These review meetings ran no longer than 90 minutes and occasionally, were completed in less than 30.

Kanban

Kanban means "billboard" in Japanese, and it was developed in Japan by Toyota as a way to track and manage tasks on a complex project (Kniberg, 2010). We wrote down all of our tasks on separate sticky notes, referred to as cards, and placed them on the wall organized into columns, as illustrated in Figure 6.

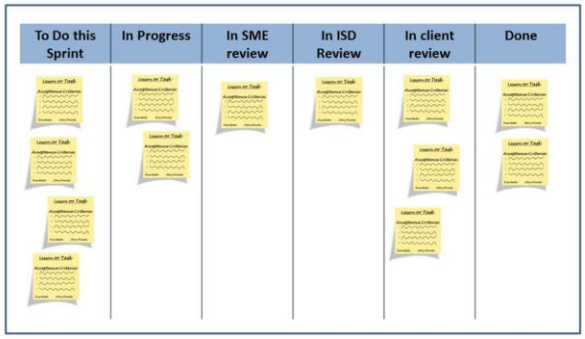

Figure 6. Agile ISD Kanban Board

The columns represent task status. Each card refers to a lesson or an objective, or to an administrative task. Team members reviewed the board daily, and pulled the work they would work on. As task status changed, the people working on them moved the cards to the next column. Usually, the team had a brief conversation following our daily scrum about who was working on what to ensure that dependencies were taken into account.

Each card included a short name (usually a lesson which mapped to an objective) and detailed completion criteria. These criteria had to be met before the team would move the card to the "Done" column. Each card also included a due date and a point assignment.

Sometimes an internal SME or ISD review during a sprint revealed that more work was needed. When this happened, we put the card back into the sprint backlog. In this way, cards moved left and right across the Kanban board. Other times during a Sprint Review meeting, significant work was still needed to complete the task. When this happened, we move the card back to the product backlog. Each card's final completion criteria included the phrase "Client approval obtained" which ensured that the only cards to make it to the "Done" column had been accepted by our client

Limiting Work in Progress (WIP)

Multi-tasking is the enemy of efficiency (George, 2014) and this applies to teams as well as individuals. Keeping WIP in control is a central tenet of Agile. Research has shown that tasks get accomplished more quickly and with higher quality when teams limit the amount of work in progress (Bradberry, 2014). Basically, by focusing on just one or a few tasks at a time, the team can work collaboratively to complete them. If they try to work on too many tasks at once, the quality of work degrades on each task, and overall the tasks take longer to complete. Agile tries to limit the amount of work being done at any one point in time to achieve maximum efficiency. Team members pull items from the Kanban board one at a time. If they need to stop work on a task to move to another task, then they put the card back in the "To Do" column and pull the new card. This serves several purposes, the most important one being transparency. At any point, the client or any team member can look at the board and immediately see what each team member is working on.

We limited work in progress by not allowing too many cards to build up in the "In Progress" column, and by carefully regulating work we pulled into any one sprint. We also kept a close eye on bottlenecks, which are easy to spot because a lot of cards pile up on one column.

Definition of Done

An important concept in Agile is the "definition of done." Saying that a lesson is "done" does not mean it will never change. The definition of done is relative to one particular task that is included in one particular increment of work, and is documented in the completion criteria for each task. If later on the product owner (the client) wants to modify the lesson to add new content, or an ISD wants to modify the lesson to fix instructional flaws, they can create a new task which gets added to the list, or backlog. For training material, a "definition of done" should include most of the following: Completion criteria are met; material has been peer reviewed; material has been tech edited; material has been integrated into the overall course; material has been accepted by the product owner.

Agile ISD follows Agile Values and Principles

The Agile values and principles apply as well to courseware development as they do to software development. On our project, the Agile value of responding to change over following a plan (Beck, 2001) was critical because we did not have access to the real-life program documents necessary to complete our exercises, which made up at least 50% of the course. Out of necessity, we had to use scenarios from other courses, or simply make up scenarios. Once we finally got access to real program documents, we had to go back and adjust the exercises, and this often entailed changing the course content as well. Agile processes harness change for the customer's advantage, and we welcomed changing requirements late in the process, in part because we expected them. Our stakeholder base was broad, with unexpected input sometimes coming in late, but we simply adjusted our material and kept on moving forward.

The Agile value of working software (courseware) over comprehensive documentation (Beck, 2001) in the context of our project meant that we put all of our focus on creating useful, content-rich course material and spent less time documenting our work in spreadsheets or reports. There was ample documentation on our Kanban board, which was constantly visible to the whole team, including our client, and which lessened the need for documenting our work. The Agile principle of face-to-face communication meant that we avoided misunderstandings about the nature of the work, and established trust early on, which also reduced the need for elaborate records of our progress.

The Agile value of customer collaboration over contract negotiation means that it is more important to have a conversation with the customer and ensure that we both understand the nature of the work, the progress we have made and our plans for the future than to resort to renegotiating a contract in order to redefine the work. Contract negotiation is still important, it is just less important than customer collaboration. Trust and collaboration between the vendor and the customer result in the need for fewer contract modifications.

On our team, this meant that we communicated several times a week with our client. Rather than engage in discussions about how the work would get done or who was responsible for what, we kept the focus on the business value – a great course – as opposed to focusing on team logistics. We checked our priorities weekly with our client, who informed us when we needed to shift them around based on new requirements. Because we worked in two-week sprints, we had an agreed-upon set amount of work in a time-box, so we put all requests for new work into the product backlog for future sprints. This enabled the development team to honor the commitment they made at the beginning of the sprint to get a set amount of work done in a set time period, and it enabled our client the freedom to add work to the project. During planning sessions, we would ask the client to prioritize the work, and some things became so low priority that they never got done.

Our goal was early and continuous delivery of valuable courseware. For us, this meant every two weeks we gave our client a lesson or set of topics that were 98% complete (lacking only a copyedit). Even if it was a small amount of material, what we provided was a complete solution for the piece of content. An instructor could pick up our slides, instructor guide, exercises and handouts, and teach the class, as developed up to that point. In this way, we showed the client directly the value of our work, and gave them ample opportunity to comment on it and make corrections. Through this regular communication, we learned what our client valued most highly, and we were able to apply that knowledge to future lessons, which sped up the process of lesson development by reducing the number and nature of client comments.

The most important Agile principle on our project was simplicity - maximizing the amount of work not done. The concept here is to eliminate unnecessary work, following the principles of Lean (Kniberg, 2010). The first Lean principle is eliminating waste, which means not doing any work that can be avoided while still achieving a high-quality product quickly. The guiding concept is minimum viable process. Do what is necessary, and nothing more.

With our extremely tight timeframe, we had little leeway for rework, and no time at all for unnecessary work. Early and frequent customer and stakeholder feedback went a long way towards helping us be lean. Having all of our work visible on the Kanban board daily, with any member of the team able to see at a glance what the others were doing, enabled us to maximize our efficiency.

Frequent Feedback

A fundamental concept of Agile is early and frequent customer feedback. For Agile ISD, this means the client must review draft material at the earliest possible moment, and must provide detailed feedback immediately. For our team, this meant giving our client small bits of content at a time that could be assessed quickly and given the "thumbs up" to continue. Or, if the product was not meeting expectations, the development team was in a position to converse with the client to understand the adjustments needed. These conversations are the heartbeat of Agile ISD; without a fearless ISD and an engaged and honest client, the process will not work. The ISD must relinquish material as soon as possible, even with imperfections. In turn, the client must review the material right away, and provide detailed feedback quickly. At appropriate points, other people must be brought into the process as well: SMEs; sample learners; senior management and so on. Conversations between and among all these stakeholders provide the critical feedback necessary to improve the course. Everyone must have a shared understanding that early material is young, and will undergo multiple iterations before all accept it as final

Quality Control

Quality control was managed in several ways. The multiple layers of internal and client review ensured that many eyes reviewed each portion of training material. The "Client Review" column sometimes included external groups; this was clearly spelled out in the completion criteria. By working iteratively, we were able to improve the course materials on an ongoing basis, including going back to earlier material to make adjustments when later changes to other material required this. Finally, before going to pilot, all material went through a copy edit process.

Meetings

Agile ISD requires frequent, brief face-to-face interactions, which lead to remarkable efficiencies because these interactions help the team avoid costly rework and wheel-spinning. Daily scrum meetings enabled the team to be clear on who was doing which task. Usually an issue or two would arise during our morning scrum, and we would hold a post-scrum meeting to resolve the issue before getting to work. Achieving clarity on the work to be done is very important; teams are much more efficient and productive when they know exactly what is needed.

Our team also held sprint retrospectives each sprint, during which we inspected ourselves honestly and came up with ways to improve and be more efficient. Each retrospective yielded specific action items to which the team was held accountable during the following sprint, or beyond. Retrospectives were a great way to establish team norms and a shared understanding.

Benefits and Challenges of Agile ISD

The benefits of Agile ISD for an instructional project are similar to those for a software project; at the end of the project, you end up with something that truly meets the business need. In the training realm, the business need is two-fold: providing the workforce with training they need to do their jobs and delivering it at a cost savings.

We discovered that there are definite barriers to implementation success and they are mostly cultural. One of our team members struggled with sharing early drafts of her work with the team. The notion of sharing a draft product (that might have imperfections) with the customer generated such a level of anxiety for her that it was best to remove her from the team. We were fortunate to have a collaborative relationship with our customer who understood the need for us to get early approvals in order to ensure we were on the right track. Our lessons learned from this project were that Agile ISD requires people who 1) are willing to work collaboratively, and 2) are able to adapt to changing the design late in the process. It became clear to us that Agile is not for everyone, especially those who are uncomfortable with uncertainty, change, and transparency. This is an important consideration for teams who may wish to consider adopting Agile ISD; the processes are actually quite intuitive and do not seek to change good design strategies or methods. The real concern is cultural. If your staff is used to working in a manner that is driven strictly by static linear processes or need to have their work tasks directed to them, it is likely the Agile practices will not be well received.

In our case, we were able to unravel the cultural problem with our reluctant staff member. In fact, even though we were technically "short staffed" once she left the project, we noticed a productivity increase because everyone else was finally operating in sync. From our customer's perspective, our team's transparency about our work made it easy for her to be fully aware of our progress (without the need for time-consuming briefings or written reports), and feel a level of comfort with the work we were doing. Prioritizing our work visually using cards on the wall prevented confusion about what tasks were being worked on, and what had highest priority. If our customer asked us to complete new work during a sprint, we would ask our client the question, "what work that we had planned for this sprint would you like us to set aside so that we can tackle this new assignment?" Unsurprisingly, new work often took a back seat to the urgent work already in the sprint backlog. During the next backlog refinement session, we would discuss the new work and our customer might decide to shift priorities; the development work of the team could continue uninterrupted during the sprint.

Although the cultural aspects were our biggest concerns for this project, teams do need to ensure that they have the proper talent base to maximize the opportunity for success. We had support from two external subject matter experts from time to time, an external technical editor, and support from a graphic artist. Our program manager, while not officially part of the development team, has a Ph.D. in instructional design and was able to support us with guidance and support weekly. Each person on the team brought strong complimentary skill sets to the project, but was capable of filling in for each other when someone was out or unavailable. Our ISDs had program management and Agile expertise which allowed them, on occasion, to serve as SMEs for part of the course material. Both ISDs were capable of working with graphics, which reduced expensive graphics development or costly rework. One ISD was a former editor so our materials were fairly "clean" by the time they went to the tech editor. Our SMEs were conversant in instructional design techniques and able to deliver raw content in very good shape. Being cross-functional is an essential characteristic of an Agile team, which relies on self-organization to get work done in the fastest most efficient way possible. Our team was small but cross-functional, which was an important element of our success.

DATA ANALYSIS

While it is not possible to say what would have happened had we not used Agile ISD for this project, we can make some educated guesses. Traditional instructional design practices would dictate having a design document written and approved before beginning work; however we started writing early lessons before we had completed the design for later lessons. Traditional ISD would have meant completing fewer iterations, raising the risk that the course we were designing and writing did not match our customer's needs, would not meet stakeholders expectations or (worse yet) not provide students with a valuable learning experience. The communication structure allowed us to have frequent meetings with stakeholders to confirm the design and development of each lesson in an iterative fashion, with their feedback being implemented not only in the particular lesson they had just reviewed, but also being incorporated backwards and forwards in the course. Lessons were developed out of order, which was not a problem because our constant communication on the team enabled us to adjust older material easily - we created new tasks for "fixes" to draft material.

For comparative purposes, we tracked the number of hours required to design, develop and publish all of the training assets. Utilizing ISDs, SMEs and a copy editor (for final quality assurance) the total number of hours expended was 2,788. This translates to 34.85 hours of development time per seat time of the course. Within the training community, most developers rely on the Association for Talent Development (formerly ASTD) metrics which range from 43 to 185 hours per hour of seat time (2009), depending on the complexity of the training content and the level of interactivity of the training delivery method.

Our students and instructors were very happy with the quality of the course. One experienced instructor said that the instructor guide and student materials (handouts and exercises) were the best he had seen. One student told us that this was "the best course I have taken in the government because it forced me to use critical thinking." In the second offering of the course, in response to the statement "I would recommend this course to others," 10 of the 17 students chose "strongly agree" and the other seven chose "agree." Nine of the students strongly agreed with the statement "this training will improve my job performance" and the rest agreed.

All team members, including the project manager, SMEs, ISDs, writers, and editors, felt liberated to be creative due to the self-organization of the team. Pulling work from a Kanban board is very different than being assigned work,

and gives team members a sense of commitment to the work as well as a sense of autonomy and self-direction that increases motivation and productivity. Agile ISD is sustainable; teams should be able to continue their pace indefinitely, thus avoiding burnout and turnover which are both so damaging to productivity and ultimately, undermine meeting the business need. Agile ISD maximizes creativity, minimizes re-work, and uses self-organizing teams to maximize individual responsibility for a successful project outcome.

CONCLUSION

Various instructional designers and training teams have adopted parts of the Agile philosophy into their processes. The Successive Approximation Method (SAM), a Lot Like Agile Management Approach (LLAMA) and Align, Get set, Iterate & Implement, Leverage, and Evaluate (A.G.I.L.E.), and Agile Learning Design are all instructional design approaches that have inserted favorite Agile practices. What is missing from all of these is an emphasis on the Agile values and principles fundamental to Agile ISD's success, and a meaningful commitment to incorporating scrum methods. Instructional design work is pure knowledge work, and people are at the heart of it, from the design and development side to the receiving end. To do their best work, they need to feel a sense of purpose and mission, they need transparency into the work in order to achieve collaboration, and they need to have time-boxed iterations that push everyone to work towards an achievable short-term goal that leads to a similarly achievable long-term goal. High-performing teams are sought-after, but creating them is more of an art than a science. Agile ISD allows us to move from an average team to a high-performing team, and provides a framework to enable the early delivery of high-value training content for any medium. The benefits are compelling, and those instructional design professionals who adopt Agile ISD will delight their clients by providing high-value training material quickly.

REFERENCES

Allen Interactions. *Agile e-learning development process.* Retrieved June 10, 2015, from http://www.alleninteractions.com/sam-process

American Council for Technology - Industry Advisory Council. (2013). *Planning for success: Agile software development in federal agencies.* Retrieved June 10, 2015, from http://www.federalnewsradio.com/pdfs/082113_agile_software_development.pdf

American Council for Technology - Industry Advisory Council. (2014). *Acquisition best practices to procure agile it services.* Retrieved June 10, 2015, from https://actiac.org/sites/default/files/Best%20Practices%20to%20Procure%20Agile%20IT%20Services%20-%20ET%20SIG%2003-2014.pdf

Beck, K., Grenning, J., Martin, R. C., Beedle, M., Highsmith, J., Mellor, S., et al. (2001). *The Agile manifesto*. Retrieved June 10, 2015, from http://agilemanifesto.org/

Begel, A., & Nagappan, N. (2007). *Usage and perceptions of agile software development in an industrial context: An exploratory study*. Retrieved June 10, 2015, from http://research.microsoft.com/pubs/56015/AgileDevatMS-ESEM07.pdf

Bradberry, T. (2014). Multitasking damages your brain and career, new studies suggest. Retrieved June 10, 2015, from: http://www.forbes.com/sites/travisbradberry/2014/10/08/multitasking-damages-your-brain-and-career-new-studies-suggest/

Branch, R., (2009). *Instructional design: The ADDIE approach.* New York: Springer.

Cohn, M. (2013). *Succeeding with Agile: Software development using scrum*. Upper Saddle River, NJ: Addison-Wesley.

Denning, S. (2010). *The leader's guide to radical management: Reinventing the workplace for the 21st century.* San Francisco: John Wiley & Sons.

Drucker, P. (2001). *Management challenges for the 21st century.* New York: Harper Business.

George, C. (2014). Why limiting your work-in-progress matters. Retrieved June 10, 2015, from: http://leankit.com/blog/2014/03/limiting-work-in-progress/

Gorans, P., & Kruchten, P. (2014). *A guide to critical success factors in Agile delivery*. Retrieved June 10, 2015, from http://businessofgovernment.org/sites/default/files/A%20Guide%20to%20Critical%20Success%20Factors%20in%20Agile%20Delivery_0.pdf

Government Accounting Office. (2012). *Software development: Effective practices and federal challenges in applying agile methods.* Retrieved June 10, 2015, from http://www.gao.gov/assets/600/593091.pdf

Kniberg, H. (2010). *Kanban and scrum – making the most of both.* USA: C4Media.

Kundra, V. (2010). *25 point implementation plan to reform federal information technology management*. Retrieved June 10, 2015, from https://www.dhs.gov/sites/default/files/publications/digital-strategy/25-point-implementation-plan-to-reform-federal-it.pdf

Neibert, J. (2014). *Effective performance with A.G.I.L.E. instructional design.* Retrieved June 10, 2015,from http://www.learningsolutionsmag.com/articles/1346/effective-performance-with-agile-instructional-design

Northern, C., Mayfield, K., Benito, R., Casagni, M. (2010). *Handbook for implementing Agile in department of defense information technology acquisition.* Retrieved June 10, 2015, from *http://afei.org/WorkingGroups/ADAPT/Documents/Agile_Implementation_Handbook_2-3-11_public%20releaseable%5B1%5D.pdf*

Pentland, A. (2012). The New Science of Building Great Teams. *Harvard Business Review.* Retrieved June 10, 2015, from https://hbr.org/2012/04/the-new-science-of-building-great-teams

Sutherland, J. (2014). *Scrum: The art of doing twice the work in half the time.* New York: Crown Business.

Torrance Learning. *Get Agile! Business changes constantly – your project plan should at least keep up.* Retrieved June 10, 2015, from http://www.torrancelearning.com/agile/

U.S. Digital Service. (2014). *TechFAR handbook for procuring digital services using agile processes*. Retrieved June 10, 2015, from https://playbook.cio.gov/assets/TechFAR%20Handbook_2014-08-07.pdf