ADAPTIVE-ADDIE: SCRUM FRAMEWORK FOR INSTRUCTIONAL SYSTEMS DESIGN

Ranvir Singh Bahl

University of British Columbia

ETEC 511 – Section 64D

Dr. Matiul Alam

2012

Abstract

Instructional systems design (ISD) is a complex intellectual process. It requires higher-level cognitive thinking integrated with professional skills and abilities to systematically design training solutions (Nelson, Macliaro & Sherman, 1988). It is hard to argue that ISD requires proper planning, analysis and design which reinforces why instructional purists revere Dick and Carey's well known ADDIE model even today. However many instructional designers agree that ADDIE is 'showing its age' and needs agility to adapt to the shifting learning needs, training goals in today's knowledge economy. Classic models like ADDIE provide a structured, linear approach to ISD very similar to the Waterfall model in software development. The traditional approach works great when learning needs are very clear, instructional context is stable and the possibility of changes during development are minimal. In today's competitive work environment, change is inevitable and customers are looking for flexible systems that embrace change rather than control it. In this paper I review the limitations of linear ADDIE approach and propose Adaptive-ADDIE, a hybrid ISD approach that blends best practices from Agile/Scrum methodology with ADDIE to provide a collaborative, value driven instructional design methodology.

The ADDIE Model

Instructional Systems Design (ISD) is systems based approach for creating instructional experiences that allows for sound decision making by obtaining overall view of the learning process and developing required support material in an effective and efficient manner. ISD is sometimes referred to as ADDIE (Analysis, Design, Development, Implement, and Evaluate) or SAT (Systems Approach to Training). ADDIE was developed at Florida State University in 1975

and was selected by the US military as the primary methodology for developing training. Most of the current ISD models are variations of ADDIE model and follow five phase progression:

- Analyze: In this initial phase, the instructional problem is clarified, instructional goals and objectives are established and learning outcomes are identified. From a process standpoint, the instructional designer interviews various stakeholders to understand the learning environment, user personas, identify their existing knowledge and skills gaps, behavioural outcomes, pedagogical considerations and learning constraints as applicable. From the project management standpoint, it is important to understand the project scope (business, functional and non-functional requirements), delivery timelines, allocated budget and acceptance criteria in order to ascertain the work gets completed 'on time and within budget'. In corporate training, it is vital to understand how the behavioural outcomes will be measured against the business objectives. For instance, the metrics for successful training for a business could be x% increase in sales figures or y% increase in customer satisfaction.
- Design: With a good understanding of the instructional and project goals, next comes the design phase which typically includes drafting design specifications, preparation of course outline, user interface (UX) design, storyboarding and a prototype design (blueprint) that closely resembles the final outcome. At the end of this stage, there is typically a 'contract sign-off' and any changes moving forward usually go through a change management process.
- Develop: Next is the development phase which includes execution of the design, content development and multimedia production using appropriate software development tools.
- Implement: Once the courseware is developed, it's time to test it to make sure it works properly before deploying in production. This step includes quality assurance, content

deployment on the selected delivery environment (i.e., LMS or website) and providing appropriate training to instructors and learners to use the instructional system effectively.

- Evaluate: This final phase includes formative and summative evaluation of the instructional system against the success/acceptance criteria defined in the analysis and design phase to assess how well the courseware met the instructors and learners expectations.

Figure 1 provides a graphical representation of the ADDIE model.

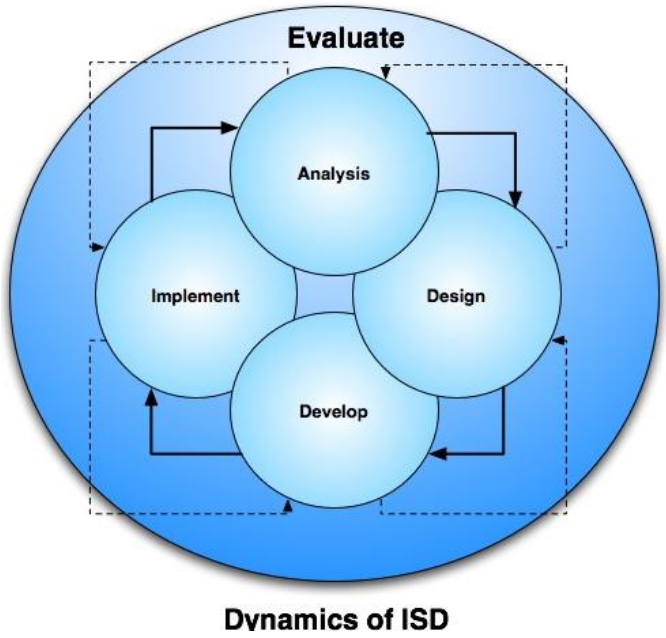

Figure1: Dynamics of ISD. Retrieved July 26, 2012 from http://www.nwlink.com/~donclark/hrd/sat1.html

Limitations of the ADDIE model

The following illustration describes the limitations of linear instructional system design process:

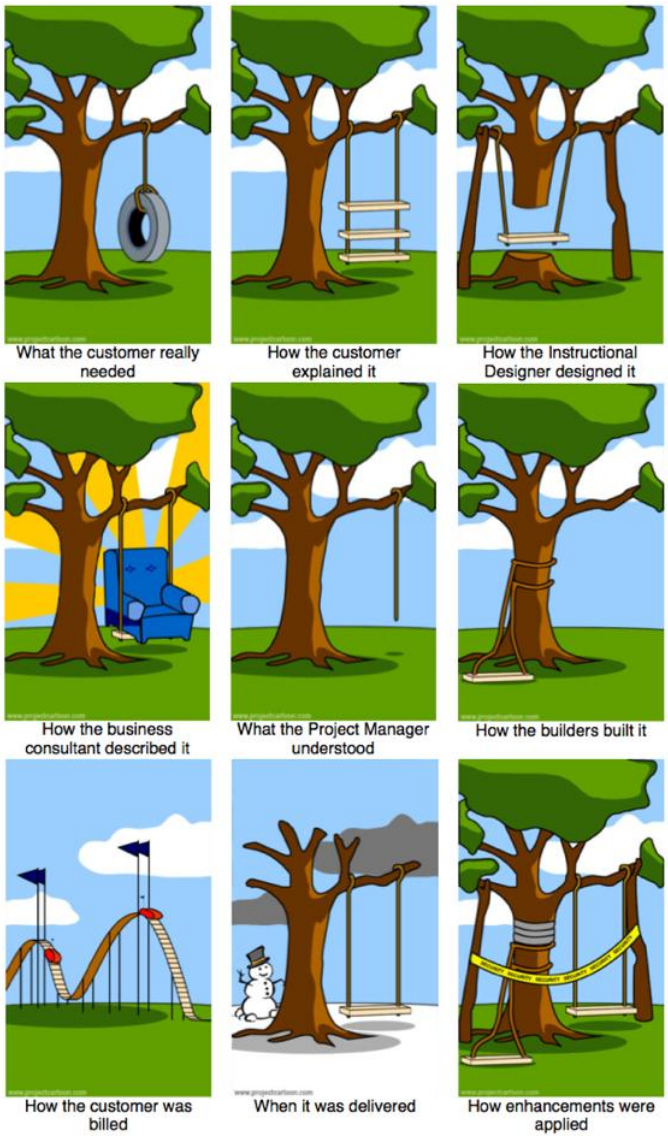

Figure 2: Project Management cartoon. Retrieved June 17, 2012 from http://www.projectcartoon.com

Figure 2 describes a common problem experienced in educational technology projects if we follow the ADDIE model similar to classic Waterfall software development model. ADDIE works great as a generic framework, however it can present some challenges if followed

dogmatically. For instance, if learner needs were not identified appropriately or performance objectives were not understood accurately in the analysis phase, the associated cost of rework could be detrimental. Allen Interactions, a leading instructional design company in the United States lists seven limitations of ADDIE:

- Linear ID processes require comprehensive up-front analysis that often leads to analysis paralysis before something "working" is implemented. Teams fail to access critical design elements, teaching and learning expectations of various user personas which can be validated and improved via iterative user feedback.
- ID projects would hardly fail if, at the onset, user requirements were properly established, sufficient resources and political support was available, and users wouldn't change their minds. Unfortunately, we do not live in an ideal world and face some hard realities such as insufficient resources, plummeting budgets, lack of appropriate political support and shifting targets that force use to constantly realign our work.
- Storyboards are bit ineffective for creating, communicating and evaluating design options. Hence, bad designs aren't recognized until too late in the implementation.
- Although having a defined ID framework and detailed process for project implementation has its merits, sometimes following them too closely resists creativity and innovation.
- Linear ID model typically expects all user input to come early in the Analysis and Design phase. There is little accommodation for dealing with errors or improvements based on user feedback throughout the development process which is critical for success. Also, the cost of change due to missed requirements coming later in the development cycle could jeopardize the project potentially.

- Learning applications are designed to meet measurable criteria such as scope, schedule and budget and usually fail to meet underpinning pedagogical, behavioral requirements.
- Evaluation criteria such as post-tests provide little useful information to assess impact of learning and/or assist in improving instruction.

Agile Project Management with Scrum

Agile is a lean approach to project management that promotes adaptive planning, progressive development and delivery, a time-boxed iterative approach, and encourages swift and flexible response to change. Agile methodologies such as Scrum feature self-organized teams that are empowered to achieve specific business objectives. Agile methodologies focus on rapid and frequent delivery of partial, working solutions that can be evaluated by the users and their feedback can be used to determine next steps. Using the agile approach, solutions are built in an iterative and incremental manner via a Plan –Do – Check – Act cycle.

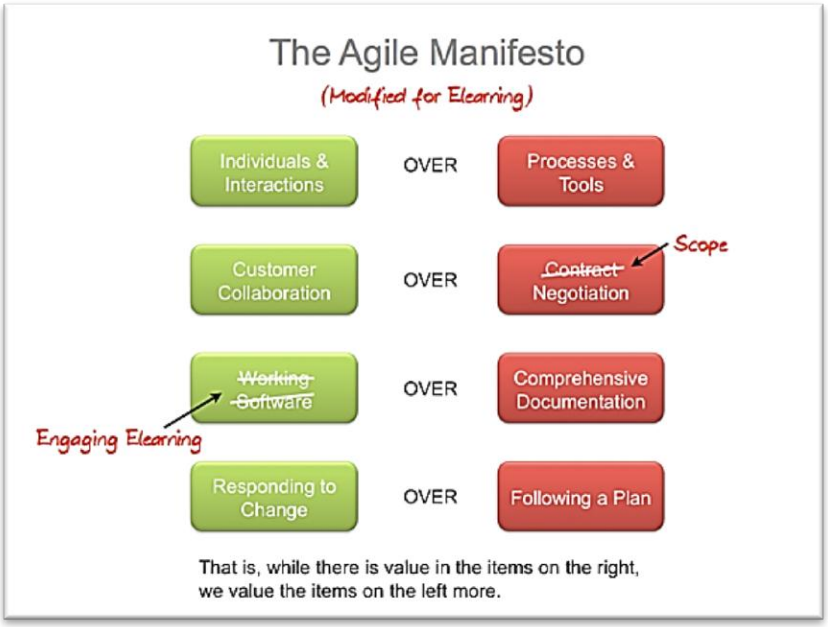

Figure 3: Moghe, Sumeet (2009) *The Agile Manifesto*, [illustration]. Retrieved June 17, 2012 from The Learning Generalist website: http://www.learninggeneralist.com/2009/06/agile-elearning-design-manual-agile-re.html)

The Agile Manifesto describes four guiding principles that have proven to be valuable for managing complex software projects and can readily be applied to ISD as seen in Figure 3. Scrum is a proven, widely used agile software development approach that can be readily applied to ISD projects. Instead of a complete methodology that defines detailed processes for project management, Scrum is a framework for software development that provides flexibility for project teams to deal with problems in a way they think best.

Scrum defines three main roles in a project team – Product Owner (responsible for maintaining prioritized list of functionality that needs to be added to the product), Scrum Master (team facilitator who shelters the team from outside distractions so that the team can focus on their work) and the project team itself. The primary artifact of a Scrum project is the product itself in a potentially shippable (production) stage. The Product Owner maintains a Product Backlog which contains prioritized list of functionality (requirements/ issues) that need to be added to the product in the form of SMART (specific, measurable, achievable, realistic and time bound) user stories. User stories (features/issues) from Product Backlog can be pulled into a Release Backlog and Sprint Backlog which includes amount of work to be completed in a specific time-boxed iteration. Sprints typically run for 2-4 weeks and the integrated work from few sprints can be delivered in a Release. At the beginning of each Release and Sprint cycle there is a planning meeting where Product Owner and the project team pull highest priority items from the Product Backlog into a Release and Sprint Backlog respectively. At the end of a Release and Sprint cycle, there is a review meeting where the project team demonstrates the completed functionality to the Product Owner and other invited stakeholders and get their feedback on work accomplished, identify issues, risks, revised priorities and re-align as necessary.

Why Agile works

According to the 2011 CHAOS Manifesto from the Standish Group, agile projects are successful three times more often than non-agile projects. The report goes so far as to say, "The agile process is the universal remedy for software development project failure. Software applications developed through the agile process have three times the success rate of the traditional waterfall method and a much lower percentage of time and cost overruns." Reflecting on the constantly changing training requirements and shorter delivery schedules at IBM, their training team figured that even though there was no reason to reject ADDIE and any of its benefits, the 'classic' approach is no longer enough. Understanding the need to be more lean and flexible, they used agile project management approach to successfully deliver training to their developers and engineers around the world (Groves et al., 2012).

One of the biggest reasons why agile methodologies are so successful is because they significantly reduce the feedback cycle from the inception of an idea to its realization and therefor the cost of change is significantly less as compared to traditional Waterfall methodology. Figure 4 and 5 show comparison of cost of change curve between a traditional and agile software development process. The traditional cost of change curve shows that the longer it takes you to find an issue (or misalignment with client expectations) within the development process, on average the more expensive it is to address and fix it because the resulting impact could significantly affect the scope, time and cost of the project.

In an agile process, the average cost of implementing a change doesn't change as much because the issues can be identified and resolved early on through collaboration and feedback during the iteration cycles and the future work can seamlessly adapt to the changes. As

organizations are adjusting to today's volatile economy and changing market conditions, there is need to train and re-train their employees on various products and services. Could there be a better evidence of training success than a 'working' learning intervention that objectively addresses a performance objective?

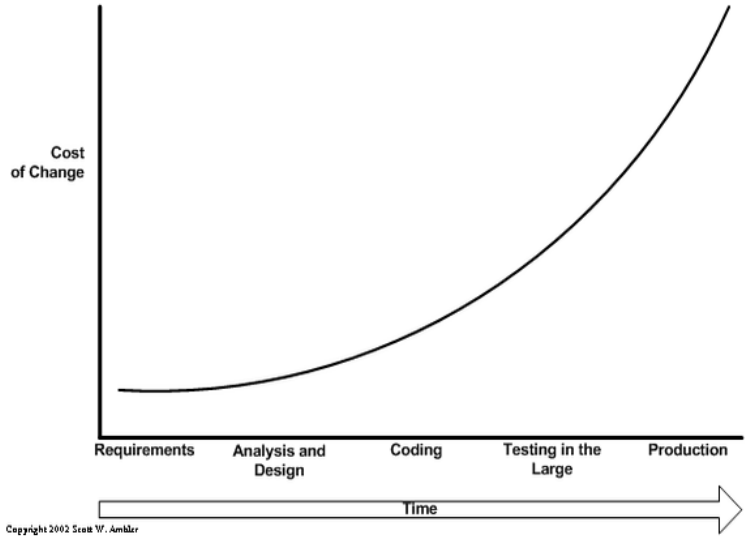

Figure 4: Traditional cost of change curve. Retrieved July 27, 2012 from http://www.agilemodeling.com/essays/costOfChange.htm

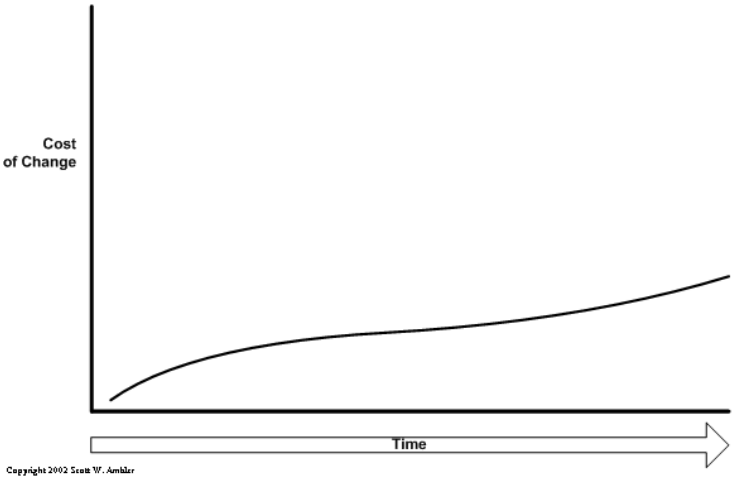

Figure 5: A realistic cost of change curve for agile software development. Retrieved July 27, 2012 from http://www.agilemodeling.com/essays/costOfChange.htm

Adaptive-ADDIE: Scrum framework for Instructional Systems Design

Most of the ISD models developed in the last decade are spin offs from ADDIE blended with Agile practices. Rapid Prototyping model (Tripp & Bichelmeyer, 1990) for instance suggests construction of prototype(s) to allow short feedback loops and continual improvement during the development process through team collaboration (Agile principle 1 & 3). Although having a defined structure and standardized ISD model generally leads to efficiency, instructional designers need to be careful that the processes are "lean" and can be adapted based on design complexity and customer feedback. Applying some of the good practices learnt from hugely successful Agile/Scrum software development framework to the proven ADDIE model gives birth to Adaptive-ADDIE, a Scrum framework for instructional system design.

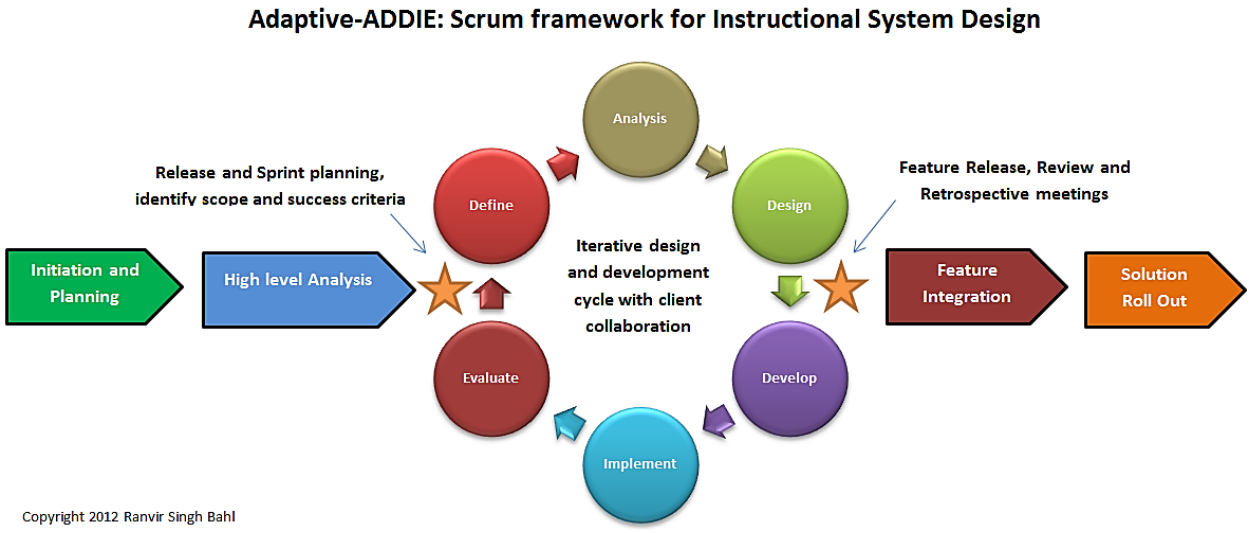

Figure 6: Adaptive-ADDIE framework for Instructional System Design

Adaptive-ADDIE framework suggests a hybrid project management approach where some phases at the beginning and end of a project are linear and the project execution (design

and development) are iterative in nature. There are five phases in this framework – initiation and planning, high level analysis, iterative design and development, feature integration and roll out.

- Initiation and Planning: The first phase is linear and includes project definition, identification of business and pedagogical need, specific goals and objectives for the project, stakeholder identification, high level assumptions and constraints, high level budget and timelines, risks, dependencies and acceptance criteria. This phase includes high level planning for overall project depending on level of clarity available. There will be iterative planning and review sessions at the beginning and end of each execution cycle. The Product Owner, Scrum Master and Scrum team are defined at this stage.
- High Level Analysis: The second phase is linear and includes high level analysis to understand the over-arching business, functional and non-functional needs of the project in order to prepare a high level project plan, schedule and budget; initial course outline and high level user stories (epics). The Product Owner is a client representative to the ISD (Scrum) team who collaborates with business and academic leaders, SMEs and executive stakeholders to get their approval for starting design and development work.
- Iterative Design & Development (Execution): The Project execution happens in phase three which is iterative in nature. The Product Owner (PO) in collaboration with business and academic stakeholders maintains a prioritized product backlog containing refined user stories. In the Release and Sprint planning meetings, the project team selects the amount of work they believe they can do in each Sprint (2-4 week iteration cycle in which ‘working’ functionality is developed) and how many Sprints will be required to properly deliver or Release functionality. Each functionality typically goes through iterative DADDIE (define, analysis, design, develop, implement, evaluate) cycles with

close collaboration and feedback from PO at each stage. Inspections in the form of reviews at the end of each Sprint ensure continuous improvement and realignment with business and academic objectives. Release reviews are an opportunity for ‘show-n-tell’ where the functionality is reviewed by a larger audience and their feedback is incorporated. Retrospectives are an opportunity for the Scrum team to reflect on the work done in previous Sprint and have a lessons learnt session.

- Feature Integration: In this phase, the functionality released in various Release cycles is integrated to form an end to end solution. The courseware is deployed on the test system running the selected Learning Management System (LMS) or website for Quality Assurance and user acceptance testing (UAT). Although individual functionality is demonstrated and tested at the end of each Release, the integrated UAT testing is an opportunity to do regression and performance testing before the solution is deployed in production. In addition, the instructors, students and staff members can pilot and have hands on experience on the system, receive appropriate training and provide feedback to fix any loose ends and ensure the integrated system works smoothly.
- Solution Roll Out: Once the solution is properly tested and end user feedback has been incorporated, the solution is ready for deployment on the production system and Go Live. As each project brings about changes to the status quo, this phase typically includes communication and change management to ensure wider adoption. The end of this phase marks the ‘closing’ of the project and transition into operations. Finally, it might include an evaluation survey against the project success criteria and a formal acceptance sign-off from the sponsors.

Conclusion

Adaptive-ADDIE promotes frequent "inspect and adapt" philosophy in which the product development happens in multiple iterations or sprints based on close collaboration and feedback from the client or Product Owner. Similar to Scrum, it works on empirical process control mechanism that provides transparency in the design and development process and exposes any risks, inefficiencies or potential misalignment with the customer's and/or end user's requirements thereby improving success rate. Adaptive-ADDIE if implemented correctly forces realistic expectations, embraces change and allows team to deliver value at consistent intervals without doing extensive planning, getting into analysis paralysis and developing mounds of project documentation.

References

Cohen, Mike (2012). Agile succeeds three times more than waterfall. Retrieved from http://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall

Doherty, I. (2010). Agile project management for e-learning developments. *Journal of Distance Education (Online*), *24*(1), 91-106. Retrieved from http://ezproxy.library.ubc.ca/login?url=http://search.proquest.com/docview/867666594?accountid=14656

Nelson, W. A, Macliaro, S. & Sherman, T.M. (1988). The intellectual content of instructional design. *Journal of Instructional Development, 2*(1), 29-35

Groves, A., Rickelman, C., Cassarino, C., & Hall, M. J. (2012). Are you ready for agile learning design? T + D, 66(3), 46-51,6. Retrieved from http://ezproxy.library.ubc.ca/login?url=http://search.proquest.com/docview/1015775738?accountid=14656

Tripp, S., Bichelmeyer, B. (1990). Rapid prototyping: An alternative instructional design strategy. *Educational Technology Research and Development, 38*(1), 31-44

Wirfs-Brock, R.J. (2009). Designing with an Agile Attitude. *Software, IEEE , 26*(2), pp.68-69