**Spring**

2012

ARINC 838 Statement of Work

Team Information Overload V2

Mike Deats, Scott Griffin, Ryan Neal, Brandon Sutherlin, and Liron Yahdav

Submitted 24 Feb 2012

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Modifier | Comments |
| 0.1 | **23 Jan 2012** | **Team IO 2.0** | **Initial Draft** |
| 0.2 | **31 Jan 2012** | **Mike Deats** | **Updated project cost, success criteria, and metrics.** |
| 0.3 | **7 Feb 2012** | **Ryan Neal** | **Added risk section and reformatted tables for consistency** |
| 0.4 | **19 Feb 2012** | **Brandon Sutherlin** | **Synchronized voice added conclusion** |
| 0.5 | **24 Feb 2012** | **Scott Griffin** | **Incorporated reviewer comments** |

Table of Contents

[Executive Summary 4](#_Toc317835189)

[Team and Client Introduction 4](#_Toc317835190)

[Client 4](#_Toc317835191)

[Team 4](#_Toc317835192)

[Client Needs 5](#_Toc317835193)

[Services Provided 5](#_Toc317835194)

[Project Work Breakdown Structure 5](#_Toc317835195)

[Milestones 7](#_Toc317835196)

[Project Cost 8](#_Toc317835197)

[Risk 8](#_Toc317835198)

[Success Criteria 10](#_Toc317835199)

[Metrics 11](#_Toc317835200)

[Process 11](#_Toc317835201)

[Conclusion 12](#_Toc317835202)

[Appendix A: Glossary 13](#_Toc317835203)

List of Tables

Table 2: Proposed user stories 6

Table 3: Create XDF builder story breakdown 6

Table 4: Create BDF builder story breakdown 6

Table 5: Iteration breakdown 7

Table 6: Release & Deliverable breakdown 7

Table 7: ARINC 838 Cost Breakdown 8

Table 8: Iteration Cost Breakdown 8

Table 9: Risk Mitigation Strategies 10

Table 10: GQM Results 11

# Executive Summary

This document outlines the goals, Client needs, success criteria, risks, and work-breakdown structure that Team Information Overload 2.0 (IO2) will apply to develop the first prototype avionics-software loader based on the Aeronautical Radio, Incorporated (ARINC) 838 standard. It identifies the primary Client needs as the following:

* Determine the quality and practicability of the ARINC 838 specification
* Provide both informal and formal gap analysis of the specification
* Examine the efficacy of Agile as a method for developing software in the aerospace industry

This Statement of Work (SoW) also identifies the risks management plan. It defines the criteria IO2 will use to determine the success or failure of the project, which is heavily dependent upon the Client acceptance of user stories. It lays out a preliminary schedule of major project milestones. Finally, it outlines the Extreme Programming (XP) approach that will be applied, and the project cost in terms of effort, which is estimated to be 1225 man-hours.

# Team and Client Introduction

## Client

Chris Ellison, [christopher.m.ellison@boeing.com](mailto:christopher.m.ellison@boeing.com), of The Boeing Company has proposed a project to design and implement a prototype system capable of installing various types of software onto a mock avionics unit following the proposed ARINC 838 standard.

## Team

Team Information Overload 2.0 (IO2) will work on the ARINC 838 project. This team consists of the following software engineers:

|  |  |
| --- | --- |
| Member | Email |
| Mike Deats | [mike.deats@sv.cmu.edu](mailto:mike.deats@sv.cmu.edu) |
| Scott Griffin | [scott.griffin@sv.cmu.edu](mailto:scott.griffin@sv.cmu.edu) |
| Ryan Neal | [ryan.neal@sv.cmu.edu](mailto:ryan.neal@sv.cmu.edu) |
| Brandon Sutherlin | [brandon.sutherlin@sv.cmu.edu](mailto:brandon.sutherlin@sv.cmu.edu) |
| Liron Yahdav | [liron.yahdav@sv.cmu.edu](mailto:liron.yahdav@sv.cmu.edu) |

Table : ARINC 838 Software Engineers

Edward Katz, Ph.D., [ed.katz@sv.cmu.edu](mailto:ed.katz@sv.cmu.edu), of Carnegie Mellon Silicon Valley is fulfilling the role of faculty adviser and coach for IO2.

# Client Needs[[1]](#footnote-1)

Many modern avionics systems include field-programmable, non-volatile memory. These systems may be “safety-critical systems.”[[2]](#footnote-2) Therefore, the software installed on these systems must be tightly configuration controlled and its integrity verified by the system before installation. On legacy avionics systems, the software delivery utilizes many different formats. ARINC 838 aims to standardize and simplify the software load format.

The Client desires feedback on the ARINC 838 standard, which is still in development. This may include proposing changes to the specification that would define the minimal set of data needed to allow loading software on a target device. The Client also desires a small, certifiable, proof of concept software loader that utilizes the ARINC 838 standard to install software on a wide range of hardware.

The Client also desires to demonstrate that the aerospace industry can successfully utilize Agile techniques and methodologies. The Client believes, and IO2 concurs, that the use of Agile will help lower maintenance and development costs for future commercial avionics systems including flight-critical systems. Specifically, this project will examine the efficacy of Agile software practices when applied to developing software that is certifiable by DO-178B level-D standards. It is noteworthy that while the Client desires to show that Agile practices are effective for aerospace projects, both IO2 and the Client are open to evidence against this hypothesis.

# Services Provided

IO2 will learn the ARINC 838 standard. Once digested, the team will use the standard to accomplish three major tasks:

1. Implement a basic software loader and verifier using the ARINC 838 standard
2. Document any shortcomings in the standard and identify a minimal set of data sufficient to load software successfully
3. Create a proof of concept module that can handle installation of software to varied avionics systems

This resulting software will demonstrate that a low-power system could create a target loader. In addition, the target loader can employ the information in the ARINC 838 wrapper to install the software correctly. The team will implement and release this prototype software as Open Source Software (OSS).

IO2 will employ an Agile methodology, specifically a modified version of Extreme Programming (XP). This will facilitate the final project goal of examining if Agile can produce aerospace-grade software.

# Project Work Breakdown Structure

A Work Breakdown Structure (WBS) is an artifact of traditional phased development. It provides a sense of the levels of work needed to satisfy a set of requirements. The WBS is particularly useful when the requirements and scope are well established and thereby unlikely to change. Clearly, the WBS employs prediction rather than adaptation.

IO2 has opted to employ XP and has deemed a WBS not to be a value-add activity. Rather, the team captures the Client’s high-level needs in user stories. These stories are simply a reminder to have a conversation about that which adds value to the Client. IO2 will employ planning poker[[3]](#footnote-3) to estimate the stories for the imminent iteration. The Client then orders these user stories based on priority. When the team is ready to implement, they begin the conversation and discuss the functionality required with the Client. The team then further clarifies the story and begins implementation.

|  |  |
| --- | --- |
| ID | User Story |
| 1 | Create XDF builder |
| 2 | Create BDF builder |
| 3 | Create BDF vs. XDF Checker |
| 4 | Create BDF format verifier |
| 5 | Create XDF format verifier (verification vs. format spec) |
| 6 | Create Checksum Verifier |
| 7 | Create mock data loader/target loader (happy path only, full load only) |
| 8 | Implement short loading (target loader) |
| 9 | Implement compatibility checking (target loader) |
| 10 | Create BDF Builder (from XDF) |
| 11 | Create XDF Builder (from BDF) |
| 12 | Evaluate ARINC 838 for potential forward/backward compatibility hooks |
| 13 | Sorting/filtering of lists of LSAPs on dataloader without using Target Hardware IDs |
| 14 | Test load process error cases using ARINC 838 |
| 15 | Add a user interface where applicable |
| 16 | Cross-loading (load 1 target loader and it updates software on other target loaders) or parallel loading (simultaneous loading of multiple target loaders via one dataloader request) |

Table : Proposed user stories

It is likely that each of these initial stories is too large based on the team’s lack of familiarity with the spec. The team will work to break each story down into smaller, more manageable, stories. By way of example, IO2 has examined the first two stories. Each was broken into smaller stories.

|  |  |
| --- | --- |
| Create XDF Builder | Story Points |
| [Write XDF File](https://www.pivotaltracker.com/story/show/23773675) | 5 |
| [Data validation for XDF file](https://www.pivotaltracker.com/story/show/23773693) | 3 |
| [Create a UI for XDF file builder](https://www.pivotaltracker.com/story/show/23773711) | 3 |
| Total | 11 |

Table : Create XDF builder story breakdown

|  |  |
| --- | --- |
| Create XDF Builder | Story Points |
| [Write BDF File](https://www.pivotaltracker.com/story/show/23773793) | 5 |
| [Verify current builder data validation is correct for BDF file](https://www.pivotaltracker.com/story/show/23773813) | 1 |
| [Add read and write BDF to UI](https://www.pivotaltracker.com/story/show/23773803) | 1 |
| [Read BDF file](https://www.pivotaltracker.com/story/show/25232849) | 5 |
| Total | 12 |

Table : Create BDF builder story breakdown

IO2 allotted 10 points to the first iteration. This is the peak velocity reached by the original Team IO. The current team should be able to achieve at least this level due to an additional team member, more experience with Agile, and a more familiar tool set.

## Milestones

IO2 will work the projects in two-week iterations. At the beginning of each iteration, the team will allocate stories until the estimated velocity is reached. The iterations will then breakdown as in Table 5.

|  |  |
| --- | --- |
| Iteration | Start Date |
| 0 | 23 Jan 2012 |
| 1 | 6 Feb 2012 |
| 2 | 20 Feb 2012 |
| 3 | 19 Mar 2012 |
| 4 | 2 Apr 2012 |
| 5 | 16 Apr 2012 |

Table : Iteration breakdown

Based on the Client’s request, IO2 debated the relative merits of measuring the schedule by functionality or by time. The team decided to constrain the schedule and target features at those dates. The course mandates certain dates and so it made the most sense to divide the schedule accordingly. Hence, the ARINC 838 project shall have two releases: 3 Mar 2012 and 4 May 2012. The releases and corresponding deliverables are loosely[[4]](#footnote-4) defined as follows:

|  |  |  |
| --- | --- | --- |
| Release | Date | Activity |
| 1 | 3 Mar 2012 | Completion of iteration 2  Preceding scheduled break |
| 2 | 4 May 2012 | Completion of project  Final deliverables negotiated w/ the Client  Test cases  Design documents  Source code |
| Final Presentation | 30 April 2012 | Project presentation |
| Final Course Deliverables | 30 April 2012 | Engineering Notebook |

Table : Release & Deliverable breakdown

# Project Cost

IO2 will harness its talents and diversity through practices like Pair Programming[[5]](#footnote-5) and Open Workspace to provide a quality product that meets or exceeds the Client’s expectations. Each engineer plans to spend on average 15 – 20 hours a week on the project. The table below provides the total investment for the ARINC 838 project.

|  |  |
| --- | --- |
| Component | Cost |
| Weekly Effort | 17.5 Hours |
| Team Size | 5 People |
| Duration | 14 Weeks |
| Total | 1225 Hours |

Table : ARINC 838 Cost Breakdowns

IO2 has chosen to use two-week iterations. Each iteration shall commence with a planning meeting and conclude with a reflection meeting. In addition, the team will hold weekly meetings with the Client. The following table provides the cost per iteration in hours.

|  |  |
| --- | --- |
| Component | Cost (Hours) |
| Iteration Planning Meeting | 1 |
| Client Meetings | 2 |
| User Story Estimation | 2 |
| Documentation | 2 |
| Coding and Test | 27 |
| Reflection Meeting | 1 |
| Total | 35 x 5 = 127 |

Table : Iteration Cost Breakdown

# Risk

IO2 employed introspection and discussion to analyze the project and identify worst-case scenarios. The team then applied the “5 Whys” methodology to identify the core risks. Figure 1 provides the details of the “5 Whys” analysis and resulting root causes. Finally the team used Wideband Delphi technique to create a risk profile based the formula, *Risk Profile = Impact x Probability*. Table 9 provides the resulting initial risk profile.It is noteworthy that many of the risks originate from concerns with the technology stack and how additional commitments (e.g. day jobs and academics) may affect the schedule.

1. The Specification is missing important components
   1. The specification developers have not completed a prototype implementation
      1. The specification is still a draft
         1. **The project is still in its early stages**
2. The standard is not practical
   1. The specification developers have not completed a prototype implementation
      1. The specification is still a draft
         1. **The project is still in its early stages**
3. Our schedule and effort estimates are unusually inaccurate
   1. **The project is still in its early stages**
   2. **We lack understanding in the domain and specification**
      1. We are students, not hired for our domain knowledge
4. Project runs over schedule
   1. Underestimate the time required for school artifacts
      1. We are trying to exercise all of the techniques we have learned
         1. We have not put all the pieces together before.
   2. Overestimate velocity
      * 1. New domain
        2. New team
   3. Burn out
      1. We are trying to exercise all of the techniques we have learned
         1. **Day jobs take precedence**
   4. Team availability
      1. **Day jobs take precedence**

Figure 1: 5 Why Breakdown

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Risk | Avg. Probability | Avg. Impact | Risk Factor |
| 1 | Difficulty understanding the specification | 3 | 2 | 6 |
| 2 | Real jobs and life events take precedence | 2.2 | 2 | 4.4 |
| 3 | No experience with loadable software tools | 2 | 2 | 4 |
| 4 | More features and school artifacts than the schedule and available resources permit | 2.2 | 1.8 | 3.96 |
| 5 | DO-178B constraints | 2 | 1.6 | 3.2 |
| 6 | The standard is still early in its lifecycle | 1 | 3 | 3 |
| 7 | Gaps in tool knowledge, license knowledge, etc. | 1.4 | 1.6 | 2.24 |

Table : Initial Risk Profile

IO2 created a management plan for the risks identified in Table 9. This management plan includes strategies to prevent and mitigate risks. It is noteworthy that risk-management activities continue throughout the duration of project. IO2 will utilize reflection meetings to reassess risks unless urgency demands otherwise.

|  |  |  |
| --- | --- | --- |
| ID | Risk Factor | Mitigation Strategies |
| 1 | 6 | Rely on Client’s domain knowledge  Utilize pair programming to disseminate knowledge |
| 2 | 4.4 | Keep calendar up to date and team informed  Anticipate individual work and personal life loads and plan accordingly |
| 3 | 4 | Rely on Client’s domain knowledge  Utilize pair programming to disseminate knowledge |
| 4 | 3.96 | Divide work between pairs to balance artifacts and code |
| 5 | 3.2 | Utilize pair programming to keep high coding standard  Use SCM system to track commits to stories |
| 6 | 3 | Utilize Client’s close contact to fix developing problems |
| 7 | 2.24 | Utilize school resources to understand constraints  Continually be aware of license  Utilize pair programming to maximize domain knowledge |

Table : Risk Mitigation Strategies

# Metrics

In order for any team to measure its progress, metrics must be collected and utilized. To select a reasonable and appropriate set of metrics, Team IO2 used the “Goal, Question, Metric” or GQM approach for metrics selection. Defining a goal, coming up with questions that address if the team is meeting the goal, and finally selecting metrics that will provide answers to the questions.

The following table shows the results of the GQM exercise.

|  |  |  |
| --- | --- | --- |
| Goal | Question | Metric |
| Apply Agile software practices to DO-178B, Level D project | Are we meeting the 28 objectives of Level D? | Code Coverage |
| Improving use story estimation | Is team’s actual effort for each user story proportional to the point estimation? | Effort logged per story, Story points |
| Stabilize velocity (+/- 10%) | Each iteration, how much does the velocity change? | Average velocity |
| Maximize time spent coding (75% of effort) | How much effort is the team spending on coding versus ‘overhead’ (e.g. documentation, meetings) | Effort logged, time spent coding |

Table : GQM Results

# Process

IO2 selected the process of XP for use on the ARINC 838 project. However, IO2 is modifying the process by limiting the user of pair programming to critical pieces of code and requiring a link to a story for each code commit.

The team opted to limit pair programming because:

* As a remote team, pair programming is burdensome
  + Tools
  + Time coordination
* With the use of TDD, pair programming is not beneficial for trivial tasks
* All developers are skilled, reducing the value of the teaching aspect of pair programming

Please note that IO2 is not eliminating pairing. There is value in pair programming for teaching, sharing knowledge of the code base, and code review. However, IO2 will reserve this tool for situations in which it can provide the most value.

IO2 requires each commit to tie to a story in order to provide traceability in the project. Embedding Pivotal Tracker IDs within each commit message does this. This simple step provides information to identify all of the code associated a particular story. This traceability is a critical component of satisfying D0-178B requirements. Developers enforce this requirement using a Git hook on their workstation.

# Success Criteria

This project has several major tasks as outlined in the “Services Provided” section. The first two are the main goals for the project:

1. Implement a basic software loader
2. Document any shortcomings in the standard

This is the main concern for the Client, and thus is the focus of the project. Completing these tasks will constitute a successful project. The third task, “Create a demonstration-quality module” is a stretch goal that IOS will attempt based on our progress as the project unfolds.

IO2 will utilize the Client acceptance of user stories as a measurement of value and success. As stories are completed in each iteration, the Client will review the product and accept the story if the acceptance criteria are met. This continual process allows the team to confirm with the client that the stories are meeting their needs.

The use of Agile is also a critical concern for the Client. As stated previously in this document, there is a desire to demonstrate that a team can utilize Agile software practices to produce avionics software that is certifiable. This industry has been very slow to adopt Agile as an accepted practice, mainly due to the perception that Agile does not provide the same rigor and discipline as traditional approaches. At the end of this project, IO2 will provide a document examining their experience utilizing an Agile methodology to develop aerospace software.

# Conclusion

Team IO2 is a five-member team of software engineers with a proven record of accomplishment. This team will employ a tailored version of XP to solve the Client’s hard problems including, assessing the ARINC 838 specification quality and practicability, implementing basic loader to said specification, and assessing the efficacy of Agile methodologies on aerospace projects. This endeavor will include two releases constrained by time rather than functionality. IOS will deliver all deliverables by the hard stop of May 4, 2012. The Agile approach excludes any documentation that does not add value including the WBS. IO2 does employ value-add activities including but not limited to a risk management plan and metrics tracking.

Appendix A: Glossary

|  |  |  |
| --- | --- | --- |
| **Term** | **Definition** | **Source** |
| **certifiable** | DO-178B, Software Considerations in Airborne Systems and Equipment Certification is a document dealing with the safety of software used in airborne systems. | <http://en.wikipedia.org/wiki/DO-178B> |
| **safety-critical system** | A computer, electronic or electromechanical system whose failure may cause injury or death to human beings. | <http://dictionary.reference.com/browse/safety-critical+system> |

1. Loadable Software Specification, Boeing Project Proposal, Christopher Ellison, December 2011 [↑](#footnote-ref-1)
2. See Appendix A for a glossary [↑](#footnote-ref-2)
3. Mike Cohn (November 2005). "[Agile Estimating and Planning](http://www.mountaingoatsoftware.com/books/1)". Mountain Goat Software. [↑](#footnote-ref-3)
4. The term *loosely* is used because Agile is by definition adaptive rather than predictive, hence the change is expected around releases [↑](#footnote-ref-4)
5. The team does not intend to use Pair Programming for rote or boilerplate tasks [↑](#footnote-ref-5)