Reverse Engineering a UAV Prototype using Agile Practices

Lead Author: *Phyllis Marbach*

This example shows how Agile Practices were applied to an unmanned air vehicle (UAV) that had been developed as a prototype and was intended to be produced and marketed (Marbach 2012). At the time it was required to have the Federal Aviation Agency (FAA) approve the use of a UAV in populated areas. FAA required artifacts such as requirements, architecture representations and test procedures to grant this approval. A team was established to reverse engineer the artifacts needed from the operational prototypes being flown at the time. The test manager requested that the life cycle process to produce these artifacts be agile. Stakeholder needs were determined, and a plan was written to describe the problem and the process to be applied. Then the approach was presented to the team, the product backlog was developed, and a training and planning session was started.

☐

Contents

Description

The agile process is described in the SEBoK here. When defining requirements (Carlson 2010) proposes that the requirements be identified, gathered, defined, and developed in iterations (or sprints). These requirements are written in the User Story format and controlled and managed in a product backlog. The User Stories in the Product backlog are selected and estimated by the team based on importance and need. The most important user stories (or requirements) are prioritized and moved to the top of the product backlog. Then those User Stories are broken into tasks and the tasks are estimated. The number of tasks that the team can complete are then put into the Sprint Backlog (or iteration backlog) and those tasks are then worked on by the team. For this work the UAV and its code were already operational. We had working code, a test bed, user interfaces and user procedures. The goal was to produce requirements documentation, architecture and design diagrams, a trace matrix of tests to requirements, software test descriptions and a Hazard Analysis to take before the FAA.

The team assembled were experienced engineers, but not with this UAV system. There were UAV subject matter experts (SMEs) still on the program, but they were not always available to answer questions or come to reviews. There was existing documentation in program repositories such as charts and operator procedures, but we did not know where to find this information. Given these challenges it was decided to use collaborative tools to manage the information as it was discovered and make it visible to the team and the UAV SMEs.

The first set of information produced was the product backlog. An example of the product backlog is shown in Figure 1. Epics are a set of User Stories that take more than one iteration to complete. A user story is broken into tasks such as those shown in Figure 2. These templates were developed to understand the scope of each task and what the definition of done for that task was. The goal was to identify tasks that take a maximum of 16 work hours to complete. The product backlog was developed and managed in an Application Lifecycle Management (ALM) Tool. Our team did a trades study when selecting a tool for use. Parameters considered in the trade included ease of use, cost, and features of the tool itself. The tool selected was VersionOne.

The team then determined what collaboration tool would

be used to make our discovered information visible. The requirements for this tool were: easy to access, easy to use, easy to comment on and easy to change. At the time these tools were available: Mediawiki, an open source, TWikiTM, another open source, Confluence, SharePoint, and Socialtext. It was decided to use TWikiTM.

For each of the epics in work, such as “Power On”, the Description of Functionality was written in the collaboration tool. Figure 3 shows the list of content in the collaboration tool for the artifacts being developed.

- **30 Epics were created from the User Interface Features, examples:**
  - Power On
  - Start Up Feature
  - Shutdown Feature
  - Operate Component
  - Operate Another Component
- **Product Owner prioritized the most important ones**
- **Each epic has 5 significant backlog items (took 3 Iterations to reach these 5):**
  - Functional Analysis
  - Requirements
  - Hazard Analysis
  - Draft Test Procedure
  - Finalize Test Procedure

Figure 1. Example Product Backlog for an Unmanned Air Vehicle (UAV) (Marbach 2023, used with permission)

Backlog Item Templates

Filter

Move to Project

Title	ID
User Story Template - OLD	B-01017
Task Template - OLD	B-01018
Update Documentation or Work Products Template	B-01243
Functional Analysis Template	B-01225
Research and Document Functionality	TK-02479
Requirements Template	B-01216
Generate Functional Requirements	TK-02420
Peer Review Requirements	TK-02422
Update and Post Requirements	TK-02469
Hazard Analysis Template	B-01219
Identify and Analyze Potential Hazards	TK-02546
Peer Review Hazard Analysis	TK-02547
Update Hazard Analysis	TK-02548
Draft Test Procedures Template	B-01252
Generate Draft Test Procedures	TK-02543
Peer Review Draft Test Procedures	TK-02544
Update and Post Draft Test Procedures	TK-02545
Finalize Test Procedure Template	B-01251
Run Test Procedures	TK-02539
Update and Post Finalized Test Procedures	TK-02540

Figure 2. User Story Templates and Task Templates for Consistent Development (Marbach 2023, used with permission)

Figure 3. Collaboration Tool Table of Contents for a system being analyzed (Marbach 2023, used with permission)

The Collaboration Home Page had an introduction about the analysis underway. It had links to a list of functional threads that were links to the work products themselves. There were also links to the references used, links to the test environment information and links to templates for the work products with instructions. The work products being developed, as shown in Figure 3, were Collaboration Tool Templates, Functional Descriptions, Requirements including Use Cases, Hazard Analysis and Risk Mitigation, and Test Procedures including Test Cases and Test Descriptions.

Once the requirements were complete for one Epic, such as “Power On” the material in the Collaboration Tool was exported into a Word Document and that was parsed into a Requirements Management tool. This team used the Dynamic Object-Oriented Requirements System (DOORS). The final Software Requirement Specification (SRS) was created from DOORS. After peer review the release documents were baselined into the Data

Management Tool Repository that provided Configuration Management control. The Integrated Toolset used for this project is shown in Figure 4.

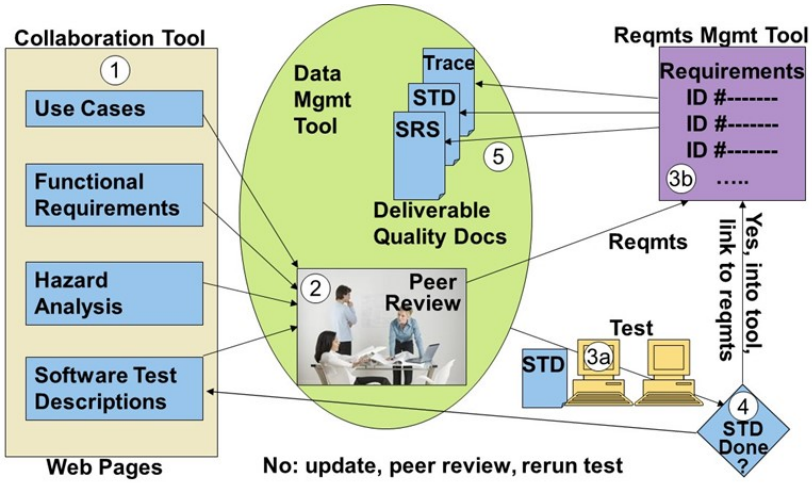

Figure 4. Integrated Toolset for Analyzing a Prototype UAV
(Marbach 2023, used with permission)

The sequence of development started with the prioritized user story to be worked first. Then the Collaboration tool was used to capture the information that members of the reverse engineering team worked with SMEs to reach an understanding of the functional requirements, hazard analysis and software test descriptions for the user story in development. A formal peer review was conducted and when agreed it was ready those documents were parsed into the requirements management tool. The Software Test Description (STD) was used to test the UAV code using the test platform. If the STD is determined to be complete, then it is also parsed into the requirements management tool and traced to the requirements that have been verified by testing. From the requirements management tool formal artifacts such as the Software Requirement Specification, the Software Test Description and the Trace Matrix were produced. Those were put into a configuration management tool. If the STD used for the testing was not Done then any markups were made into the Collaboration Tool and the Peer Review was conducted again. Essentially, each iteration resulted in potentially deliverable products that could be delivered to a stakeholder.

The data management tool, shown in Figure 4 by the green oval contained a repository of draft folders, peer review records, action items created, tracking and closure, a repository of release folders, the calendar, meeting notifications, distribution lists, access control to records, configuration management workflow and approvals, and it provided collaboration across

companies, subcontractors, and customers.

The documentation created included software requirements specification that was created epic by epic rather than all at once, software test descriptions created as each feature is analyzed, and a Hazard Analysis being performed one epic at a time. These documents were updated each increment. An increment is a set of iterations. Each backlog item included conducting peer reviews of the content, as shown in Figure 2 list of tasks of each user story. Records of the peer reviews were maintained in the data management tool as mentioned above. The definition of done for these artifacts was that the work was not complete until the information was posted into the Requirements Management Tool. The Software Test Descriptions were linked to the requirements in the Requirements Management Tool thus beginning the Trace Matrix.

The Agile Practices described in this article can be mapped to LEAN Disciplines as shown in Table 1.

Table 1. Agile Practices Drive LEAN Disciplines (Used with Permission)

LEAN Disciplines	Agile Requirements Analysis
1. Establish Clear Priorities	1. Product backlog is always prioritized; Team works on highest priority items first
2. Eliminate Bad Multitasking – Focus and Finish	2. Team is shielded from interruptions that cause bad multitasking
3. Limit the Release of Work in Process (WIP) to Deliver Earlier	3. Tasks are pulled from the iteration backlog one at a time to limit individual WIP
4. Prepare! Start to Finish	4. Requirements are not selected from the product backlog until everything needed is available
5. Use Checklists to Prevent Defects and Traveled Risk	5. Checklists and guides are used to prevent costly rework
6. Face into and Resolve Issues Quickly	6. Daily stand-up meetings force issues and risks to be identified and resolved quickly
7. Drive Daily Execution	7. Daily stand-up meetings drive team-based execution

Summary

Systems engineering best practice is to perform requirements analysis, verification and validation as a system is being developed. Artifacts are created and

configuration controlled as the system matures. This example describes how an existing operating prototype could be transitioned to a production system by performing requirements analysis, risk mitigation and hazard analysis even after the prototype is developed and operational. There is value in performing these system engineering tasks for an existing prototype to verify it is safe to operate and to achieve approval to fly. Using iterative and incremental development of these artifacts limited the work in process (WIP). The whole team worked one epic at a time to produce artifacts that addressed that one epic and verified the requirements, testing and analysis of hazards relative to that one epic, such as “Power On”. Then they would work the next epic focusing on one capability at a time therefore reinforcing each other’s work quite effectively.

Lessons Learned

- Developing Systems Engineering products such as Systems Requirements Specifications, Hazards Analysis, Test *Procedures, and Verification Trace Matrix in an iterative incremental way is effective.
- Some new to using these principles and methods did resist at first but then saw the value and became advocates of the iterative incremental process.
- The use of tools helped keep the work visible, aiding in communication and accuracy.
- Access to Subject Matter Experts (SMEs) was critical to producing accurate products.
- The team focused on known elements first. Then the knowledge learned was applied to elements with more uncertainty. This applies the Lean Principle of limiting the work in process.
- The team did not start work on an element until they had what was needed to accomplish the analysis. This applies the Lean Principle of working start to finish.

References

Works Cited

Carlson, R., Matzuc, P., 2010, “A Viable Systems Engineering Approach”, Proceedings of the Systems and Software Technology Conference (SSTC), 2010, Salt Lake City, Utah, USA.

Marbach, P., 2012, “An Experience Report on Agile Systems Engineering Requirements Analysis,” Proceedings of the INCOSE-LA Mini-conference, 2012, Los Angeles, CA, USA.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >
SEBoK v. 2.10, released 06 May 2024
