THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

COLLEGE OF INFORMATION SCIENCES AND TECHNOLOGY

APPLICATION OF AGILE METHODOLOGY IN INSTRUCTIONAL DESIGN

DREW ZUCKER
SPRING 2016

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Information Sciences and Technology
with honors in Information Sciences and Technology

Reviewed and approved* by the following:

Fred Fonseca
Associate Professor of Information Sciences and Technology
Thesis Supervisor

Steven Haynes
Senior Lecturer of Information Sciences and Technology
Honors Adviser

* Signatures are on file in the Schreyer Honors College.

ABSTRACT

This thesis explores the existing models that are being utilized by Instructional Designers and as well as some software development methodologies, with a strong focus on the Agile methodology. Following that, it makes recommendations on the possibility of a new and improved approach to instructional design for online courses by borrowing techniques from Agile. It concludes that a hybrid methodology that combines both the existing practices in the instructional design community with the currently leading programming methodology of Agile would improve online course design. It starts by exploring the existing models that are being utilized by Instructional Designers. Following this, it discusses the advantages and disadvantages of the Agile methodology. Finally, it proposes the new model, taking the best practices from methods old and new and forging them into one.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank the following people who helped make this thesis a reality. I truly could not have done it without them:

- Dr. Fred Fonseca, my thesis supervisor, for his assistance in helping me choose a topic and focus my research, as well as meeting with me for an entire year to make sure I completed this work.
- Dr. Steven Haynes, for serving as my honors advisor and offering his help and advice throughout the process.
- Dr. Lisa Lenze, for her continued support and guidance throughout my entire college career.
- Dr. Luke Zhang, for his assistance with the structuring of my thesis and his eagerness to answer any questions I had.
- My family, for their support, motivation, and love that allowed me to complete this work.

Chapter 1

Introduction

Dating back as far as the 1920s, technology and education have been closely tied (Purdue University, 2016). Early devices such as the overhead projector and on-air radio classes allowed for increased distribution of information from teachers to students. As technology has advanced, its application within the educational world has evolved as well. Handheld calculators allowed for faster, more complex calculations to take place, while videotapes and VCRs gave teachers the ability to display information more dynamically than with a simple chalkboard. Technologies' impact goes beyond the classroom though. With the rise of the Internet and the increase of personal computers, an entirely new form of education has been created in the form of online courses. While these online courses certainly exist due to the advances of modern technology, the influences for these courses are more than just physical devices.

Since their inception, online courses have been designed with specific practices based on popular software development methodologies of the time (Rawsthrone, 2005). Recently though, software methodology focus has moved from clear cut, step-by-step processes such as the Waterfall model, into a more fluid and feedback based methodology as seen in the Agile model. This same shift in focus has not yet occurred in the design of online courses and many have felt the hindrances of the current processes. With a paradigm shift currently underway in the field of software development, it is time to look to the future of instructional design of online courses with a focus on collaboration and efficiency. This is demonstrated clearly by this quote by Reuben Tozman of Learning Solutions Magazine, "… our work environment, our education, our

tools, and our learners have all changed. Yet we instructional designers still try to operate, and do our jobs, in the same way that we did them ten years ago" (Tozman, 2007).

Chapter 2

Instructional Design

What is Instructional Design?

Instructional Design is "…the systematic development of instructional specifications using learning and instructional theory to ensure the quality of instruction. It is the entire process of analysis of learning needs and goals and the development of a delivery system to meet those needs. It includes development of instructional materials and activities; and tryout and evaluation of all instruction and learner activities" (University of Michigan, 1996). Instructional design includes everything from deciding what the students are intended to learn, what information will be used to educate them, how that information will be presented, how the students will interact with the materials, and how they will be evaluated throughout the course. The process in which these decisions are made and implemented will be further referred to in this paper as Instructional Systems Design (ISD), while the individuals behind the design are known as Instructional Designers (ID). Preconceived instructional design models are utilized in order to assist IDs, supplying the designers with clear-cut steps to take and goals to reach in order to create an effective course. While dozens of models have been introduced with varying degrees of success, several have been widely recognized and become staples of the industry. ISD models that have been established and accepted by the instructional design community include the ADDIE (Rawsthrone, 2005) and Dick & Carey methods (Rawsthrone, 2005), as well as

ASSURE (Smaldino & Russell, 1999), Backwards Design (Wiggins & McTighe, 2005), and the Kemp Design model (Akbulut, 2007).

Methodologies

The ADDIE Model

Due to the evolving nature of the model, the first appearance of ADDIE cannot be solidly stated, but it is known to have shown up sometime during the 1970's (Allen, 2006). Originally, the methodology was invented for use by the US Army for training in the varied military roles that would be undertaken by entry-level soldiers. Soon after, the model was adapted to fit in any instructional or training environment. When broken down to its core, the bare bones of the model are the five steps Analysis, Design, Development, Implementation, and Evaluation. The model's flowchart can be found in Figure 1.

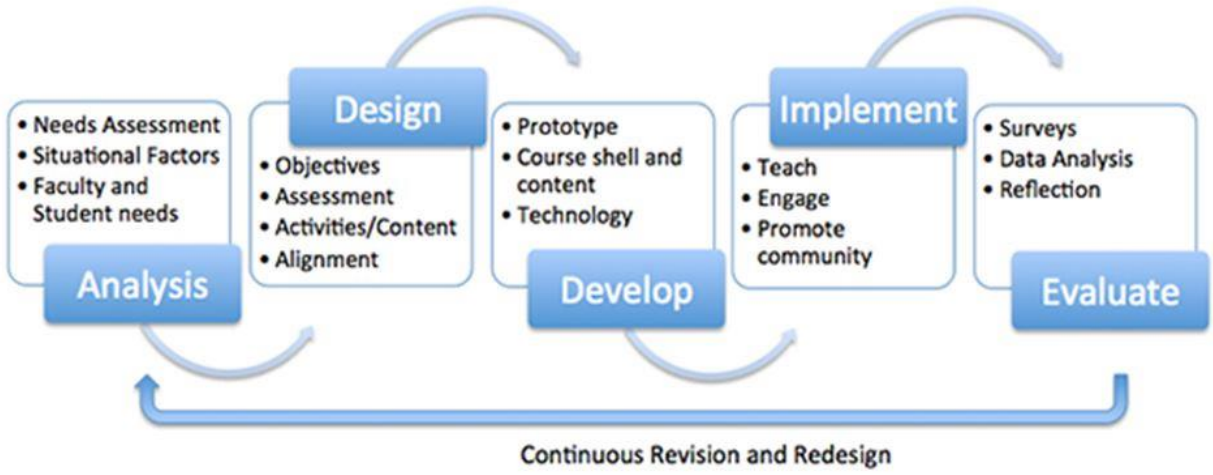

Figure 1. The ADDIE Model

The model is intended to follow a linear progression, where each step must be completed in order before the following step can be started. Many versions of ADDIE exist with varying sub-steps encompassed within each of the five core phases, but every variation is based around those main five segments. The Analysis phase is used for requirements gathering. In the context of course design, it is during this stage that the instructor decides who the target audience is, what the goals for the course are, and standards by which pupils will be evaluated. This is accomplished by considering several different variables. First, the background knowledge of the students must be taken into account. It is a waste of both the pupils' and the instructor's time to repeat information that is already known. The end goals for the course must also be considered. What are the instructor's goals in teaching this class and what are the learners' goals in taking it? Finally, the ID must determine the limiting factors that will affect overall course. These factors can include the length of the class, technical skills of the students, or the financial resources of the instructor/institution.

Following the Analysis step comes the Design phase. It is during this phase that the ID plans out how the course will be presented to and interacted upon by the students. The designer must decide what kind of media they will utilize to communicate the material to their learners. In addition to how this information will be presented, this phase is also used to determine when the information will be presented. In fact, it is during this step that the entire course schedule will be decided. This schedule should include when content is deployed, project deadlines, and assessment dates.

Only after all the planning has been completed can the ID continue on to applying their analysis and design in the Development phase. Development is the first step where something physical is actually being created (Shelton & Saltman, 2008). The phase makes use of the two

that come before it by reviewing the requirements gathering and tentative course schedule that have been completed and attempting to shape them into an actual course. Instructional materials, assessments, and course modules are all created during Development. Additionally, the media and methods that will be used to deliver this material to the students is decided upon and created.

Next, is the critical step of Implementation. As the name implies, this is the stage where a completed version of the course is finally presented to the class's students. Class is underway, and the planning, designing, and developing that have come before are put to the test in actual course situations, happening in real time. This is a crucial time to watch the course unfold and watchful eye for any problems the instructor, learner, or course itself may be experiencing. This assessing and monitoring of the course's effectiveness is so important that it earned its own place in the ADDIE model: Evaluation.

The final step of the ADDIE model has the purpose of deciding whether or not the initial learning goals set by the instructor were met by the pupils (Clark, 1995). All parties (the ID, the instructor, and the learners) must come together to determine what the course did well and what can be improved upon in the following version. Only after Evaluation has been finished will the ADDIE model of ISD be considered completed and the job of the Instructional Designer fulfilled.

Dick & Carey Model

Following the ADDIE model of design, the next most well known ISD methodology would be the Dick & Carey model. The model was originally conceived by Walter Dick and Lou Carey in 1978 (Dick, 1996). The methodology differs significantly from ADDIE in several ways.

From a quantitative perspective, Dick & Carey possess ten stages rather than the five steps of ADDIE. Additionally, the phases of the model are referred to as stages rather than steps because the Dick & Carey model is much less linear than ADDIE. While the stages do flow from one to another, strictly adhering to the ordering of the diagram in Figure 2 is not a necessity (Dick, The Dick and Carey Model: Will It Survive the Decade?, 1996). Dick & Carey looked at instruction with a systems view rather than as a sum of isolated parts (Dick, Carey, & Carey, The Systematic Design of Instruction, 1978). Due to this outlook, the Dick & Carey model does not necessarily need to be applied to an entire course, but could be used for a singular lesson or unit within the course. The ten stages outlined by the pair are:

1) Identify Instructional Goal(s)
2) Conduct Instructional Analysis
3) Analyze Learners and Contexts
4) Write Performance Objectives
5) Develop Assessment Instruments
6) Develop Instructional Strategy
7) Develop and Select Instructional Materials
8) Design and Conduct Formative Evaluation of Instruction
9) Revise Instruction
10) Design and Conduct Summative Evaluation

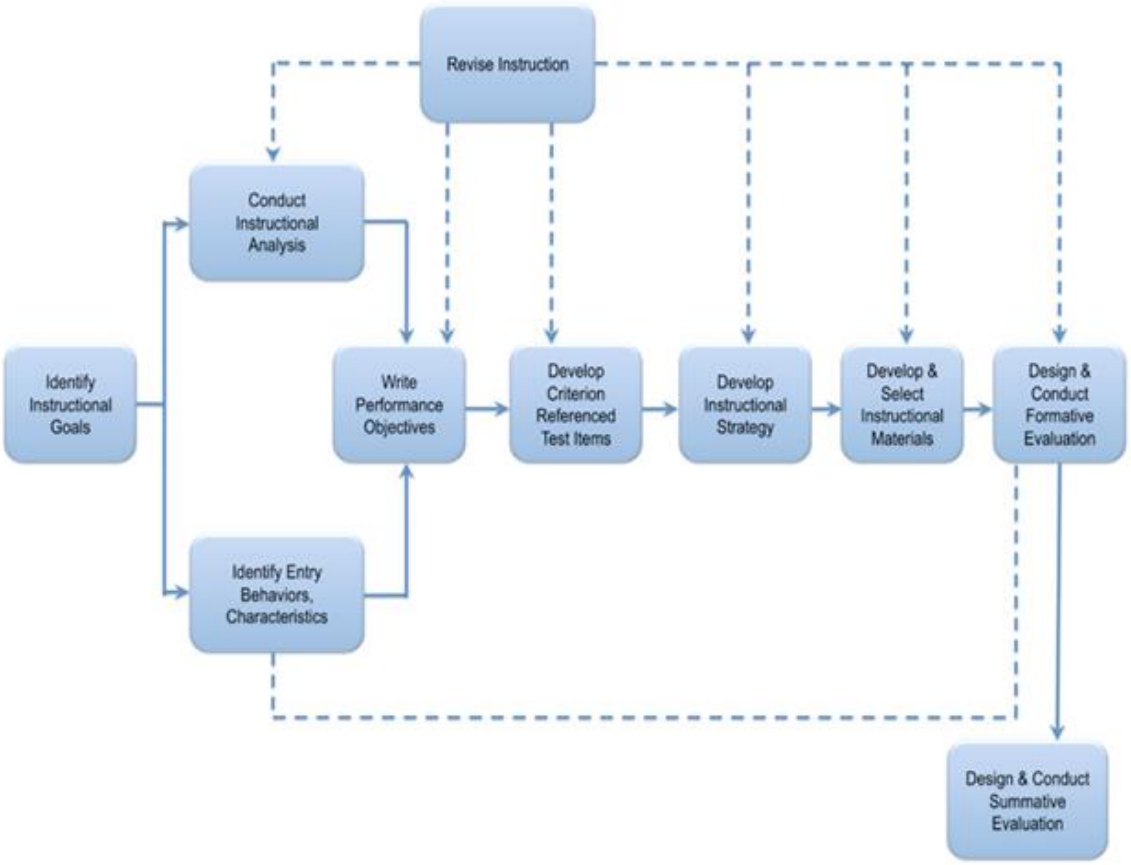

Figure 2. Dick & Carey Model

Due to the descriptive nature of stage titles, I will only briefly describe what is to be accomplished in each stage. Stage 1 consists of identifying exactly what it is that the students will be expected to learn from the course. Stage 2 is used to determine what prior knowledge and skills each student *should* possess in order to complete the instructional goal. Stage 3 is used to determine what prior knowledge and skills each student *does* possess. Stage 4 will be a written, detailed objective that the students' performance can be judged against. Stage 5 is used for the creation of test items that will provide checkpoints for the learners and educators on what the students have or have not mastered (Forest, 2013). Stage 6 is where the lesson plans are created and decisions are made on how the content will be delivered to the learners. Stage 7 is the actual

implementation of the decisions made during the previous stage. Stage 8 is an evaluation stage in which the ID attempts to determine what went well with the course or lesson, and what could use improvement. Stage 9 is the time for revision to the instruction based on the formative evaluation of Stage 8. Finally, Stage 10 is a chance for the ID to evaluate the entire process, start to finish, and decide how to improve for the following iteration of the course or lesson (Forest, 2013).

Despite the differences between the models, ADDIE and Dick & Carey also share many similarities. The first three stages of Dick & Carey could be compared to the Analyze step of ADDIE, stage four being Design, stages five and six serve as Development, stage seven is Implementation, and finally stages eight through ten are comparable to ADDIE's Evaluation. These comparisons are not exact and the approach that an ID would attempt each model is fairly distinct, but the associations between them help to highlight the fact that any issue with either model could be reflected in the other.

Other Instructional Systems Design Models

Additional ISD models do exist and are utilized by some, yet ADDIE and Dick and Carey are by far the most popular (Rawsthrone, 2005). Examples of some of the other models are ASSURE, Backwards Design, and the Kemp Design model (Forest, 2013). ASSURE is an acronym standing for Analyze Learners, State Objectives, Select Materials, Utilize Materials, Require Learner Response, and Evaluation. "This model is designed to help you effectively integrate media/technology into your lesson or presentation - to help "assure" learning" (Smaldino & Russell, 1999). Backwards Design has three stages: Identify Desired Results, Determine Acceptable Evidence, and Plan Learning Experiences and Instruction (Wiggins &

McTighe, 2005). The main idea with this model is to start with the end goals and create instruction based on meeting those goals. The Kemp Design model uses nine-step circular structure rather than the more popular linear models of ADDIE and Dick and Carey. This allows for increased flexibility in the job of the Instructional Designer (Akbulut, 2007). No further detail is required for these three models, as they are only examples of alternatives to the big two (ADDIE and Dick and Carey). Moreover, a more careful study at the alternatives will reveal that they are merely variations of ADDIE, Dick and Carey, or both.

Instructional Design of Online Courses

With the advent of online education, instructional design has had to adapt to the more varied and diverse environment of learning. These models, that were originally created with physical classrooms in mind, have to evolve in order to meet the needs and desires of educators and learners alike in the modern age of technology. Rather than start from scratch and create new models from the ground up, most Instructional Designers have instead elected to take the models used for physical settings and simply apply them in the same way for courses that will be taught online (Shelton & Saltman, 2008). This has only worked with varying degrees of success due to the additional and complex factors online education adds to the learning environment. New roles that were never before considered such as software engineer, web programmer, and graphical interface designer must now be taken into account. The means by which the information is presented and delivered to students is another difficulty that is greatly increased when dealing in virtual space. While in the past the educators could double as the Instructional Designer, it is now a more common practice for the ID to be a completely separate individual from the

instructor due to the need for more specific knowledge of the field. One example of an online-specific methodology that has been attempted is the Virtual University (VU) approach from Michigan State University (Koehler, Mishra, Hershey, & Peruski, 2004). As demonstrated in Figure 3, the VU approach makes use of three different team members when designing their courses. There are the design programmers who create the widgets available for online course use, the faculty member who will be serving as the instructor for the particular course, and the "producer" who combines the desires of the instructor with the content made available by the programmer. Similar models have been attempted, but all of them possess complicated issues of their own which will be discussed later in this paper.

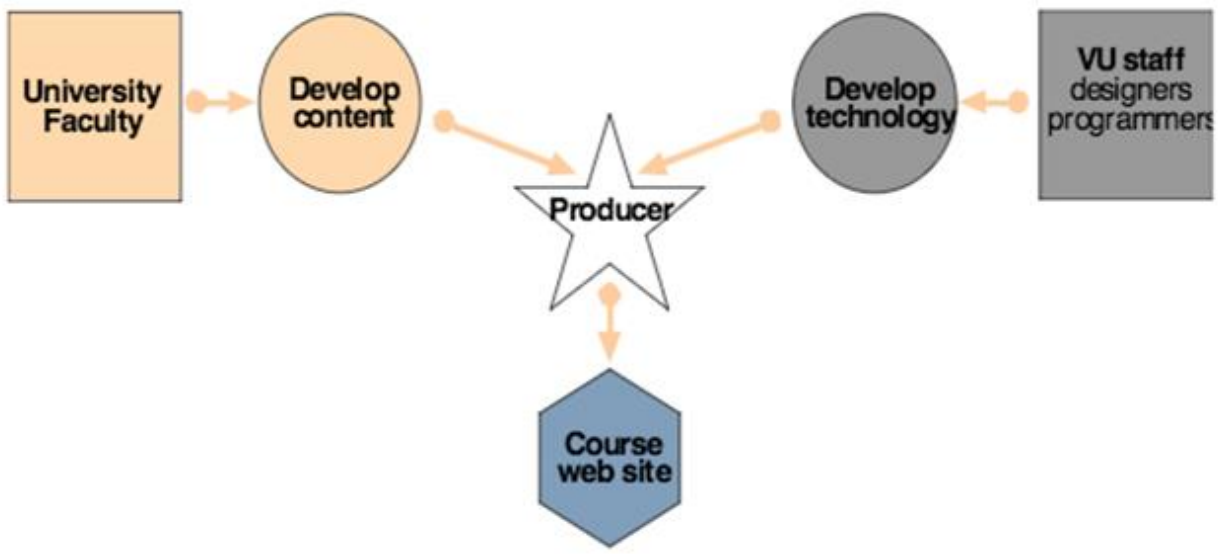

Figure 3. Virtual University Model

Chapter 3

Agile Methodology

Just as ISD methodologies are now in need of an update or possibly a new model altogether, so too were software development methodologies during the 1990s. Many software engineers of the time complained that the widely used Waterfall methodology was hindering the development of software rather than assisting it (Leffingwell, 2007). Waterfall methodology consists of five steps with each leading into the next. The steps are Requirements, Design, Implementation, Verification, and Maintenance. It is to be noted that this model greatly represents ADDIE. Similar to the ADDIE model, this model is meant to be followed linearly, and it is that linear structure which caused so much trouble for the software development community. Developers were expected to complete all of their requirements gathering in the very first step of the process. This meant that they were not supposed to add in any additional requirements following this step. Since much functionality that is needed in any program is often found during its development, or later during its testing, this upfront requirement gathering proved to be much too restrictive and ultimately detrimental. Software was being delivered unfinished, or teams were missing deadlines altogether. In order to remedy these issues, and several others that came along with Waterfall and other linear models, Agile methodology was officially introduced in 2001 (Fowler, 2005). An image of its flowchart can be found in Figure 4.

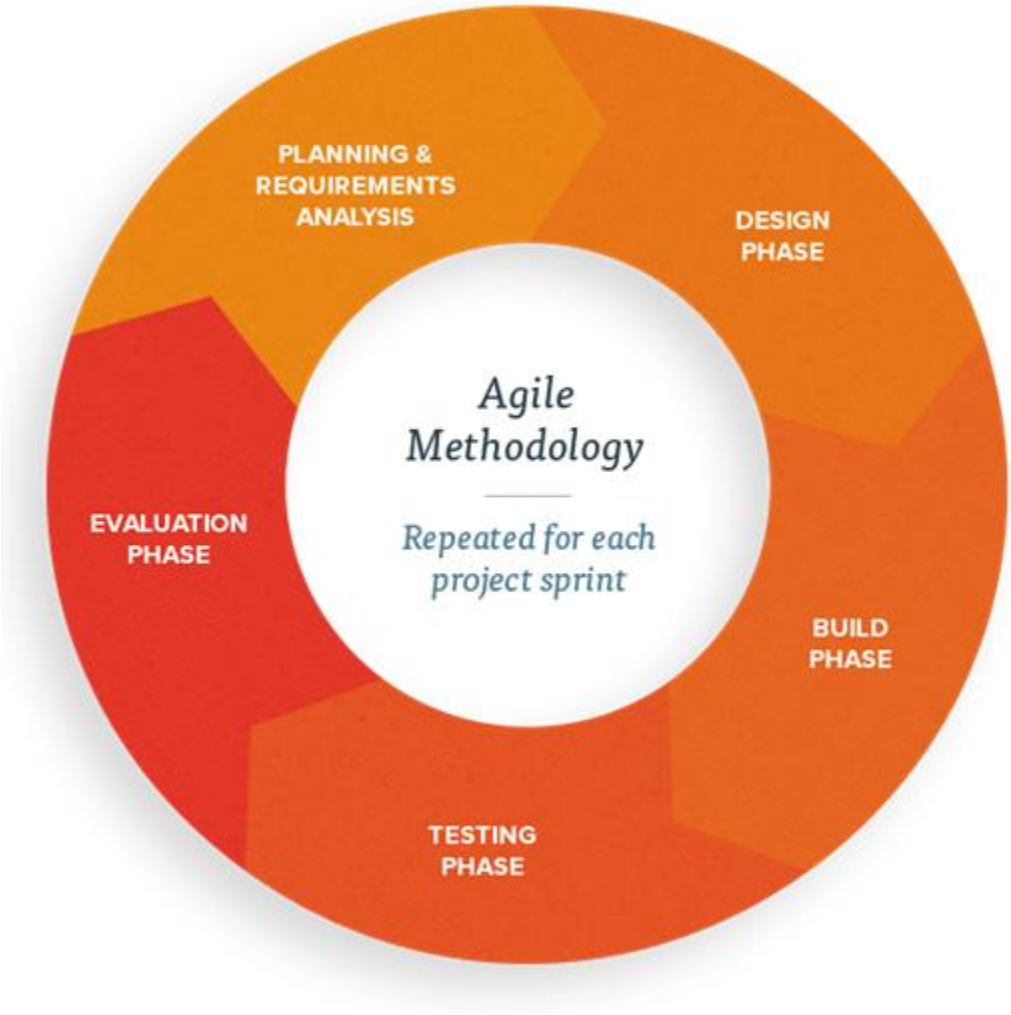

Figure 4. The Agile Methodology

A manifesto was released in order to introduce the world to this new methodology and its values. The Manifesto for Agile Software Development reads: "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more." (Manifesto for Agile Development, 2001).

Bertrand Meyer, author of "Agile!: The Good, the Hype, and the Ugly", further summarizes the manifesto to these five values of Agile:

1) Redefined roles for developers, managers and customers.
2) No "Big Upfront" steps.
3) Iterative development.
4) Limited, negotiated functionality.
5) Focus on quality, understood as achieved through testing.

Together, these values aim to rectify the issues that the Waterfall model caused for many programmers. Firstly, roles have been redefined for every person involved in the creation of the software. Developers and managers are no longer mutually exclusive as Agile stresses self-organizing, cross-functional teams. Teams assign their own leader, called a scrum master, who keeps the team and project on track. The customer is also a part of the team in the form of a customer representative called the product owner. The product owner is the voice of the customers and communicates the wants and needs of the end user. This is a great contrast to the passive role customers have played in the past (Meyer, 2014). "No big upfront steps" refers to the extensive planning and documentation that went along with the Requirements and Design stages of the Waterfall model. Agile methodology strongly supports cutting back on documentation and beginning to code almost immediately. Gathering requirements is instead achieved by constant discussion with the product owner. Working code is a huge tenet of Agile programming, and for this reason the development process is broken up into short (usually four week) "sprints". At the beginning of each sprint the team decides on a reasonable list of

functions they can have coded and working by the end of the sprint. Short daily gatherings (scrum meetings), allow for each member of the team to vocalize their accomplishments and issues since the day before, as well as state the goals they plan to attack that day. Creating the program one feature at a time is part of Agile's "iterative" development, arguably the most important principle of Agile. By the end of each sprint, the team should be able to produce a completely workable version of the program. Coupled with this is the idea of "limited, negotiated functionality". This ideal stresses only implementing functionality that customers will actually use and never sacrificing meeting a deadline in order to add additional functionality. The final value, as outlined by Meyer, deals with the continuous testing needed to ensure the quality of the code. All functionality must completely pass all of its tests for it to be considered successfully implemented in the eyes of Agile programmers.

Advantages of Agile Methodology

The Agile methodology, its values, and its principles offer many advantages, not only in software development, but also in any product development process in which it is applied (West & Grant, 2010). One of the biggest and most appealing of which are the time and cost benefits of the methodology. Due to the greatly increased focus on development rather than documentation, projects that utilize Agile often get off the ground quicker than projects that use more analysis based models. Coupled with the Agile ideas of time-boxed development and the commitment to meeting product deadlines with no excuses, projects can be completed and delivered to market 25%-50% faster than those using other methods (Reifer, 2002). These same principles have contributed to an average of 5%-7% cost decrease with Agile-minded production (Reifer, 2002).

Many times, cheaper applications that take less time to develop result in lower quality programs. This is not the case when Agile is utilized however. Software developers who have applied the Agile methodology to their projects have reported that their software has fewer bugs and more stable functionality (Begel & Nagappan, 2007). This can be attributed to the test-driven design that the model advocates. By testing each and every feature as they are developed and implemented, the programmers are able find more of the defects and glitches that have the potential to plague the system. Furthermore, by focusing on only a small number of features during each sprint, the developers are able to make sure every function works exactly as it is supposed to prior to the official release. If any bugs within the system are discovered, the increase in communication due to daily scrum meetings make the fixing of those issues much more efficient. Improved communication itself is a major advantage of Agile, not only allowing for higher quality products, but also a better understanding of customer wants and needs. With the end user having a representative on the team in way of the product owner, customer focus and satisfaction are increased (Begel & Nagappan, 2007). Daily meetings also make for better team morale by allowing the entire group to be more aware about what they need to work on, and what has been accomplished, and where the project stands overall.

Disadvantages of Agile Methodology

Although Agile offers many benefits, it is not without its faults. One of the main problems that the methodology presents is with large-scale, distributed teams. Daily scrum meetings increase communication, but if teams are spread out in different office locations, or even among different floors, it can be difficult for these meetings to occur. Some have also

argued that the meetings were too frequent and distracting (Begel & Nagappan, 2007). If the meetings are not run efficiently, they can quickly become a waste of time. When time-wasting meetings are a daily occurrence, deadlines can be missed, team member morale will weaken, and the overall quality of the product will diminish. Another issue the model has shown is an inability to accurately predict schedules and costs. The Agile principle of no "Big Upfront Anything" attempts to dismiss the need for gathering requirements or planning in general prior to development (Meyer, 2014). This can make it extremely difficult to predict how much the project will cost in the end. While studies have shown that Agile methods decrease costs in general, not being able to estimate that cost with confidence can make it problematic to get a project green lit to begin production. In addition, Agile teams have claimed that they occasionally lose sight of the big picture because they are too concerned with the feature of a particular sprint. As one team member put it, "The focus is on today's work more than what the feature team is trying to achieve" (Begel & Nagappan, 2007).

Chapter 4

Proposed Hybrid Methodology

Issues with Current Methodologies

Prior to exploring the hybrid methodology that is the purpose of this paper, it is useful to first discuss the many issues that hinder the ISD models (specifically for online courses) being used today. Many of the problems that ISD models face are the same issues that plague sequential software methodologies such as Waterfall. Criticisms are mainly aimed at the lack of opportunity to change requirements or incorporate user requested features after development has begun. When utilizing the ADDIE method, designers are expected to gather all requirements of the course and system during the Analysis phase. The very first step of the model is the only one where IDs are able to decide on the goals and objectives of the instructors, students, and classes. This becomes a major issue when a problem in the course is discovered later on. Instead of being able to adapt and re-release a working version in real time, Instructional Designers are forced to wait until there is an entire project redesign. Not only does this waste the time and learning potential of the pupils, but it also leads to an increase in cost.

Another obstacle of the ADDIE model is the fact that the development phase does not occur until halfway through the practice. Since there is no working version until relatively late in the process, even if there is a glitch that can be fixed or a feature that can be added, they will be found too late to be able to be corrected. This also makes it impossible for the ID to design the course with the preferences and needs of the students playing an integral part. Both the ADDIE

and Dick & Carey models attempt to remedy this issue with the inclusion of their final stages of Evaluation. “The evaluation stage main goal is to determine if the goals have been met and know what will be required moving forward in order to further the efficiency and success rate of the project” (Forest, 2013). While this is of course incredibly important for improving the program for the following iteration, it does next to nothing for the students who are forced to struggle through the flawed original version.

These issues and others only become more apparent when considering online courses particularly. As Tozman put it, “However, our work environment, our education, our tools, and our learners have all changed. Yet we instructional designers still try to operate, and do our jobs, in the same way that we did them ten years ago” (Tozman, 2007). In a physical classroom environment, educators are able to adapt their lesson plans and teaching methods on the fly based on the rapport and feedback from their students. At the moment, online instructors do not share these luxuries, and the effectiveness of the courses suffers as a result. Even when online-specific models are introduced, such as the VU approach from Michigan State University discussed earlier, the results are not favorable. The division of labor of the model (a faculty member creates the content, a programmer designs the application, and a producer brings it all together) causes a lack of the necessary communication for a successful course to be created. The instructor and programmer never communicate directly, only through the producer. This disconnect makes it difficult for either member to properly present their ideas to the other. Furthermore, the programmer creates general widgets that in theory could be used in any number of courses, regardless of content or teaching style. In reality, this restricts the producers to only using the widgets created no matter what content they are given by the educators. This robs the instructor of any unique or distinct teaching style they may have wanted to utilize (Koehler, Mishra,

Hershey, & Peruski, 2004). The issue of late stage evaluation is also present in this model, not only for the students but for the instructor as well. The faculty is not given a real chance to work with the software until they are teaching the actual course. This means they are not only unfamiliar with their own course software, but if they have any suggestions or requests regarding it, it is already too late.

Even worse, is when no online-specific model is attempted at all. In this case, IDs attempt to simply present an already existing physical course through a new medium (the Internet). This practice of "content that is simply moved from one medium to another without regard for the capabilities of that medium" (Shelton & Saltman, 2008) is referred to as "shovelware" by Alistair Fraser of the "Chronicle of Higher Education". Some Instructional Designers, when faced with the issue of not having an established ISD model specifically engineered for online use, fall into the trap of creating shovelware. This means they are ignoring the potential benefits that online education brings to the table, as well as relying on the hope that the educational content can be successfully delivered to students by use of strictly text-based material. Teaching and learning require far more than the distribution of text, and it is for this reason that shovelware is an unacceptable option in online education.

Recommendations

It is clear from the numerous issues cited by Instructional Designers that the current ISD models are insufficient for the design of online courses. Software development methodologies have had a strong influence over ISD models for a long time. Peter Rawsthorne of Cape Breton University states "The influence of software engineering methodologies and practices over

Instructional Systems Design (ISD) has been ongoing since Computer Based Training (CBT) has been available" (Rawsthrone, 2005). With more and more developers making the switch from linear, sequential models to Agile methods, it is time IDs look to apply this switch to iterative development in their models as well. However, IDs should not just blindly adopt all of the Agile practices. Instead, Instructional Designers should pick and choose which of the Agile values best supplement the ISD models. Agile has been shown to work best in other industries when its methods are blended with the already existing methods of said industries (West & Grant, 2010). Keeping this in mind, it is time to discuss which parts of the process will be beneficial and which will be better left behind.

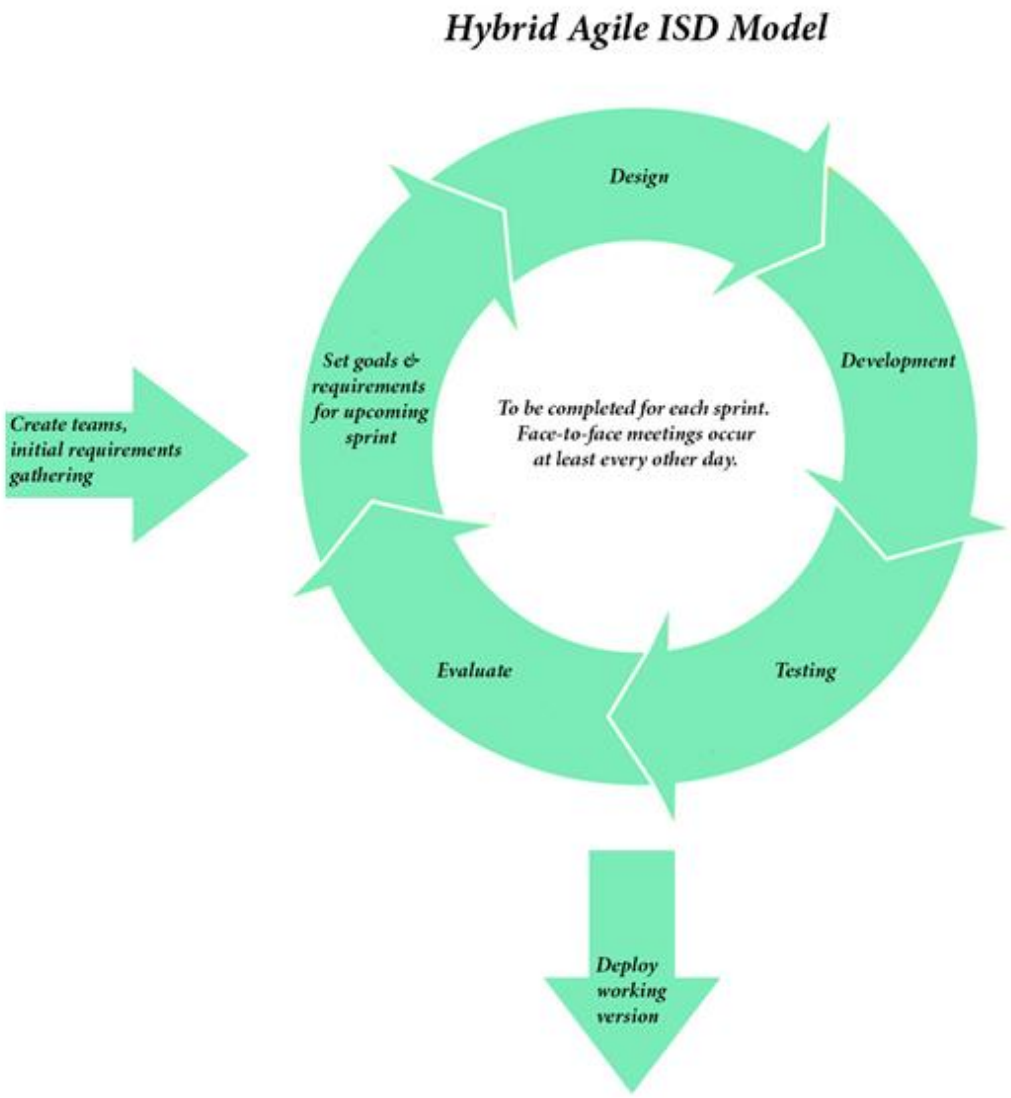

Figure 5. The Hybrid Model

Cross-functional Teams

One of the most necessary Agile principles that should be utilized in ISD models is the idea of cross-functional teams. As evidenced by Michigan State's Virtual University, it is important to have a team with members chosen whose skills vary greatly from one another, each with an expertise in a crucial facet of instructional design. Group members should consist of at

least the following: an ID, a web designer, and a course instructor. Other, optional team members could be a graphic interface designer, a volunteer graduate student, or even just additional IDs, web designers, or instructors to add more varied opinions. The Instructional Designer would serve as the team's scrum master. In this position they would be the ones running scrum meetings and making sure everyone is communicating and on track. The designer is chosen for this role because they are ultimately the ones bringing everything together. The web designer and GUI expert would serve as regular team members available to offer their knowledge and skills in designing how the course should look and function for its users. The course instructor should serve as the product owner since the end product is as much for them as it is for their students. In this role they would act as the customer representative, supplying the team with wants and needs of the end users. The graduate student could also assist by being an example voice of the students. In addition to testing the system and offering user insight, the course instructor will of course also be the one making the final decisions in what content is delivered to the students. Since each team member's role is fairly straightforward, the Agile practice of self-organizing teams can be disposed of.

In the end, this team and the functions of each member do strike a strong resemblance to the VU teams. The biggest difference between the two is the increase in collaboration and communication among the Agile team. Unlike the VU teams, every member will have direct contact with every other member of the team. This will allow for greater understanding of the goals and requirements of the course. It will also allow for the instructors to achieve a more distinct teaching style, despite the limitations set upon them by the virtual aspects of online education. Finally, by giving the course instructor access to both the web designer and ID, the likelihood of "shovelware" is decreased by a significant degree. The resulting course will be a

much more natural fit for the virtual space rather than just a physical course translated to text-only form.

Sprints & Daily Meetings

A major criticism aimed at Agile methods is the argument that the daily scrum meetings are in fact more distracting than beneficial. Still, constant communication and reporting of progress is extremely important for effective Agile development. For this reason, it is recommended that sprint meetings be held at least 2-3 times per week. This allows for adequate face-to-face time for the group without becoming excessive or tedious. The length of each sprint should remain fairly similar to that of regular Agile projects, usually somewhere between two and four weeks. The number of weekly meetings and the length of each sprint are decisions ultimately made by the ID.

Analysis Phase

Another Agile practice that must be tweaked is the "depreciation of upfront tasks" (Meyer, 2014). Agile looks down on trying to gather requirements prior to the beginning of production. It is true that attempting to try and pin down each and every requirement of a course before its creation is underway is an exercise in futility. Despite this, there is no reason for "shunning the… studying of a problem before attempting to solve it, and of defining the architecture of the solution before embarking on the details" (Meyer, 2014). To remedy this, a stage similar to ADDIE's Analysis should be utilized, although it should be limited to a few weeks at most to prevent the excess documentation or planning that can bog down a project.

Iterative & Test-driven Development

The next major practice to be adopted is that of iterative, test-driven development. Agile projects boast higher quality products produced in less time due to having working models that can be continually tested. ISD models would be wise to do the same. At the close of each sprint, a working model of the course should be available for users to test and evaluate. Maria Puzziferro and Kaye Shelton of Colorado State University wrote a paper detailing the need for more high-quality online course development and emphasized, "the objective is to have a… model online "demo" course to which the instructional development team can benchmark their progress" (Puzziferro & Shelton). This further supports the claim that ISD models should be continually testing and improving their system as development goes on. During the initial development of the course, these testers will be the course instructor and the optional graduate student that has volunteered to assist the team. Every piece of functionality within the system should have at least one test associated with it in order to make sure everything runs smoothly. After the semester is underway and the course is live, the testers will also include the actual students of the course. A biweekly survey should be distributed that asks the students what works well with the system, what could be improved, and if they experienced any bugs or glitches during their time using it. While in the past this information would only be valuable for the following semester, thanks to iterative development the ID will be able to apply updates in real time. This means the whole team must stay intact throughout the entirety of the semester.

Time-boxing

Time-boxing every iteration is another process that will ensure the quality and stability of the project throughout. Time-boxing entails making sure that every deadline set by the team is met with no excuses. It requires the team to consider what they can realistically achieve carefully, instead of making promises that cannot be met. This also means the project will get completed with minimal delays and costs will be kept within predictions. The "closed-window rule" is closely tied with time-boxing and should be applied as well. The rule prohibits any team member from adding functionality that was not originally planned for within the current sprint. This is just another regulation that keeps the team on track and discourages adding unneeded or wasteful features.

The Hybrid Methodology

In summary, the hybrid ISD methodology should be as follows. The design team should consist of members with differing areas of expertise, but the team should not be self-organizing. The minimal requirements for this team are an Instructional Designer who will serve as the scrum master, a course instructor who will serve as the product owner, and a web designer who will be the technological lead. A requirements gathering stage should exist, but it should last no longer than two or three weeks. Development should be broken up into sprints lasting a few weeks at a time. A working version of the course should be delivered at the close of every sprint. Scrum meetings should be changed from daily to several times a week, but face-to-face communication between all group members is still highly encouraged. At each meeting, every member of the group will present what they have accomplished, what they are stuck on, and

what they plan on having done by the following meeting. The goals of a sprint should be laid out before it is underway and only those functions originally intended will be attempted during that sprint. Missing deadlines is unacceptable. Every function or feature that is added to the system must have at least one test associated with it to ensure performance. During initial development the course will be tested by the course instructor and optional graduate volunteer. Following implementation (the start of the semester) the course will continue to be tested the students taking it. A survey will be distributed to the students every other week to collect their feedback. The course will then continue to be updated and improved throughout the semester until the class comes to an end. For a visual of this model, refer to Figure 5.

Conclusion

ISD models have closely followed the patterns of the software development methodology field for years. Just as software developers have shifted their focus to Agile practices that emphasize iterative and test-driven development, so must Instructional Designers. However, not all Agile values are as important as others and some are not needed all together. Instead, a hybrid methodology that is some parts Agile, some older ISD models, and some original ideas should be applied. The principle tenets to be practiced are cross-functional teams, shorter requirements gathering, iterative "sprint" design, test-driven development, increased face-to-face communication, time-boxing, and the delivery of a working system at the end of each iteration. In order to truly know the effects the usage of such a model would have, it would have to be used to create multiple courses and survey both the designers of the system and the end-users. The hypothesis, however, is that the application of this hybrid methodology would result in higher

quality courses, cheaper and shorter development, and increased satisfaction from both students and teachers alike.

BIBLIOGRAPHY

Akbulut, Y. (2007, April). IMPLICATIONS OF TWO WE LL-KNOWN MODELS FOR INSTRUCTIONAL DESIGNERS IN DISTANCE EDUCATION: DICK-CAREY VERSUS MORRISON-ROSS-KEMP. *Turkish Online Journal of Distance Education* .

Allen, W. C. (2006, November). *Overview and Evolution of the ADDIE Training System.* Retrieved October 21, 2015, from ProQuest: http://ezaccess.libraries.psu.edu/login?url=http://search.proquest.com.ezaccess.libraries.psu.edu/docview/221180962?accountid=13158

Begel, A., & Nagappan, N. (2007, September). *Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study*. Retrieved January 2016, from Microsoft Research: http://research.microsoft.com/apps/pubs/default.aspx?id=56015

Clark, D. (1995, July 13). *Evaluating Instructional Design*. Retrieved October 21, 2015, from Big Dog, Little Dog: http://www.nwlink.com/~donclark/hrd/sat6.html

Dick, W. (1996, September). The Dick and Carey Model: Will It Survive the Decade? *Educational Technology Research and Development* .

Dick, W., Carey, L., & Carey, J. O. (1978). *The Systematic Design of Instruction.* Pearson.

Forest, E. (2013, November 13). *Dick and Carey Instructional Model*. Retrieved November 2, 2015, from Educational Technology: http://educationaltechnology.net/dick-and-carey-instructional-model/

Fowler, M. (2005, December 13). *The New Methodology*. Retrieved September 13, 2015, from Martin Fowler: http://www.martinfowler.com/articles/newMethodology.html

Koehler, J. M., Mishra, P., Hershey, K., & Peruski, L. (2004). With a Little Help From Your Students: A New Model for Faculty Development and Online Course Design. *Journal of Technology and Teacher Education* .

Leffingwell, D. (2007). *Scaling Software Agility.* Addison-Wesley Professional.

Manifesto for Agile Development. (2001). Retrieved March 20, 2015, from Agile Manifesto: agilemanifesto.org

Meyer, B. (2014). *Agile! The Good, the Hype, and the Ugly.* Springer.

Purdue University. (n.d.). *The Evolution of Technology in the Classroom*. Retrieved December 11, 2015, from Purdue University Online: http://online.purdue.edu/ldt/learning-design-technology/resources/evolution-technology-classroom

Puzziferro, M., & Shelton, K. (n.d.). A MODEL FOR DEVELOPING HIGH-QUALITY ONLINE COURSES: INTEGRATING A SYSTEMS APPROACH WITH LEARNING THEORY. *Journal of Asynchronous Learning Networks* .

Rawsthrone, P. (2005, December 10). *Agile Methods of Software Engineering s hould Continue to have an Influence over Instructional Design Methodologies.* Retrieved August 24, 2015, from Agile Instructional Design: http://cmapspublic.ihmc.us/rid=1138535398625_1951587770_6507/Agile%20Instructional%20Design.pdf

Reifer, D. (2002). *How Good are Agile Methods?* Reifer Consultants.

Smaldino, S. E., & Russell, J. D. (1999). *Instructional Technology and Media for Learning.* Pearson.

Shelton, K., & Saltsman, G. (2008). *Adapting Information and Communication Technologies for Effective Education.* Hershey, PA, USA: Information Science Reference.

Tozman, R. (2007, December 10). *The Next Generation of Instructional Designer*. Retrieved September 16, 2015, from Learning Solutions Magazine: http://www.learningsolutionsmag.com/articles/120/the-next-generation-of-instructional-designer

University of Michigan. (1996, October 18). *Definitions of Instructional Design*. Retrieved September 16, 2015, from University of Michigan: http://www.umich.edu/~ed626/define.html

West, D., & Grant, T. (2010). *Agile Development: Mainstream Adoption Has Changed Agility.* Forrester.

Wiggins, G., & McTighe, J. (2005). *Understanding by Design.* Assn. for Supervision & Curriculum Development.

ACADEMIC VITA

Academic Vita of Drew Zucker
Dwz5051@psu.edu

Education

B.S. in Information Sciences and Technology, Penn State University

Honors in Information Sciences and Technology

Thesis Title: Application of Agile Methodology in Instructional Design

Thesis Supervisor: Fred Fonseca

Work Experience

Software Engineer Intern

Summer 2015

Coded, improved, and debugged existing company programs.

PNC Bank, Philadelphia PA

Jim Snyder

Awards and Activities

Awards: Dean's List, Schreyer Honors Academic Excellence Award, College of IST Internal Scholarship, Lewis C. Cowley Scholarship, Francis R. and Helen M. Pentz Memorial Excellence Scholarship

Community Service Involvement: THON Dancer, THON Chairman- Sigma Alpha Mu, Philanthropy Chairman- Sigma Alpha Mu

International Education: Anglo-American University