Module 7 – Retrospective

Terry Sides

Southern New Hampshire University

February 13, 2023

**Agile Scrum Approach**

At its core, Agile changes the methodology and approach to the software development lifecycle. In a waterfall environment a system’s requirements are gathered up front from start to finish. The system is then designed by an architect in its entirety. Once the design is complete, the development teams begin the development process and develops the system from start to finish. Testing then begins on the entire product for defects that need to be addressed. Even with relatively simple systems this process can take months to complete. If a requirement was captured incorrectly, the developers made a mistake in coding the solution, or the requirements and/or the market has changed, the end user would not see these issues until the very end of the process.

The waterfall process shown in the above diagram depicts the process graphically. The process is inefficient because issues and changes may not be identified until very late in the process and are then difficult to address at that late date. This is especially true for changes in the requirements of the system. Even small requirements changes can result in major design changes, redevelopment, and retesting of the system.

In most cases people view Agile as a major departure from Waterfall, the truth however is that it is not. Anytime a system is built we must have requirements. Requirements tell us what we are building. We must have design. The design tells us how we are building the product. Development of course actually builds the product, and testing is evaluating that product against the requirements that were gathered. What agile does do is it breaks the process into manageable dependency ordered chunks. Agile then organizes around a cyclic approach that includes testing at every iteration.

Agile breaks the requirements into much smaller chunks that are expressed as user stories. These stories are assigned point values that represent level of effort. Stories are grouped into sprints or iterations that usually last 2 to 4 weeks. Ideally, for an Agile process to work well, the architect should be working at least 1 iteration ahead of the development team. This allows for problems to be identified by the architect and resolved with the product owner before the stories associated with that sprint are committed for development. The Agile process begins with a project charter. This document defines the Business Case / Vision and the Mission Statement. If defines the team, roles, and people involved in the project, it defines the success criteria, key project risks, and rules of behavior. The project charter is the founding document for the team.

**Roles**

**Product Owner**

The product owner acts as a liaison between the Agile Team and Business Stake Holders. The product owner is responsible for the overall direction of the team, the priorities in terms of work, helping to resolve questions or problems that arise during the development cycle, and perhaps most importantly, the product owner works with Business Stakeholders to define features and capabilities that the system will have. This information is captured as a “User Story”. The product owner also acts as the team’s representative to executive leadership. This person is responsible for communicating issues, questions, resource needs, delivery blockers, and delivery schedule to executive leadership. The product owner participates in the Iteration Planning and Sprint planning to help direct the team’s work effort and decide priority in terms of the Product Backlog. In the case of SNHU Travel, the product owner was responsible for gathering requirements needed for the implementation of a wellness focused travel feature. These requirements were then written as user stories for the team to work on. This effort resulted in the following user stories.

|  |  |
| --- | --- |
| **User Story Number:** | 2 |
| **User Story Name:** | Profile Settings - Number of destinations shown |
| **User Story Size:** | Small |
|  |  |
| **User Story Value Statement:** | As an end user I want to be able to control the number of vacation packages shown when I login to the system. |
|  |  |
| **Acceptance Criteria:** | • The user should be able to set the number of vacation packages shown at login to the following values  • 5 records  • 10 records  • 20 records  • The user should be able to enter a custom value in a text box. This value may not exceed 50. |

|  |  |
| --- | --- |
| **User Story Number:** | 4 |
| **User Story Name:** | Set travel package price range in User Profile Settings |
| **User Story Size:** | Medium |
|  |  |
| **User Story Value Statement:** | As an end user I want to be able to set a price range in my profile settings to control the list of popular travel packages shown. |
|  |  |
| **Acceptance Criteria:** | • The user will be able to specifiy a price range in their profile settings page. The price range will be expressed in the users default currency based on location.  • The settings page will include a text box to allow the user to enter a minimum value. This value cannot be less than 1 dollar.  • The settings page will include a text box to allow the user to enter a maximum value. This value cannot be less than 1 dollar. |

|  |  |
| --- | --- |
| **User Story Number:** | 6 |
| **User Story Name:** | Package Sorting |
| **User Story Size:** | Large |
|  |  |
| **User Story Value Statement:** | As an end user I would like to be able to sort and filter the list of vacation packages shown. |
|  |  |
| **Acceptance Criteria:** | • The vacation packages list will have the ability to sort the results on;  • Price  • Destination Location Country / Region  • Vacation Type • The vacation packages list will have the ability to filter the results on;  • Price  • Destination Location Country / Region  • Vacation Type |

**Scrum Master**

Human communication is fundamental to most everything we do as a group. Good leadership, clear and concise communication, feedback, and retrospectives are particularly important in the realm of software engineering and project management. Scrum events and artifacts like a product roadmap, architectural runway, iteration / sprint planning, epics, features, and stories capture the vision of what is to be accomplished, but the rubber meets the road so to speak, in the daily standups. This single scrum event is the most important of them all. It is this team meeting where the team comes together, shares ideas, problems, roadblocks, and progress overall. The Scrum Master is servant leader on the team that facilitates these activities, coordinates communication, and ensure that the team is following the principals of Agile.

**Architect**

The architect role on a scrum / agile team is responsible for defining the architectural runway as well as any enabler product backlog items that need to be addressed pre-development stage. This person is responsible for determining the component structure of a system, identifies interfaces and system touchpoints. The architect ensures that the solution under development remains in line with the overarching goals and vision of the organization. Architects are normally spread across several Agile teams that work together to form an Agile Release Train (ART). Although we did not cover this role in this course, it is nonetheless critical to the success of an organization.

**Development / Testing**

In an Agile setting, the development and testing roles are typically both performed by Software Development Engineers. Using test first practices like Test-Driven Development (TDD) and Behavior-Driven development (BDD) for automated acceptance testing, both of these roles will typically write considerable amounts of code. Test cases and test code are housed in a shared repository and available to all for transparency. Code is written according to the user stories that are part of that sprint. It is checked into a shared source code repository and a pull request is created. This request is code reviewed by another developer, and if approved is scheduled for a release.

**Process Implementation**

The Agile manifesto says that organizations should prefer individuals and interactions over processes and tools. This is a good general guideline, however anyone who has worked in a large development organization will likely agree that process is still needed. If you have an organization with 50 different Agile teams all working in their own way, with different structures, directions, intent and tools what you end up with is a mess. In fact, Agile at its core is nothing more than a process. The difference is, that in Agile the process is tailored around accommodating change. This was demonstrated in the course by the changing requirements where wellness focused vacation packages were identified as critical to the success of the SNHU travel application. In this scenario a change to the design was needed to accommodate categorization of the travel packages on offer. The following email was sent to clarify this user story requirement.

|  |
| --- |
| To: Jane Doe, Product Owner; John Smith, Test Engineer  Date: February 3, 2023  Subject: Categorization of Travel Packages  Ms. Doe, Mr. Smith  In our meeting earlier today, we discussed the change in focus of the carousel control displaying travel packages to the user. This change is such that the focus will now be detox/wellness. This request raises a larger issue in this platform. We do not want to build a system such that every time the business wants to change the focus, they must come back to development to make this change. In support of this I would request the following 2 user stories.  We need a user story that documents the degree of categorization desired of this system. For example, we may say that a travel package will have 3 levels of categorization. The top level might be “Health and Wellness”, the second level might be “Addiction” and the third level might be “Smoking Cessation”.  We need a user story that captures the business need for being able to manage and change the focus of the carousel control as market conditions dictate.  Please let us know if you have any questions.  Thanks  Terry |

**Supporting Tools**

Organizing a team to accomplish a goal as complex as software development especially given a large and complex application is a daunting task. Considerable information will be gathered, correlated, and disseminated to the team over multiple sprints covering large amounts of time. Tools that can be used to track the artifacts of Scrum/Agile such as Epics, Features, Stories, Backlog Items, and tasks are essential. Not just for the team but also for organizational leadership. It allows for progress to be easily tracked and helps to identify bottlenecks and problem areas where skill sets may be lacking. Azure Dev Ops boards provide an excellent toolset to accomplish this task. The following illustrates how an Azure Dev Board might be organized.

Text

Description automatically generated with medium confidence

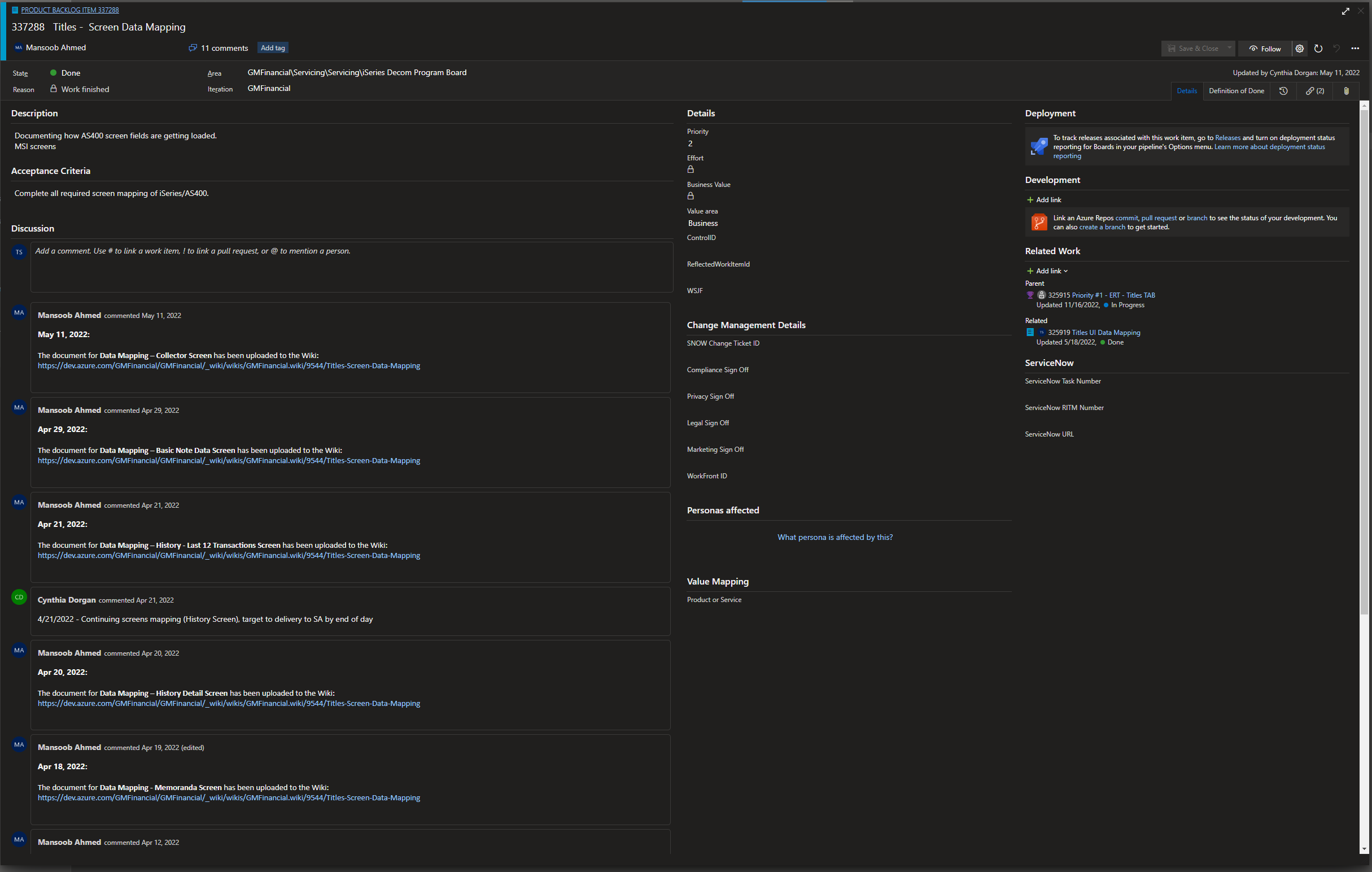
The above image shows an actual project board as defined in Azure Dev Ops. The overall project is setup as an Epic. The Epic contains features that define the work broken out into manageable chunks. As one can see this view of a project, known as a backlog view, allows leadership to see a summary view of the project and the progress being made on each feature.

The following image shows the same project with a feature expanded to show its child product backlog items.

Graphical user interface

Description automatically generated with medium confidence

In this image we can see the product backlog items that are required to implement that feature. The feature in this case is analogous to a user story. For any given feature there may be multiple people working on different aspects of that feature. The next image shows an open product backlog item.



Note that product backlog items or (PBI) is organized around a discussion board metaphor where the individuals involved in that work can discuss issues and ideas without having a formal meeting.

Using tools like Azure Dev Ops is critical to the success of a large project. Keeping track of this much information in an email client, or spreadsheet is very tedious at best and virtually impossible at worst.

Project Effectiveness

Given the project used for this course, the SNHU Travel Project, it’s fair to say that Agile would be an effective way to accomplish the goal. Building applications like this one that are self-contained and easily defined are good examples of a project that fits well within the Agile framework. The application is based on well known design patterns like the shopping cart, and there are a multitude of examples on the market that one can draw from for inspiration.

Certainly, handling changing requirements would be considered a pro in the Agile argument. In this case, the change was providing a list of more focused travel packages that required the ability to categorize packages according to type. This was an easy change to accommodate, and probably would not have resulted in a design change because any good architect would have designed categorization into the object model from the beginning.

This leads to the major issue I would consider an argument against Agile. At least as it’s presented in this course. The role of an Architect was never mentioned. Too often Agile teams are organized around a product owner, a scrum master, and then a team of developers. There is no person on the team responsible for looking ahead at the problem. Laying the groundwork, designing the approach, and resolving problems that are foreseen before they are encountered by the development team. User stories are developed in isolation and then tossed over the fence, so to speak, for the development team to work on. What happens when you get ten sprints down the road and then realize you’ve coded yourself down a one-way street? If Agile is going to work effectively there must be someone on the team working at least one, but preferably two sprints ahead of the development team focusing on architecture and the product roadmap.