**COIS-ADMN 3850H – 2018 FA – QUESTION SET #6**

Submitted by: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Konrad Bartlett\_\_\_\_\_\_\_\_\_\_\_ # CORRECT = \_\_\_\_\_\_\_\_

Evaluated by: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # **FOR REVIEW** = \_\_\_\_\_\_\_\_

**Instructions:**

1. first, convert this this document to a .pdf
2. then, type your answers clearly and concisely on these pages to the questions that follow.
3. **MAKE SURE YOU INCLUDE YOUR PAGE NUMBER REFERENCE FROM THE TEXT IN THE SPACE PROVIDED.**
4. submit these completed pages at the beginning of the next class for evaluation.

**Text: MANAGING AGILE PROJECTS** | Edited by Kevin Aguanno

**Part Two: Managing Agile Projects is Different [pages 67-164]**

# PAGE #

77 -82 1) Compare the "waterfall", "spiral", "iterative", "incremental" and "Agile" development processes.

Waterfall

* Sequential, phase-based approach
* Analyze problem at beginning
* Create plan for project at beginning
* Follow that plan through the entire project, with very little change to plan
* Fast, simple, easy to understand

Spiral

* A phased, iterative approach
* Perform analysis until ready
* Design product, if analysis is wrong analyze again
* Sequentially go through phases, but if you find problems with current phase, go back to the previous phase and start again
* Kind of fast, pretty complicated, makes for good progress

Iterative

* Assume you make mistakes, so develop deliverable multiple times
* First iteration do lots of analysis and little design, collect feedback
* On subsequent iterations take feedback and do more analysis (less than previous iteration), and more design.
* The benefit of redesigning the product is that you will end up with a deliverable that is closer to the clients intention
* Better designed product
* Very slow, not very complicated, best quality product

Incremental

* Go through multiple iterations of analysis and design
* Until mostly complete
* Do iterative multiple times
* Once iterative is done once, deploy
* Get feedback from users and make another version
* Keep doing this until satisfied
* As fast/slow as iterative but requires upkeep once deployed
* Pretty complicated, but even better quality product than iterative

Agile

* Completely based on collecting feedback and just going back to the drawing board and reworking things.
* Deliver software in small increments and deliver functionality in order.
* Work through increments, each increment the same size.
* Work through an incremental process however at any point you can go back and redo something if agreed to
* Daily meetings to go over things with the team, and see progress
* Could be faster could be slower, depends on how many times you need to rework requirements. Needs to have very flexible time contraints
* Not complicated, it’s like just do something, if you don’t like it oh well! Do it again but better
* Really produces some fine products

\_\_84\_ 2) List the benefits that can be gained from working incrementally:

* Easy delivery of useful software
* Delivery of functionality based on the value users give to that functionality (business value-driven instead of architecture or risk driven)
* Clear, tangible feedback on progress: each increment delivers finished, production-quality functionality.
* Clear and regular feedback on the quality and fit of the software from the users to the development team.
* The ability to adapt the project plan.
* Clear scheduling and predictable delivery.
* The development of the simplest possible system that satisfies the requirements, without any unnecessary adornments and complexity.

\_\_85\_ 3) What are the dangers of this approach?

* By only looking at small pieces (increments) of the software, we may have to do massive reword, which might have been avoided if we analyzed or designed the whole system correctly.
* We might “paint ourselves into a corner;” that is, we can get a system that is incapable of supporting some new requirement.
* We may be unable to add global requirements, such as alternate language support, performance, and security. A typical example is trying to retrofit “security” to an application that was not designed with this requirement in mind.
* We might g slower because we have to restart analysis and design sessions for each increment, instead of doing it once at the start of the project.

\_\_90\_ 4) What are the 10 principles [detailed on page 90] that have shown themselves to be useful in setting up and running projects?

1. Different projects need different methodology trade-offs.
2. A little methodology does a lot of good; after that, weight is costly.
3. Larger teams need more communication elements.
4. Projects dealing with greater potential damage need more validation elements.
5. Formality, process, and documentation are not substitutes for disciplines, skill, and understanding.
6. Interactive, face-to-face communication is the cheapest and fastest channel for exchanging information.
7. Increased communication and feedback reduces the need for intermediate work products.
8. Concurrent and serial development exchange development costs for speed and flexibility.
9. Efficiency is expendable in non-bottleneck activities.
10. Sweet spots speed development.

\_\_152\_ 5) What does it mean to be an Active Participant?

An active participant is an on-site representative or customer who have the authority and ability to provide information pertaining to the system being built, to make pertinent and timely decisions regarding the requirements, and to prioritize them. So basically, it’s like being somebody on site who is a representative of the end user or client who can be asked or referenced for next steps. They’re like stakeholders that just always hangout around the project.