

A Roadmap for Using Agile Development in a Traditional Environment

Barbara Streiffert and Thomas Starbird†*

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109

Sven Grenander‡

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109

One of the newer classes of software engineering techniques is called “Agile Development”. In Agile Development software engineers take small implementation steps and, in some cases, they program in pairs. In addition, they develop automatic tests prior to implementing their small functional piece. Agile Development focuses on rapid turnaround, incremental planning, customer involvement and continuous integration. Agile Development is not the traditional waterfall method or even a rapid prototyping method (although this methodology is closer to Agile Development). At the Jet Propulsion Laboratory (JPL) a few groups have begun Agile Development software implementations. The difficulty with this approach becomes apparent when Agile Development is used in an organization that has specific criteria and requirements handed down for how software development is to be performed. The work at the JPL is performed for the National Aeronautics and Space Agency (NASA). Both organizations have specific requirements, rules and processes for developing software. This paper will discuss some of the initial uses of the Agile Development methodology, the spread of this method and the current status of the successful incorporation into the current JPL development policies and processes.

I. Introduction

Agile Development is a new software methodology that is gaining acceptance throughout the software community. Most Agile developments are used to reduce risk and to produce usable software in short time increments. Generally, the time span lasts one to four weeks. During that time a full life cycle is performed on a small set of functional capabilities. The goal is to have working software at the end of each cycle. The customer is a key part of the process and participates in all aspects of the development process. Agile Development is often considered the antithesis of the Waterfall Methodology. In the Waterfall approach requirements analysis, design, implementation and testing are performed in a linear fashion and only once. Usually the customer only participates in the requirements and design phases and then again during performance testing. The major difference is that the Waterfall approach is considered to be a more predictive approach in being able to determine schedule, deliveries and in general all the functional capabilities and the Agile approach is considered to be more adaptive and deals better with changes in requirements and with user feedback. Waterfall Development falls into the predictive category because the entire development activity including the milestones is determined at the beginning of the software project. Agile Development falls into the adaptive category due to its short time cycles. Changes in functions to be implemented or ordering (reordering) of functions can be based on customer needs.

II. Agile Development at JPL

The beginnings of one of the Jet Propulsion Laboratory’s (JPL’s) agile developments started in a research and development task called Maestro sponsored by the Mars Technology Program in 2004. This task and another related

* Software System Engineer, Section 317, M/S 301-250 D 4800 Oak Grove Dr., Pasadena, CA 91109 USA.

† Principal Investigator, Section 317, M/S 301-250 D 4800 Oak Grove Dr., Pasadena, CA 91109 USA.

‡ System Engineer, Section 317, M/S 301-250 D 4800 Oak Grove Dr., Pasadena, CA 91109 USA.

task, the Next Generation Uplink Planning System (NGUPS) task, have been looking at ways to infuse technology into the Mars Science Laboratory (MSL) ground system software. MSL is the next JPL rover going to Mars in 2009. The Maestro task began using an agile process and spawned a new team, called the Ensemble team. This team now develops not only Maestro software, but also other related software. The team members include contingents at JPL and at the Ames Research Center. This paper focuses on the JPL portion of the Ensemble team's work on Maestro. The NGUPS task hasn't followed many of the precepts of Agile Development, but has incorporated the work from the Ensemble task and has refactored legacy tools to be used with the software from the Ensemble task.

The heart of the process used by the Ensemble team is a weekly cycle. The cycle begins (or ends) with a weekly planning meeting, attended by the software designers and implementers and also by users or representatives of the Flight Projects that are depending upon the work. The meeting begins by reviewing the status of each of the detailed tasks that had been planned for the previous week. The tasks are displayed on a screen using the issue tracking system (JIRA). Each developer has recorded the progress of the previous week. It is common and expected that not all tasks planned are completed; a few tasks than can actually be completed are included each week in case other assigned tasks are completed earlier than predicted or must be postponed because of some unexpected barrier.

The next step is to plan the detailed tasks that are to be implemented in the next week. The lists of required features and the status of failure reports are consulted; discussion of priorities and ordering ensues. Tasks are defined (each in a brief phrase), and each task is assigned. The assignee estimates how long the task will take, typically in the range of 0.25 to 1 day. All tasks involving production code are implemented in pairs. Two people literally sit side-by-side in front of a dual-headed computer with two keyboards. Production code tasks are distinguished from other prototype or investigatory tasks during the planning meeting. At the end of the meeting, all the tasks are entered into JIRA as "unresolved". During the week, their status is changed by the implementer assigned to the task as the task progresses. Figure 1 shows a sample of a JIRA page. This sample shows only open issues because the screen capture was performed at the beginning of the week.

One hallmark of the Ensemble team's process is its use of various tools that support efficient collaborative development and provide documentation on the current work. As mentioned above, JIRA is used to track all tasks. It is also the tool that tracks failure reports. In addition, it is intended, in the future, to be used to track all requirements, linking them to tasks and to tests. Another tool used extensively is Confluence, which is a wiki. A wiki is server software that is used to create and modify web page content. All documentation (other than documentation in the code itself) is added to Confluence: meeting notes, design decisions, diagrams for user interface interactions, scenarios for demonstrations, use cases, the software development process, detailed requirements, and "how to" pages. Confluence allows anyone on the team to document the work that they are implementing. It also allows the team to document designs or any other data that should be available to the team. Figure 2 shows a sample Confluence page. The document shown is a copy of this paper that resides in Confluence. One characteristic of Agile Development is that it produces less documentation than other methods because it relies on face-to-face communication. The Ensemble team addresses this criticism using by Confluence and JIRA.

A third tool called Cruise Control is also used by the team. Cruise Control is the build management tool. Cruise Control understands the relationships of the various components of software to the applications that they are a part of, and not only builds the software, but also runs all necessary tests to check that the insertion of the new component(s) is viable. If the build fails because a test fails, e-mail notification is sent to the appropriate software developer(s). Figure 3 shows the summary results maintained by Cruise Control. JIRA, Confluence, and Cruise Control are all commercial products.

Along with the implementation that occurs each week, the designers hold a weekly meeting where they present and discuss design issues, particularly ones related to the user interface. Mockups of possible screen views, or prototypes of possible user interaction, are shown. Alternatives and trades are discussed and weighed. When decisions are made, they are documented in Confluence, and become the design goal for the assigned implementers.

The NGUPS team is responsible for integrating the Ensemble work with legacy tools. The legacy tools have been altered to work effectively with Ensemble as well as provide interprocess communication among all the tools. Periodically, members from both teams participate in pair programming depending on the "story" (functional element) to be implemented.

As the work from these two groups began to be noticed for its excellence and high productivity, a third group decided to adopt the work and further its development so that it could become multi-mission ground sequencing software for in-situ missions such as rovers. Until now the multi-mission ground system software has been developed according to all the rules and precepts of a traditional system including the traditional set of documents, reviews, and other gates that are not typically part of the Agile Development process – at least not part of it in a traditional sense.

JIRA Ensemble JIRA
 HOME BROWSE PROJECT FIND ISSUES CREATE NEW ISSUE

All Projects : Maestro (Key: MAE)

Project Lead: Jeff Norris
URL: <http://gong.jpl.nasa.gov:8080/>
Description:
 Main development effort for Maestro.

Create a new issue in project Maestro
 Release Notes

Select: **Open Issues** Road Map Change Log

Road Map
 View personal road map
 Scope: next 3 versions | all versions

Iteration 82 (26/May/06 | Release Notes)
 5/22 - 5/28 Progress: 0 of 2 issues have been resolved

- MAE-1168 UNRESOLVED WA/TDA for M&A task
- MAE-1155 UNRESOLVED data product view refactor

Iteration 81 (19/May/06 | Release Notes)
 5/15 - 5/19 Progress: 0 of 24 issues have been resolved

- MAE-1150 UNRESOLVED ATHLETE telemetry: JMS-PVM converter
- MAE-1178 UNRESOLVED Athlete moldump log products
- MAE-1176 UNRESOLVED Athsim Sequence upload
- MAE-1177 UNRESOLVED Athsim command dictionary
- MAE-1181 UNRESOLVED CEV: ISS data server
- MAE-1159 UNRESOLVED Choice should also have a display value
- MAE-1175 UNRESOLVED Data Type for Mini-TES missing

Figure 1 shows a sample page from JIRA. This page was taken at the beginning of the week so all of the issues are unresolved.

Dashboard > Ensemble Development > Ensemble Software Development Process > A Roadmap for Using Agile Development in a Traditional Environment

Welcome Thomas

Ensemble Development
A Roadmap for Using Agile Development in a Traditional Environment
 Added by Thomas Starbird, last edited by Thomas Starbird on May 15, 2006 (view change)
 Labels: (None) EDIT

This is a draft of a paper being submitted to SpaceOps 2006

A Roadmap for Using Agile Development in a Traditional Environment

Barbara Streiffert* and Thomas Starbird**
 Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109
 Sven Grenander***
 Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109

Abstract

One of the newer classes of software engineering techniques is called "Agile Development". In Agile Development software engineers take small implementation steps and, in some cases, they program in pairs. In addition, they develop automatic tests prior to implementing their small functional piece. Agile Development focuses on rapid turnaround, incremental planning, customer involvement and continuous integration. Agile Development is not the traditional waterfall method or even a rapid prototyping method (although this methodology is closer to Agile Development). At Jet Propulsion Laboratory (JPL) a few groups have begun Agile Development software implementations. The difficulty with this approach becomes apparent when Agile Development is used in an organization that has specific criteria and requirements handed down for how software development is to be performed. The work at the JPL is performed for the National Aeronautics and Space Agency (NASA). Both organizations have specific requirements, rules and processes for developing software. This paper will discuss the some of the initial uses of the Agile Development methodology, the spread of this method and the current status of the successful incorporation into the current JPL development policies.

I. Introduction

Agile Development is a new software methodology that is gaining acceptance throughout the software community. Most Agile

Figure 2 shows a sample page in Confluence, a wiki that offers users the ability to document and edit any data in a web browser.

CruiseControl Status Page

[Turn autorefresh on](#)

Project	Last build result	Last build time	Last successful build time	Last label
MERDataProductCatalogPopulator	passed	05/15/2006 04:45:14	05/15/2006 04:45:14	build.666
MSLFeature	passed	05/15/2006 04:22:13	05/15/2006 04:22:13	build.1196
MaestroMER	passed	05/15/2006 04:01:13	05/15/2006 04:01:13	build.859
Moonrise	passed	05/11/2005 17:34:10	05/11/2005 17:34:10	build.133
PDSToJPEG	passed	05/15/2006 04:18:32	05/15/2006 04:18:32	build.601
Phoenix	passed	05/15/2006 05:01:28	05/15/2006 05:01:28	build.593
RoverWare	passed	05/15/2006 04:51:43	05/15/2006 04:51:43	build.669
SASPaH	passed	05/15/2006 04:36:35	05/15/2006 04:36:35	build.313
Total	8			
Passed	8	100%		

listing generated at 13:14

Figure 3 shows the results of the Cruise Control run to build the software. Features contain multiple components.

III. Traditional Approach

All NASA programs must follow NASA Procedural Requirements (NPRs). These requirements are for all development phases of projects. For software projects NASA developed NPR 7150.2. California Institute of Technology's (Caltech's) JPL has developed a response to NPR 7150.2 called the Software Development Requirements (SDR). The SDR has requirements that correspond to the applicable NASA requirements in NPR 7150.2. All software development at JPL is required to be in compliance with the SDR. The SDR is also compliant with the goals of Capability Maturity Model Integrated (CMMI).

The SDR contains requirements that deal with all aspects of a software development process. It includes information on who is required to comply with the requirements in the SDR (basically all who develop software at JPL) and the various software classifications. The software classifications are A (Human Rated), B (Mission Critical), C (Contributes to Mission Objectives but not Critical) and D (No Impact to Mission Primary or Secondary Objectives). The SDR requires that a software management plan be developed. This plan describes how the software will be managed including roles and responsibilities, reviews, design, development, testing, delivery, documentation, etc. The plan must also deal with the budget, the work breakdown structure, risk management and software acquisition, if needed. The software management plan can point to other documents, such as a risk management plan, provided the other document contains the necessary information about software. The key is that this information must be documented. Typically, the Agile Development methodology doesn't describe documentation, doesn't deal with budgets, and mentions risk mostly in saying that the short cycle and user involvement mitigates the risk by ensuring that the software remains on track. In addition, JPL didn't have a process that mapped Agile Development to the SDR.

At this point in time when JPL is working on being CMMI compliant, there is a support group called the Software Quality Improvement group. This group is to help each software development team to be compliant with the SDR and CMMI. This group has been trained in software development processes, the SDR, NASA NPR's and CMMI. In addition to working with software development teams for compliance they are working with each project and line organization to develop a series of processes (including documentation) that describe software development at JPL.

IV. Combining the Two Approaches

The current task has become one of how to marry the two approaches. There have been meetings and efforts to understand this new development model. The SQI group held a series of weekly meetings to understand how the Ensemble group has implemented the Agile Development methodology. The System Engineers from the NGUPS team and the Ensemble team have started work on documenting the implementation of Agile Development in a

Confluence document called the Ensemble Software Development Process. In the newly created document the basic tenants of Agile Development have been stated with some modifications. Input from the SQI group has been received and accommodated. In general, because the Ensemble work is part of a larger system there are additional requirements on the development technique that must be met in the areas of planning, reviews, testing, etc.

For example, the process discusses planning of the software. It talks about planning in weekly increments, but it also deals with high level planning that creates a long-term agreement on release dates and general contents of the software at a high level functional capability. The next level of planning covers the planning effort that takes place after a delivery or a demonstration of the software. At this level of planning the detailed functional requirements are defined with the aid and review of the customer. These requirements are planned to be documented in JIRA. Additionally the division between core functions (project-independent ones) and project-specific functions is drawn. The final level of planning is the normal weekly Agile Development planning that occurs as part of the Agile Development Process. Typically the week of planning and implementation follows the outline listed below:

- 1) Determine Requirement/Issue/Functional Capability (typically can be completed in 1 week or less)
- 2) Discuss in weekly design team meeting
- 3) Document design in Confluence
- 4) Assign tasks for item 1 in weekly planning meeting
- 5) Enter tasks into JIRA
- 6) Software pairs design automatic tests
- 7) Software pairs implement item 4
- 8) Software pairs fix any defects
- 9) Software pairs integrate requirement
- 10) Software is rebuilt (Note: software is typically built several times a day)
- 11) Repeat weekly

There are adjustments to the process if needed. For example, if there needs to be investigative work done or a prototypical task must be performed, often an individual software engineer takes on the task instead of working in pairs. At the end of the process (item 10) the software is rebuilt using Cruise Control. All of these steps and their description are part of the Ensemble Software Development Process document.

In the area of reviews the Ensemble implementation of Agile Development calls for weekly status reviews that compare plans versus accomplishments. These meetings use metrics of schedule deviance, team velocity and budget variance to determine progress. The quality metrics of determining the number/kind of software anomalies and the number/kind of requirements implemented are also assessed. The JIRA issue tracking system allows these metrics to be accumulated. In addition to these reviews the group also participates in monthly manager reviews and typically holds demonstrations at least twice a year for stakeholders. In addition, to regularly-scheduled reviews, the Ensemble team participates in science team meetings, ground systems meetings and delivery meetings. At these various meetings the team presents current progress either in the form of presentation materials or impromptu demonstrations or both. However, the most effective review comes from the pair programming in that each participant of the pair reviews the other person's work. In this way there is an almost constant peer review of the software. The number and types of reviews are also documented in the Ensemble Software Development Process.

The Ensemble group performs the traditional life cycle for development. It includes requirements' development, both high level and detail level design, implementation, test and delivery. However, these steps are performed weekly instead of once for each stage. There are always longer term goals that are available, but often they are broken down into weekly chunks. Each automatic build results in a deliverable package. In addition, the Ensemble team provides a more formal delivery to projects.

Testing and delivery is handled in stages as well. For any level of delivery automatic tests are run as regression testing. In addition, prior to delivery every developer is responsible for developing user tests and running them. There are four stages in the delivery portion of the process. There is the stage where the developers deliver to an area that they control. In this stage software can be delivered as frequently as the developers have finished the implementation and have tested the new software. It is delivered to a loosely controlled configuration managed system. The next stage is a more tightly controlled delivery area. This area is where the software is integrated together and is used for preliminary acceptance testing. At some point in time the software then advances to a more tightly controlled area for formal acceptance testing. Finally it is delivered to the project area for user acceptance testing. This delivery scheme is typical of many of JPL software projects and has not been significantly changed from the Waterfall methodology. In JPL legacy systems it is possible to test the core aspects of the software separately from the project specific pieces because the core software is written in code and the adaptation is written in an interpretive language. However, adaptation of the new software can now include actual code in addition to portions in an interpretive language, so it is often not possible to test the two pieces separately. Figure 4 shows a

graphic representation of the delivery stages. This diagram also appears in the Ensemble Software Development Process document.

In addition to handling the normal aspects of planning, implementing, testing and delivery, the process also deals with the other SDR required aspects such as software acquisition and software safety. Generally, software acquisition is in the category of third-party commercial off-the-shelf and open source software. The Ensemble group does not contract out software. However, part of their development team is at Ames Research Center. Those team members participate in the same way as local team members attending the design and status meeting and adhering to much the same process for planning, design, software implementation and testing as the rest of the team. The Ensemble Software Development Process defines and explains how each of the SDR requirements is met.

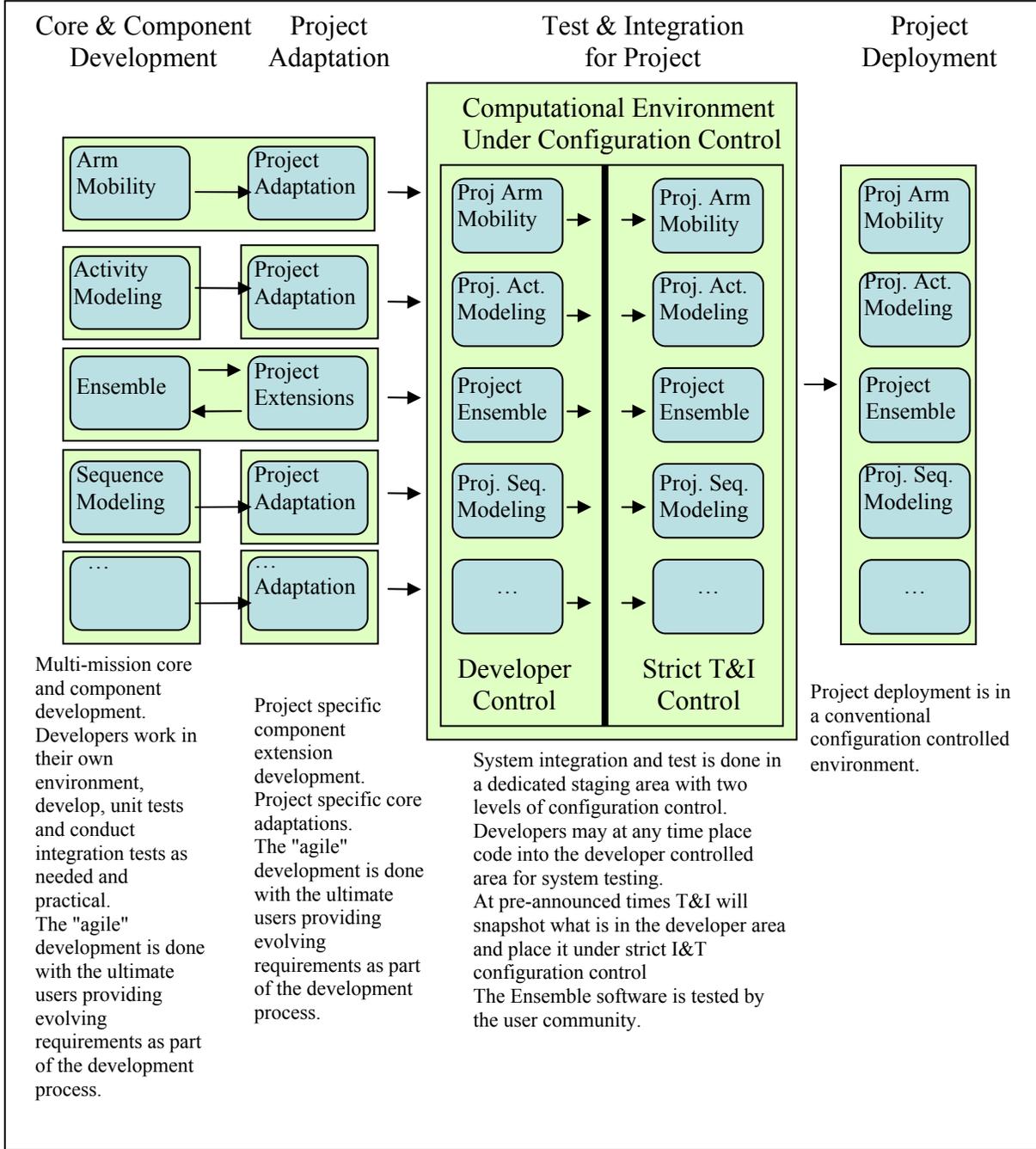


Figure 4 shows the different stages of delivery from the initial delivery that is loosely controlled to the final stage that is tightly controlled.

V. Conclusion

Currently there are a few relatively small areas that still need to be worked in having the Ensemble Software Development Process be completely compliant with the SDR. However, significant progress has been made in that direction and it is expected that the Ensemble Software Development Process will be totally compliant soon. A path has been created and the groups have been working their way to a resolution. Agile Development has become a reality in a traditional system (with a few compromises).

Acknowledgments

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work is funded by the Mars Technology Program.