Sprint, then Fly: Teaching Agile Methodologies with Paper Airplanes

Mark Frydenberg,
mfrydenberg@bentley.edu

David J. Yates
dyates@bentley.edu

Computer Information Systems Department
Bentley University
Waltham, Massachusetts

Julie S. Kukesh
julie.kukesh@mendix.com
Mendix Corporation
Boston, Massachusetts

Abstract

As industry embraces the agile methodology for application development, universities are shifting their curricula to teach agile principles along with traditional waterfall concepts. This paper describes a simulation game offered to students in a first-year computing concepts course to introduce both models of application development. Students work in development teams to design, build, and test paper airplanes following both waterfall and agile principles to experience the roles, processes, and challenges of each. Participants track their team's progress throughout the activity, so they can draw conclusions about the benefits and challenges of each approach. Survey results indicate that students learned the various roles and approaches of both methods through this experience.

Keywords: agile, waterfall, simulation games, app development

1. INTRODUCTION

The widespread implementation and acceptance of agile methodologies in industry during recent years has caused universities to reexamine approaches for teaching software development concepts across the curriculum. The agile methodology (Cunningham, Principles behind the Agile Manifesto, 2001) is a set of principles that define the people, process and work-output for the development and delivery of software and applications. In contrast to a traditional waterfall development methodology, which relies on gathering user requirements, developing, testing, and deploying software sequentially, agile makes use of cross-functional teams that collaborate closely with business stakeholders, flexible and progressive requirements gathering, and early delivery of working product followed by rapid iteration of these tasks.

Both agile and waterfall methodologies try to bring the right resources to the project at each phase of development. Waterfall is organizationally agnostic but siloed with respect to how to accomplish this, while agile leverages collaborative, cross-functional teams in every phase of development. The Agile Manifesto (Cunningham, Manifesto for Agile Software

Development, 2001) and guiding principles (Cunningham, Principles behind the Agile Manifesto, 2001) shown in Appendix 1 describe the tenets of the agile development process.

This paper presents an interactive game in which students build paper airplanes to simulate waterfall and agile development processes. Students learn about waterfall and agile models for software development, team member roles, and development processes that guide each methodology. The authors facilitated this exercise with students enrolled in an introductory technology course at a business university.

Organization

The paper presents a high-level overview of waterfall and agile methodologies, and summarizes relevant literature regarding the teaching of these in a variety of higher educational contexts. A discussion of simulation games used to teach agile concepts follows. The paper then describes a simulation game using paper airplanes to introduce agile concepts, and summarizes the results of this activity. The paper concludes with reflections from students and observations from the activity's facilitators.

Research Questions

The following questions guided the development of the simulation game described in this paper, and the study of its implementation:

- After participating in the simulation, will students be able to distinguish between waterfall and agile development concepts and processes?
- After completing the simulation, will students be able to describe the different team member roles, development approaches of waterfall and agile methodologies, and their benefits and limitations?
- After reflecting on the simulation, will students be able to identify how they might apply agile concepts beyond software development, into the activities of their own daily lives?

2. WATERFALL AND AGILE OVERVIEW

Figure 1 illustrates the differing approaches of waterfall and agile methodologies. Waterfall follows a sequential process, whereas the agile development process relies on frequent meetings with all participants who set their immediate goals and identify obstacles toward making progress.

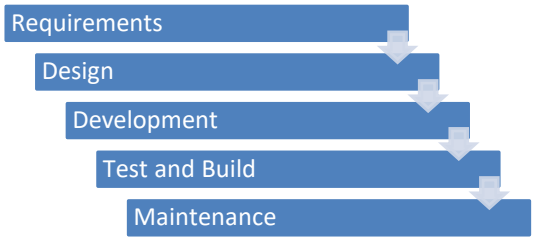

Figure 1 (a). The waterfall model is a sequential design process.

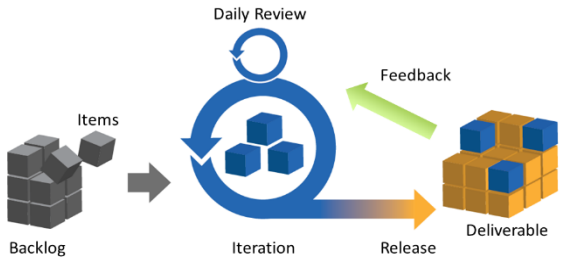

Figure 1 (b). The agile model is an iterative design process. (Fongamanda, 2012).

Both waterfall and agile methodologies place participants in roles that contribute toward the design and development of a completed product. Tables 1 and 2 present the roles implemented in many waterfall and agile development activities. The authors have simplified the names and descriptions for the sake of introductory students; agile teams in industry may use slightly different names or descriptions. In a waterfall model, participants take on the following roles, shown in Table 1.

Role	Description
Analyst	Business and systems analysts ensure that the product fulfills customer requirements and is delivered efficiently and effectively.
Developer	Responsible for design and development of product components required in the current release.
Tester	Verifies that the product is fit for customer use and validates that the product works as designed.

Table 1. Waterfall development team member roles and descriptions.

In an agile model, team members take on the following roles shown in Table 2.

Role	Description
Business Analyst	Responsible for articulating customer requirements - focusing on delivering value early and often - as well as release planning for the product.
Product Owner	Reconciles and harmonizes stakeholder requirements to help develop user stories (see Table 3) and assures the quality of the product.
Scrum Master	Owns the agile process for their team. Responsible for delivering user stories for the current iteration and coaches the team in doing so.
Developer	Merges the developer and tester roles from waterfall. Focuses on delivering components of the product needed in the current iteration.

Table 2. Agile roles and descriptions.

In addition, understanding the following vocabulary and concepts are key to participating in an agile development exercise:

Concept	Definition
User Stories	Express the "why, what and who" of desired business value for many types of product backlog items, especially features.
Product Backlog	Prioritized list of the items - features and other capabilities - needed to develop a successful product.
Standup	Short daily meeting in which team members describe what they did the day before, what they will do that day, and where they need help.
Sprint	Current iteration of product delivery, which has four activities: planning, execution, review and retrospective.

Table 3. Agile vocabulary and concepts. (Rubin, 2012).

3. AGILE ADOPTION IN THE ENTERPRISE

Implementing agile in different domains (including IT) requires a customer to whom an agile development team is going to deliver a product. Agile's methodology guides the team's activities to maximize the product's value delivered to the customer. Thus, delivering value to the customer, however defined, becomes the central mission of the team.

Agile teams are organized to provide maximum value to the business in as little time as possible. Teams rely on teamwork, focus, and collaboration. Specifically, having an agile team continuously focus on and produce value for a customer is an important measure, but not the only measure, of the team's effectiveness. It is also important to consider efficiency for a variety of reasons, not the least of which is that any team in any context has finite resources to devote to a project. In higher education, agile projects often span a semester or a quarter and deliver products usually developed by fixed-sized teams.

Companies and government agencies are adopting agile development strategies because of their focus on delivering value and their catalytic effect on innovation. According to Rigby et al. (Rigby, Sutherland, & Takeuchi, 2016, para. 7) "these days most companies operate in highly dynamic environments. They need not just new products and services but also innovation in functional processes, particularly given the rapid spread of new software tools. Companies that create an environment in which agile flourishes find that teams can churn out innovations faster in both those categories." However, these authors caution "Agile is not a panacea. It is most effective and easiest to implement under conditions commonly found in software innovation: The problem to be solved is complex; solutions are initially unknown, and product requirements will most likely change; the work can be modularized; close collaboration with end users (and rapid feedback from them) is feasible; and creative teams will typically outperform command-and-control groups." (Rigby, Sutherland, & Takeuchi, 2016, para. 12)

Despite its widespread use and professed simplicity, most organizations experience challenges when adopting agile development practices. The most recent *State of Agile* survey (VersionOne, 2017, p. 2) found that "While 94% of respondents said their organizations practiced agile, they also stated that more than half (60%) of their organizations' teams are not yet practicing agile. Similarly, although 44% of respondents stated that they were extremely knowledgeable regarding agile development practices, 80% said their organization was at or below a 'still maturing' level."

As agile adoption has increased in the enterprise (Dingsøyr, Nerur, Balijepally, & Moe, 2012), university curricula are evolving (Babb, Hoda, & Nørbjerg, 2014; Lang, 2017) to meet this market shift. In order for the next generation of technology professionals to join these organizations and contribute in a meaningful way, universities need to keep pace by teaching agile

methodology to both introductory and advanced students as part of their technology curriculum.

4. TEACHING AGILE THROUGH SIMULATION GAMES

Agile methodologies have many implications in educational settings. Even though agile methodologies are most often associated with software development, many educators have taught agile concepts using scenarios or problems from a variety of domains through simulation games.

Simulation games are a common pedagogy to introduce the roles and concepts of agile methodologies in both industry and academic settings. Wangenheim, Savi, & Borgatto (2013) describe a pencil and paper game to introduce agile concepts in undergraduate courses. Players take on roles of team members, Scrum master, and product owner, as they try to create objects from paper sheets during several sprints. (Paasivaara, Heikkila, Lassenius, & Toivola, 2014) and (Krivitsky, 2017) present Scrum simulations to introduce the events and concepts of agile development by simulating sprints while building models using LEGO™ Building blocks. (Beale, 2016) and (Fernandes & Sousa, 2010) developed a board and card games to introduce scrum-based concepts by having students describe solutions to complete a task and allocate resources available to do so over several rounds. In each scenario, students need to "follow the best practices of software engineering in order to avoid any obstacle[s]" (Fernandes & Sousa, 2010, p. 53) and reach a successful outcome. The game "makes players come face-to-face in the same space to make decisions about the game, acting as an antecedent to the role of the scrum meeting in agile project management. In this way, the scrum will be easier to identify for the players/students, and the instructor can use these moments to discuss how making decisions in the game mirrors the way that scrums work organizationally." (Beale, 2016, p. 4)

Paper Airplane Simulation

This section describes the learning objectives, delivery, and results of a simulation designed to present waterfall and agile methodology and concepts to first year students enrolled in IT 101, an introductory technology and computing concepts course at a business university. The authors adapted an activity developed by ScrumInc., used as part of certified Scrum Master class training, for this classroom exercise. The goal: "Plan, build, and test as many paper airplanes as you can in 3 minutes" (Hegarty, 2013, para. 1) following the methodologies of waterfall and agile development.

Students enrolled in five different sections of IT 101 with two different instructors, participated in the simulation, which took place in an 80-minute classroom session. The same facilitators, a university alumnus and his colleague who are now working in industry at Mendix, a company whose rapid application development platform facilitates agile methodologies through the entire application lifecycle, presented the exercise to all sections on two different days, as the instructors monitored student behavior and provided encouragement.

Learning Objectives

The instructor and industry facilitators set out to create a simulation game for introductory technology students (who have little or no prior software development experience) with the following learning objectives. After completing this simulation game, students will be able to:

- Compare processes of waterfall and agile development;
- Describe tasks, concepts, and roles in traditional and agile software development processes;
- Formulate product requirements in the form of user stories that include roles, actions, outcomes, and their motivation;
- Perform roles of agile and waterfall development team members to experience benefits of each method; and
- Apply agile development principles to personal, school, and work tasks.

Presenting the Paper Airplane Activity

The facilitators began the lesson by asking students, "What would you need to do if you were going to build an app?" Students responded, "have an idea," "find someone to program it," "test it," and "publish it." Students said these tasks should happen one after the other, in this order. The facilitators pointed out the connections between the students' responses, and the waterfall methodology shown in Figure 1(a). Next, the facilitators explained to the students they were going to experience the waterfall methodology by building paper airplanes instead of developing apps.

The facilitators first briefed the students on three key components of the exercise: the objective, the roles, and the restrictions.

Objective. The objective of the waterfall model simulation is to build as many paper airplanes as

possible in 12 minutes that will fly successfully over a specified distance of 15 feet when tested. The simulation evaluates a team's performance by measuring both volume (number of planes built) and quality (percentage of planes that flew the required distance). In practice, these metrics reflect the value (efficiency and quality) delivered to a customer in a development project.

Roles. Students self-organized into groups or "development teams" of three or four, and within their groups, identified the role that each student would play. One student took on the role of analyst, who would write the specifications for construction of the paper airplanes; one student took on the role of tester, who would test the paper airplanes constructed; and the remaining students acted as the development team responsible for building the paper airplanes.

This simulation exercise simplifies the various roles that are part of waterfall and agile development projects. In practice, analysts are part of a waterfall team; analysts give voice to the requirements on an agile team. A business analyst, however, generally is not a core member of an agile team.

Restrictions. The students had to use 8.5" x 11" sheets of white paper to build their airplanes. They had to test their planes on a "runway" in the center of the classroom, shown in Figure 2, or in the hallway (if the classroom layout was not conducive). The facilitators pre-measured the 15-foot distance prior to class and marked the boundaries with masking tape.

Waterfall Simulation

In the first round, which mimics a waterfall process, the facilitators gave exactly 12 minutes for students to spend 3 minutes for planning and design, 6 minutes building, and 3 minutes testing their airplanes. Students were restricted to performing work based on their assigned roles during their assigned time, and could not perform work reserved for other roles.

Each analyst wrote the requirements and design specifications for the team's airplanes during the planning and design phase and could not communicate with other team-members, nor could the analyst assist in building a prototype. The development team could begin work only during the build phase and was restricted from asking the analyst any questions or clarification on the design specifications provided to them, as well as from the results of testing any of the airplanes built. Each tester could not begin testing until the test phase, and could not make any modifications to the airplanes the development team built. The facilitators tracked time for each phase and recorded for each team the number of paper airplanes built during the build phase and the number that flew 15 feet successfully. Figure 2 shows students involved in the testing phase of the activity.

Figure 2. Students test their planes to see if they will fly a distance of 15 feet.

To end the exercise simulating a waterfall approach, the facilitators debriefed with the students about what worked and what did not work well. The debrief discussion ensured that students met the learning objectives of the traditional waterfall development approach by being able to observe and discuss the key challenges organizations experience when it comes to delivering custom applications.

Through their participation in this simulation, students recognized several application development challenges organizations may experience in this traditional waterfall approach:

- Defining requirements up front is very difficult because business requirements change;
- A lack of communication between developers and the customer may have unfavorable influence on the final product; and
- Delivering the application often takes a long time, and often the resulting product does not meet a business' expectations in the end.

Introducing Agile

Following the waterfall simulation described above, the facilitators told the students they were going to experience the agile methodology by repeating the paper airplane exercise with a different set of roles and rules that mimic the agile development process.

In the agile simulation activity, students had the same amount of time as before, 12 minutes, but split into three sprints: 1 minute for planning and design; 2 minutes to build; and 1 minute to test their airplanes. At the end of each sprint, they could go back, share their results with the team, whose members would offer the builders

suggestions for improving the design prior to the next testing phase. Throughout each sprint, the facilitators recorded the number of airplanes built and the number successfully flown for each group. At the end of the three sprints, the facilitators engaged students in a similar debrief discussion where they reflected on the results of the agile version of the activity, compared and contrasted how their experience differed between the traditional waterfall methodology and agile, and compared the results of agile to those from the first activity using waterfall.

The facilitators tracked the number of planes built and flown successfully using waterfall and agile approaches. The charts in Appendix 2 Figures 1-5 and data in Appendix 2, Figure 6 summarize the results for 31 teams across five sections of IT 101. Appendix 2, Figures 1 and 2 show the number of planes built and flown successfully using waterfall and agile methodologies. The majority of teams experienced higher success rates using agile over waterfall methodologies. Appendix 2, Figure 3 shows the number of airplanes built during each of the three sprints in the agile development round of the game. For most teams, the number of planes built in successive sprints was the same or more than the number built in earlier sprints. This suggests that the collaborative process in agile had a positive impact on each team's results. Appendix 2, Figure 4 shows the number of airplanes each team built using waterfall and agile methods. For almost all of the 31 teams, the total number of airplanes built using agile methods was higher than the number built following waterfall methods.

Appendix 2, Figure 5 shows the average success rates (number of planes flown successfully divided by the number of planes built) for each team using waterfall and agile methods. Of the 31 teams, three had perfect success rates using both waterfall and agile methods; six had perfect success rates with waterfall only, and seven teams had perfect success rates using agile methods. For only five of the 31 teams did agile result in a lower success rate than waterfall. This exercise gave the students the sense that agile's iterative design process involves more people and generally produces superior results (Babb, Hoda, & Nørbjerg, 2014).

5. STUDENT REFLECTIONS

This section presents the results of a written assignment completed within three days of the simulation game, in which 76 students (42 male, 34 female) reflected on what they learned. Before the original activity, most students were not familiar with agile or waterfall methodologies as shown in Figure 3.

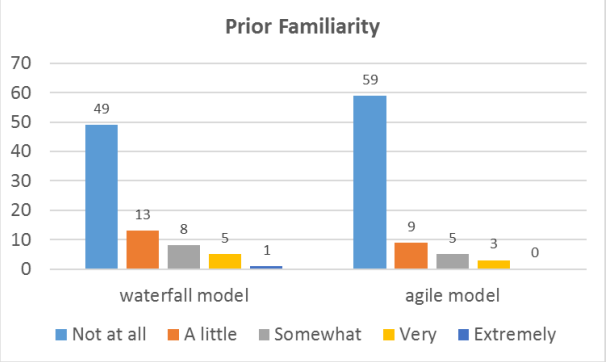

Figure 3. Prior familiarity with development methodologies.

The instructors provided three different ways for students to learn about software methodologies: reading an article introducing agile development terminology (Field, 2014) before class, attending an in-class presentation, and participating in the paper-airplane simulation activity. Figure 4 summarizes the extent to which students felt each of these contributed to their learning.

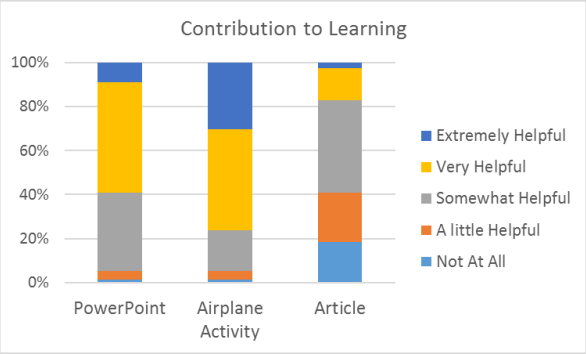

Figure 4. Contribution to learning.

Forty-five of the students thought the PowerPoint presentation was very or extremely helpful; 58 said the Airplane activity was very or extremely helpful; only 13 said the assigned article was very or extremely helpful in their understanding of agile and waterfall concepts.

When asked in a fill-in-the-blank question to identify which role, as presented in Tables 1 and 2, that they took on in each part of the simulation, most students remembered the tasks they completed, although some identified their roles with a different name, as shown in Table 4.

Role	Waterfall	Agile
Developer	24	25
Tester	20	14
Product Owner		7
Scrum Master		7
Analyst	16	6
Builder	7	5
Designer	1	1
Director		1
Don't Remember	4	5
Other	2	3
Not Present	1	1
Total	**75**	**75**

Table 4. Student roles in waterfall and agile simulations.

Developers and testers were much more likely to recall their roles correctly. Some students were confused about what to call the third role in agile: responses include builder, designer, director, product owner, Scrum master. Some responsibilities needed some clarification as well; for example, a developer develops in waterfall, but develops and tests in agile. Some students were confused that the label has a different meaning in different contexts. This may have to do with the design and presentation of the exercise.

Waterfall Reflections

During the waterfall activity, builders recalled their role as follows: "I constructed the paper airplanes as efficiently as possible;" and "I tried to build a paper airplane according to the specs that the analyst gave me." Testers said, "I had to test the product to see which ones flew 12 feet, and which ones didn't;" and "My job was to throw the airplanes at the best distance possible."

Students tried to adhere to the boundaries of their roles. Said one student in the analyst role: "I was not allowed to talk about to plan or touch the airplanes while they were being developed." A developer said, "I was not really able to change the design of the planes, or know what worked or what didn't." Another said, "It was hard to follow the directions of the analyst and make paper airplanes fast and according to the specs." "We could not communicate with the other members of our team so it was hard to build a successful airplane."

When asked about the benefits of the waterfall method, students commented, "Every member of the team had a specific focus which made the process very specialized; and "We were acting as an assembly line as the paper plane project moved through the required steps." They also recognized issues with the waterfall development method as applied to this simulation. Remarks included "It was a waste of time because everybody had to wait for the previous person to finish working;" "We were very siloed and didn't collaborate at all, which was frustrating;" and "There was attention to detail, but as the tester, I tested a finished product I had never seen before."

Agile Reflections

During the agile activity, students commented on their roles. Analysts recalled, "I had to find a model for the paper plane that was going to be built. I did these three different times after we made adjustments to the product each time to improve mistakes;" and "I was responsible for working with the other member to design a model for the airplane. Again, I drew up a more comprehensive diagram for the dev team to follow."

The activity made students aware of how the different development methods influenced their group's collaborative process and results. Comments included "The shorter times allowed us to hear from our analyst and scrum leader more so I felt like I knew what I was doing more;" "I was not able to help create the actual planes because I was the analyst and not part of the development team;" and "I did not exactly get to write up the instructions or create the airplane but was allowed to add input and collaborate with group members, so I was able to contribute much more during this process."

Regarding the benefits of the agile simulation, students commented, "We learned how to work collaboratively in a group;" "I like it better because it allowed us to see what parts we were doing wrong and enabled us to improve upon our prior attempts much easier;" "Speeding up the process let us address issues in our models quickly and efficiently;" "This was a much easier method. We were able to communicate and therefore we could tweak the plan from round to round and perfect the airplane. It helped us have a better success rate and also waste less time."

Noting difficulties of the agile process, one group stated, "Because we were more pressed for time, our quality decreased substantially. However, we tripled our production numbers;" and "I could not interfere with the analyst while she was writing the instructions even though I knew the plane she was going for was not going to work."

To answer the third research question, the facilitators asked students to consider how they might apply agile principles to other areas of their personal, school, or work lives. They commented: "When working at my internship this will come in handy and also when doing homework. Do short bursts of work and then reward yourself with a little relaxing." Another said, "I could use it to improve my studying skills. I could study for a short sprint, and test what I know. After I realize what I missed, that's what I'll focus on for the next sprint. I can continue that for as long as it takes until I know the material."

Several students noted that agile methods are a good model for improved collaboration and group work. "Increasing communication at multiple times in a products production cycle will improve the final product. This is also true in the case of a paper written by a team of five. Less iteration is necessary in a personal environment, as there is only one team member." Students reflected on the global lessons they learned from acting agile: "It is important to do your best the first time with anything, especially knowing that you always can't go back to fix it." Another student noted, "Spending a shorter amount of time planning gives you multiple opportunities to see which models work best. I can apply this to planning and creating school projects." Another student expanded, "For life in general, it is important to break larger projects into smaller groups and discuss with people your progress on your way to your long-term goal."

Regarding the paper airplane activity, students said: "I think that the activity went very well and clearly showed the difference between the waterfall and agile method. The activity possibly could be improved by slightly increasing the number of people in each group or allowing the groups to interact more and work together more during the agile method compared to the waterfall method." Another student commented, "The activity really represented the waterfall and agile methods well. It made me understand the idea and process, [and was] good way to learn how app development companies go about creating apps and software nowadays."

6. OBSERVATIONS AND CONCLUSIONS

Students were engaged in the exercise and comprehended the basic ideas of both methodologies.

Some of the analysts used search engines during the research part to look up how to build a paper airplane in order to provide accurate instructions for their team of developers. The facilitators pointed out that using a search engine for this purpose in this context is analogous to gathering requirements accurately and documenting them prior to beginning the development process. Other analysts tried to build a prototype rather than writing the instructions and specifications.

Often, students in the analyst role conflated their roles and responsibilities with those of a developer. Many developers had a difficult time sticking to their own role and waiting their turn. Universally, the developers tried to start building before the allocated "build" time had begun and after it ended. Additionally, they were vocal with their criticism of the analyst instructions both during the activity time and during the debrief discussion.

Order of Presentation

During the first two iterations of this classroom activity, the facilitators presented the waterfall method first, followed by agile, in all sections. During the following semester, they presented waterfall first and agile second in half of the sections, and agile first/waterfall second in the other half of the sections. The goal was to see if order in which students learned about agile and waterfall methods had any impact on their results.

The facilitators found that when presenting agile first and waterfall second, the waterfall activity appeared to serve as a fourth sprint. The builders already knew how to make planes effectively based on the previous three sprints, so little room for improvement was possible. Even when the facilitators changed requirements slightly (for example, requiring the use a half-sheet of paper rather than a full sheet, or changing the flying distance), distinctions between the development phases of both models were blurred because of prior experiences.

Improvements

Based on their experience facilitating this activity, the facilitators recommend several modifications to improve it for a future offering:

- Modify the simplified agile roles as presented to reflect these roles more accurately in agile development projects in the workplace: The analyst role would collapse with the product owner role; the Scrum master would be a team lead responsible for keeping time; and

the agile development team would self-organize to test and build the airplanes.
- Conduct a sprint retrospective (Rubin, 2012) after each sprint to include process, communication, and other improvements in the next iteration.
- Reorganize the activity to present, simulate, and debrief on one methodology at a time, rather than introducing both, simulating both, and then debriefing on both after the entire exercise concludes. (Facilitators implemented this change for the last two of the five sections.)
- Follow up the simulation with a simple app development activity using a tool such as Mendix, a no-code, model-driven development tool that facilitates the process of creating mobile and web apps without prior programming knowledge. Creating a working app using a development tool will allow students to apply concepts from the simulation to a real-world development project as part of an agile team without having to already know, or learn, how to program.

Conclusions

The debrief sessions at the end of each simulation reflect an agile sprint retrospective in which participants reflected on their process, communication and also the successes, and challenges, in working together. They felt very "siloed" in the waterfall simulation and much more collaborative in the agile simulation. Future iterations of this simulation might alternate introducing agile and waterfall methods first in different sections, to minimize any perceived influence that one method is superior to the other.

This lesson on introducing agile and traditional methodologies brings awareness about the process of creating apps, the roles involved in app development beyond developers, and the various tasks that each team member must complete. Through the paper airplane development simulation described here, students gained an appreciation for current software development methodologies that they otherwise might not see until later in their studies.

7. ACKNOWLEDGEMENTS

The authors thank Conner Charlebois, alumnus and Senior Solutions Consultant at Mendix Corporation, for initiating, refining, and facilitating the paper airplane simulation activity.

8. REFERENCES

Babb, J., Hoda, R., & Nørbjerg, J. (2014). Embedding Reflection and Learning into Agile Software Development. *IEEE Software, 31(4)*, 51-57.

Beale, M. (2016). Designing an Agile Game for Technical Communication Classrooms. *SIGDOC '16: Proceedings of the 34th ACM International Conference on the Design of Communication* (pp. 17:1-17:9). Silver Spring, MD: ACM.

Cunningham, W. (2001). *Manifesto for Agile Software Development*. Retrieved from Agile Manifesto: http://agilemanifesto.org/

Cunningham, W. (2001). *Principles behind the Agile Manifesto.* Retrieved from Agile Manifesto: http://agilemanifesto.org/principles.html

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A Decade of Agile Methodologies: Towards Explaining Agile Software Development. *Journal of Systems and Software, 85(6)*, 1213–1221.

Fernandes, J., & Sousa, S. M. (2010). PlayScrum - A Card Game to Learn the Agile Method. *Second International Conference on Games and Virtual Worlds for Serious Applications* (pp. 52-59). IEEE.

Field, D. (2014, November 7). *Embrace Agile Requirements Gathering And Best Practices*. Retrieved from Mendix: https://www.mendix.com/blog/an-agile-development-approach-to-requirements-gathering/

Fongamanda. (2012, May 22). *File: Agile Project Management by Planbox.png.* Retrieved from Wikimedia Commons, the free media repository: https://commons.wikimedia.org/wiki/File:Agile_Project_Management_by_Planbox.png

Frydenberg, M., & Press, L. (2009). From Computer Literacy to Web 2.0 Literacy: Teaching and Learning Information Technology Concepts Using Web 2.0 Tools. *Infomation Systems Education Journal, 8*(10), 18.

Hegarty, C. (2013, March 4). *The Serenity of Flow*. Retrieved from ScrumInc.:

https://www.scruminc.com/the-serenity-of-flow/

Krivitsky, A. (2017). *Lego4Scrum: Scrum Simulation with LEGO.* Leanpub.

Lang, G. (2017). Agile Learning: Sprinting Through the Semester. *Information Systems Educational Journal, 15(3)*, 14-21.

Paasivaara, M., Heikkila, V., Lassenius, C., & Toivola, T. (2014). Teaching students scrum using LEGO blocks. *Companion Proceedings of the 36th International Conference on Software Engineering* (pp. 382-391). Hyderabad: ACM.

Rigby, D. K., Sutherland, J., & Takeuchi, H. (2016, May). Embracing Agile. *Harvard Business Review, 50*, 40–48.

Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process.* Boston, Ma: Addison-Wesley.

Thomas, J. D., & Blackwood, M. (2010). Computer Literacy and Non-IS majors. *Information Systems Education Journal, 8*(58), 3-12. Retrieved from http://isedj.org/8/58

VersionOne. (2017). *11th annual State of Agile Report.* Alpharetta, GA.

Wangenheim, C. G., Savi, R., & Borgatto, A. F. (2013, October). SCRUMIA - An educational game for teaching SCRUM in computing courses. *Journal of Systems and Software, 86*(10), 2675-2687.

Editor's Note:

This paper was selected for inclusion in the journal as an EDSIGCON 2017 Distinguished Paper. The acceptance rate is typically 7% for this category of paper based on blind reviews from six or more peers including three or more former best papers authors who did not submit a paper in 2017.

Appendix 1. Agile Manifesto and Guiding Principles

Agile Manifesto

The Manifesto for Agile Software Development (Cunningham, 2001) outlines principles for improving the process of software development:

"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."

Guiding Principles

The guiding principles behind the Agile Manifesto (Cunningham, Principles behind the Agile Manifesto, 2001) are:

We follow these principles:

Our highest priority is to satisfy the customer
through early and continuous delivery
of valuable software.

Welcome changing requirements,
even late in development.
Agile processes harness change
for the customer's competitive advantage.

Deliver working software frequently,
from a couple of weeks to a couple of months,
with a preference to the shorter timescale.

Business people and developers
must work together daily throughout the project.

Build projects around motivated individuals.
Give them
the environment and support they need,
and trust them to get the job done.

The most efficient and effective method
of conveying information
to and within a development
team is face-to-face conversation.

Working software
is the primary measure of progress.

Agile processes
promote sustainable development.
The sponsors, developers, and users
should be able
to maintain a constant pace indefinitely.

Continuous attention
to technical excellence
and good design enhances agility.

Simplicity--
the art of maximizing the amount
of work not done--
is essential.

The best architectures,
requirements, and designs
emerge from self-organizing teams.

At regular intervals, the team reflects
on how to become more effective,
then tunes and adjusts its behavior accordingly.

Appendix 2. Paper Airplane Activity Results

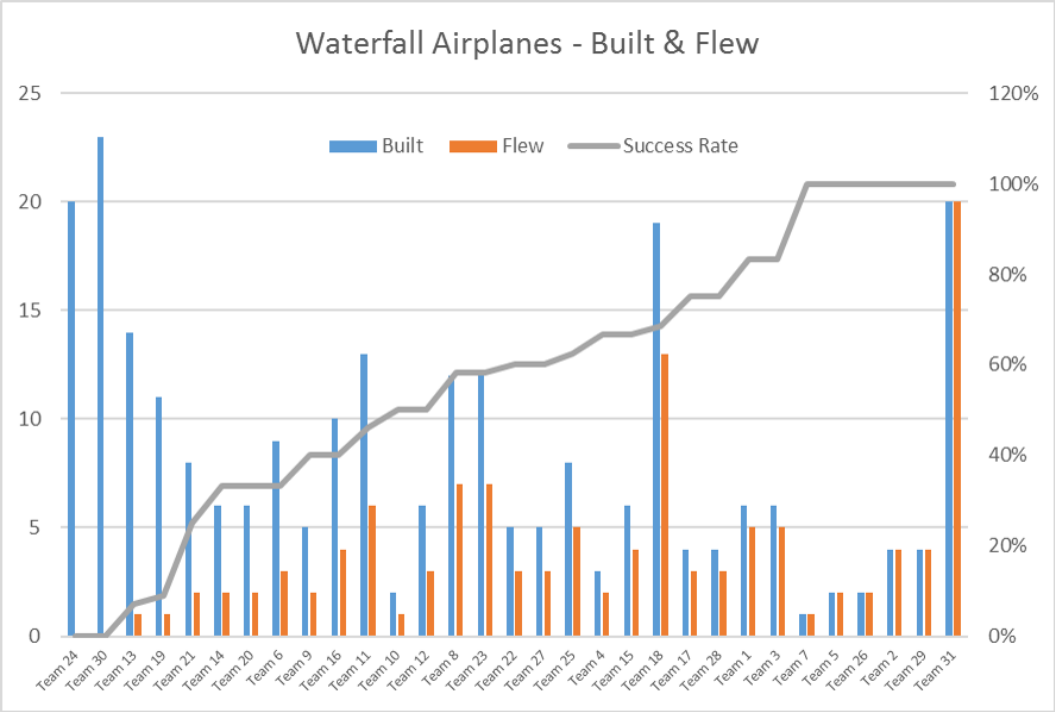

Figure 1. Airplanes built and that flew – Waterfall

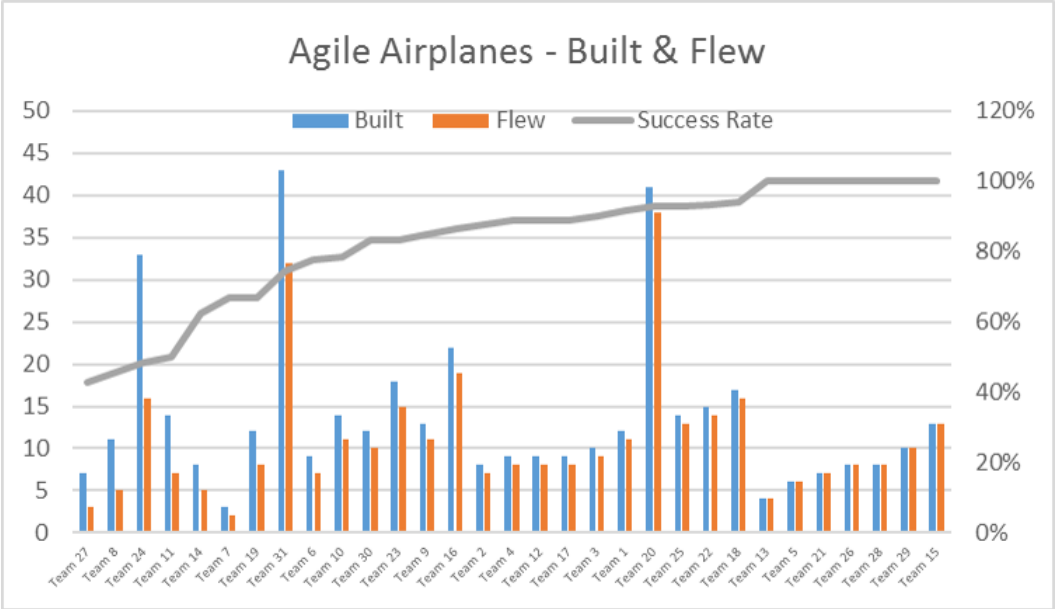

Figure 2. Airplanes built and that flew - Agile

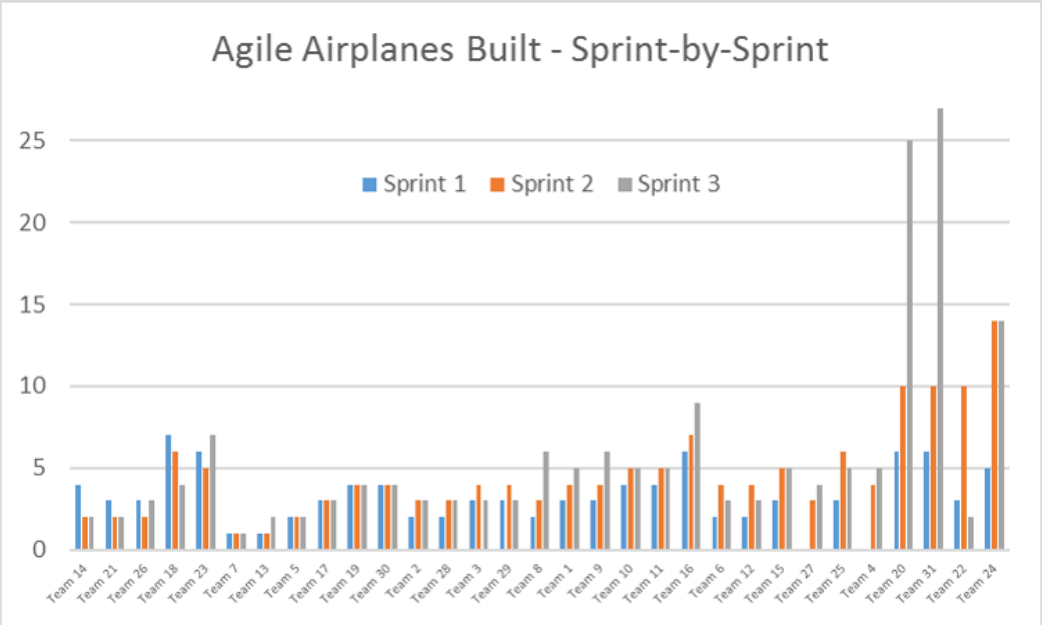

Figure 3. Airplanes built during each sprint

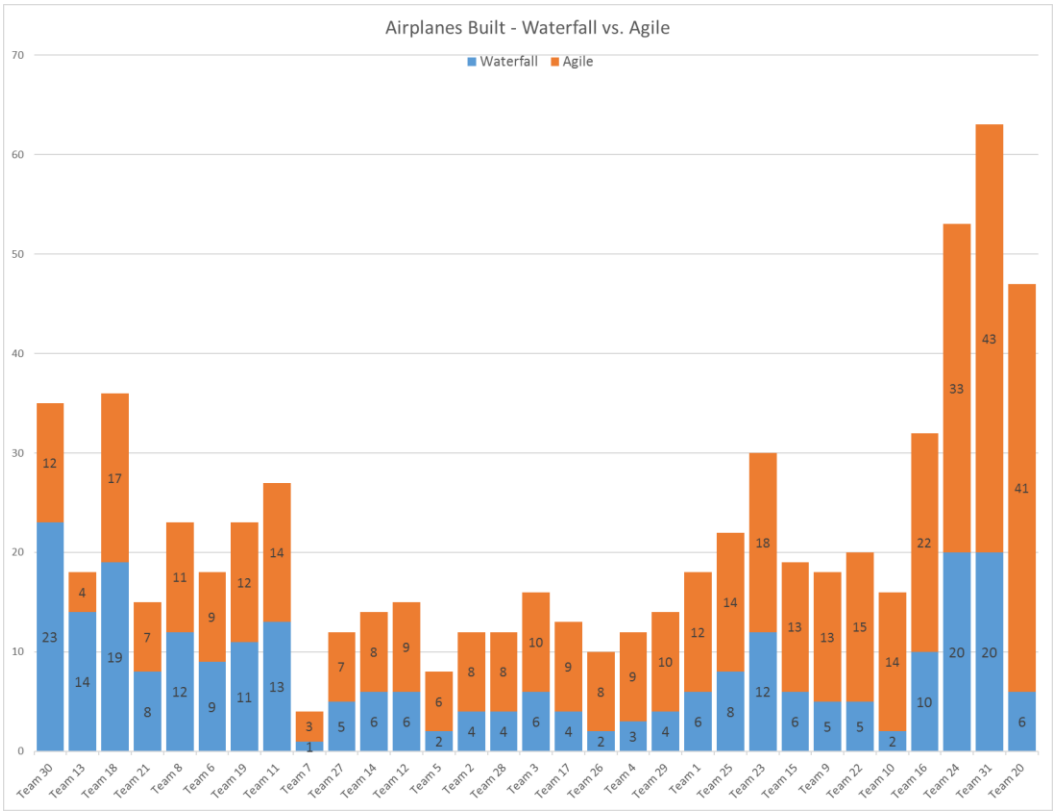

Figure 4. Number of airplanes built by each team using waterfall and agile methods

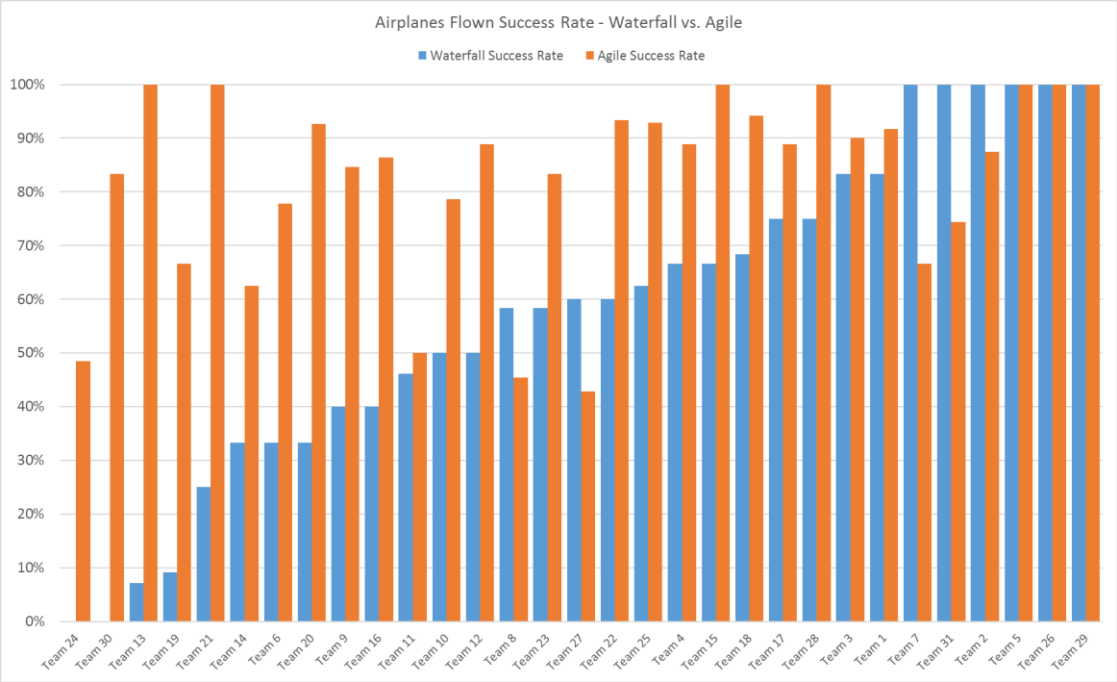

Figure 5. Increased average success rate

31 Teams in 5 Class Sections			AGILE BY SPRINT															AGILE ALL SPRINTS			WATERFALL vs. AGILE					% improvement Agile over Waterfall		
			Built				Flew				Success Rate (%)				TOTAL					Built		Flew		Success Rate			
			S1	S2	S3		S1	S2	S3		S1	S2	S3		Built	Flew	Success Rate			Trad'l	Agile	Trad'l	Agile	Trad'l.	Agile	Built	Flew	Success Rate
TEAM 1	Class A		3	4	5	**12**	3	4	4	**11**	100%	100%	80%	**92%**	**12**	**11**	**92%**			**6**	12	**5**	11	83%	92%	200%	220%	8%
TEAM 2	Class A		2	3	3	**8**	1	3	3	**7**	50%	100%	100%	**88%**	**8**	**7**	**88%**			**4**	8	**4**	7	100%	88%	200%	175%	-13%
TEAM 3	Class A		3	4	3	**10**	3	3	3	**9**	100%	75%	100%	**90%**	**10**	**9**	**90%**			**6**	10	**5**	9	83%	90%	167%	180%	7%
TEAM 4	Class A		0	4	5	**9**	0	3	5	**8**	Error	75%	100%	**89%**	**9**	**8**	**89%**			**3**	9	**2**	8	67%	89%	300%	400%	22%
TEAM 5	Class A		2	2	2	**6**	2	2	2	**6**	100%	100%	100%	**100%**	**6**	**6**	**100%**			**2**	6	**2**	6	100%	100%	300%	300%	0%
TEAM 6	Class A		2	4	3	**9**	2	3	2	**7**	100%	75%	67%	**78%**	**9**	**7**	**78%**			**9**	9	**3**	7	33%	78%	100%	233%	44%
TEAM 1	Class B		1	1	1	**3**	0	1	1	**2**	0%	100%	100%	**67%**	**3**	**2**	**67%**			**1**	3	**1**	2	100%	67%	300%	200%	-33%
TEAM 2	Class B		2	3	6	**11**	1	2	2	**5**	50%	67%	33%	**45%**	**11**	**5**	**45%**			**12**	11	**7**	5	58%	45%	92%	71%	-13%
TEAM 3	Class B		3	4	6	**13**	1	4	6	**11**	33%	100%	100%	**85%**	**13**	**11**	**85%**			**5**	13	**2**	11	40%	85%	260%	550%	45%
TEAM 4	Class B		4	5	5	**14**	2	4	5	**11**	50%	80%	100%	**79%**	**14**	**11**	**79%**			**2**	14	**1**	11	50%	79%	700%	1100%	29%
TEAM 5	Class B		4	5	5	**14**	1	3	3	**7**	25%	60%	60%	**50%**	**14**	**7**	**50%**			**13**	14	**6**	7	46%	50%	108%	117%	4%
TEAM 1	Class C		2	4	3	**9**	2	3	3	**8**	100%	75%	100%	**89%**	**9**	**8**	**89%**			**6**	9	**3**	8	50%	89%	150%	267%	39%
TEAM 2	Class C		1	1	2	**4**	1	1	2	**4**	100%	100%	100%	**100%**	**4**	**4**	**100%**			**14**	4	**1**	4	7%	100%	29%	400%	93%
TEAM 3	Class C		4	2	2	**8**	1	2	2	**5**	25%	100%	100%	**63%**	**8**	**5**	**63%**			**6**	8	**2**	5	33%	63%	133%	250%	29%
TEAM 4	Class C		3	5	5	**13**	3	5	5	**13**	100%	100%	100%	**100%**	**13**	**13**	**100%**			**6**	13	**4**	13	67%	100%	217%	325%	33%
TEAM 5	Class C		6	7	9	**22**	5	6	8	**19**	83%	86%	89%	**86%**	**22**	**19**	**86%**			**10**	22	**4**	19	40%	86%	220%	475%	46%
TEAM 6	Class C		3	3	3	**9**	2	3	3	**8**	67%	100%	100%	**89%**	**9**	**8**	**89%**			**4**	9	**3**	8	75%	89%	225%	267%	14%
TEAM 1	Class D		7	6	4	**17**	7	6	3	**16**	100%	100%	75%	**94%**	**17**	**16**	**94%**			**19**	17	**13**	16	68%	94%	89%	123%	26%
TEAM 2	Class D		4	4	4	**12**	1	3	4	**8**	25%	75%	100%	**67%**	**12**	**8**	**67%**			**11**	12	**1**	8	9%	67%	109%	800%	58%
TEAM 3	Class D		6	10	25	**41**	3	10	25	**38**	50%	100%	100%	**93%**	**41**	**38**	**93%**			**6**	41	**2**	38	33%	93%	683%	1900%	59%
TEAM 4	Class D		3	2	2	**7**	3	2	2	**7**	100%	100%	100%	**100%**	**7**	**7**	**100%**			**8**	7	**2**	7	25%	100%	88%	350%	75%
TEAM 5	Class D		3	10	2	**15**	2	10	2	**14**	67%	100%	100%	**93%**	**15**	**14**	**93%**			**5**	15	**3**	14	60%	93%	300%	467%	33%
TEAM 6	Class D		6	5	7	**18**	4	4	7	**15**	67%	80%	100%	**83%**	**18**	**15**	**83%**			**12**	18	**7**	15	58%	83%	150%	214%	25%
TEAM 7	Class D		5	14	14	**33**	2	14		**16**	40%	100%	0%	**48%**	**33**	**16**	**48%**			**20**	33	**0**	16	0%	48%	165%	Error	48%
TEAM 1	Class E		3	6	5	**14**	2	6	5	**13**	67%	100%	100%	**93%**	**14**	**13**	**93%**			**8**	14	**5**	13	63%	93%	175%	260%	30%
TEAM 2	Class E		3	2	3	**8**	3	2	3	**8**	100%	100%	100%	**100%**	**8**	**8**	**100%**			**2**	8	**2**	8	100%	100%	400%	400%	0%
TEAM 3	Class E		0	3	4	**7**	0	2	1	**3**	Error	67%	25%	**43%**	**7**	**3**	**43%**			**5**	7	**3**	3	60%	43%	140%	100%	-17%
TEAM 4	Class E		2	3	3	**8**	2	3	3	**8**	100%	100%	100%	**100%**	**8**	**8**	**100%**			**4**	8	**3**	8	75%	100%	200%	267%	25%
TEAM 5	Class E		3	4	3	**10**	3	4	3	**10**	100%	100%	100%	**100%**	**10**	**10**	**100%**			**4**	10	**4**	10	100%	100%	250%	250%	0%
TEAM 6	Class E		4	4	4	**12**	2	4	4	**10**	50%	100%	100%	**83%**	**12**	**10**	**83%**			**23**	12	**0**	10	0%	83%	52%	Error	83%
TEAM 7	Class E		6	10	27	**43**	2	3	27	**32**	33%	30%	100%	**74%**	**43**	**32**	**74%**			**20**	43	**20**	32	100%	74%	215%	160%	-26%
	ALL		100	144	175	**419**	66	125	102	**339**	**66%**	**87%**	**58%**	**81%**	**419**	**339**	**81%**			**256**	**419**	**120**	**339**	**47%**	**81%**	**164%**	**283%**	**34%**

Figure 6. Summary of Paper Airplane Exercise results across all sections