

AMMOS AUTOMATED DEPLOYMENT SYSTEM

AGILE IN A PLAN-DRIVEN ENVIRONMENT

Erik Monson

Jet Propulsion Laboratory, California Institute of Technology

INTRODUCTION

- ▶ Multi-Mission Operations and Services Office (MGSS) produces a multi-mission Ground Data System known as AMMOS
 - ▶ Used on many NASA missions primarily utilizing the Deep Space Network
- ▶ The Automated Deployment System is a new capability of AMMOS
 - ▶ Service to automate deployment/configuration of AMMOS components to mission environment
 - ▶ One of the first AMMOS capabilities to leverage agile development practices

BACKGROUND

- ▶ ADS developed in a large (primarily plan-driven) software organization, rooted in:
 - ▶ Legacy Software
 - ▶ Legacy Process
 - ▶ A long product lifecycle and funding plan
 - ▶ A very extensive, growing list of mission customers
 - ▶ **A bold plan for modernization**

WHY AGILE?

- ▶ We are designing new applications and modernizing/improving existing applications
- ▶ Our customer base is expanding (and they have opinions as well as requirements!)
- ▶ Other important factors
 - ▶ The application space
 - ▶ The technology space
 - ▶ The talent space

EARLY ATTEMPTS AT AGILE

- ▶ Early attempts at integrating agile practices (e.g. scrum) into MGSS tasks proved challenging
- ▶ Although benefits were often realized....
 - ▶ Stakeholders were more engaged in the design process
 - ▶ More creative solutions and productive development team
- ▶ ...the funding organization often felt detached....
 - ▶ Difficult to tell what the final product would evolve into
- ▶ The plan driven process simply couldn't absorb an agile process *en masse*
 - ▶ Difficult to communicate plans to sponsor
 - ▶ Too much, too soon? Agile is a culture change that requires some adaptation.

KEY DRIVERS FOR AGILE PROCESS

- ▶ ADS team wished to engage stakeholders in the evolution of the product
 - ▶ User interface is web-based
 - ▶ Lends itself to close coordination with users
 - ▶ Bring together stakeholders who rarely interact, yet have significant, inconspicuous interdependencies
 - ▶ Configuration management
 - ▶ End users
- ▶ Other institutional projects were having success with agile methodologies
 - ▶ Plenty of support available (tools and teams)
 - ▶ But how would we remain agile enough yet meet the needs of a plan-driven organization?

ADAPTING AGILE PRINCIPLES IN PLAN-DRIVEN PROCESS

- ▶ Provide high level details to program management sufficient to know what to expect *prior to beginning development*
 - ▶ High-level operations concepts
 - ▶ Early requirements (corollaries to **scrum** 'epics')
- ▶ Once these details are agreed to.....
 - ▶ Proceed with **scrum** cycle as planned, with some exceptions
 - ▶ Large changes of scope are still subject to formal approval (change requests)

THE 'TECHNICAL PLAN'

- ▶ **Work Implementation Plan (WIP)** focuses on detailing the release cycle schedule, workforce allocation, process, and gate reviews
 - ▶ A reduced-scope version of the plan-driven WIP
- ▶ A '**Technical Plan**' (ADS parlance) details technical goals for the release, *without committing to low-level deliverables*. This includes:
 - ▶ Technical drivers
 - ▶ Major new work planned
 - ▶ Usually described in an expository format; may include early technical design artifacts for descriptive effect
 - ▶ Defects to be corrected, if known
 - ▶ Proposed requirements (high-level, traces to new work or defect corrections planned)
 - ▶ **This plan is formally reviewed; approval authorizes the task to proceed**

HOW CHANGES ARE CAPTURED FOR EACH RELEASE

Program Parlance

ADS scrum Parlance

Major deliverables committed to ***prior to*** a release cycle

Draft Requirements

Epics

Major deliverables added ***during*** a release cycle

Change Requests

new Epics

Minor deliverables added ***during*** a release cycle

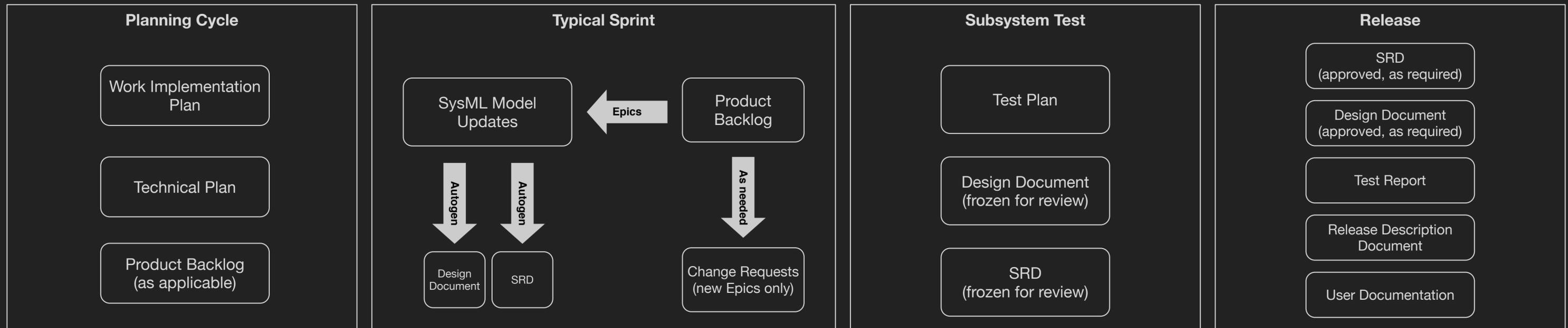
(not required per agreement)

User Stories (*tracing to Epics*)

ENGINEERING ARTIFACTS DURING DEVELOPMENT

- ▶ The existing plan-driven process typically requires the following to be frozen prior to development of a release:
 - ▶ Software Requirements Document (SRD)
 - ▶ Software Design Document (SDD)
- ▶ However, during the ADS *scrum* process, these are incrementally released during each sprint (as appropriate)
 - ▶ Epics as software requirements (export from JIRA)
 - ▶ Design artifacts in SysML (MagicDraw)
 - ▶ SDD is automatically updated/generated based on model inputs
- ▶ Artifacts frozen prior to subsystem test
 - ▶ **These go through normal review and approval process**
 - ▶ *A formality; nothing should come as a surprise to the reviewer if they were keeping up with sprint reviews!!!*

TYPICAL PLANNING AND ENGINEERING ARTIFACTS DURING VERSION LIFECYCLE



RECEPTION

- ▶ The hybrid waterfall/agile approach was well received
 - ▶ The sponsor knew what they were getting from the beginning
 - ▶ *...but acknowledged that the design was iterative....*
 - ▶ The user had input into the development process
 - ▶ Everyone had better insight into task status at any given moment
 - ▶ Developers were much happier!!

SCRUM LESSONS LEARNED

- ▶ Identifying key decision makers (often beyond formal authority) early in the process is very important
- ▶ Always seek compromise!
 - ▶ Agile requires a culture shift, which doesn't happen overnight
 - ▶ Realize that there is no 'one way forward'; much progress can be made with a hybrid approach
- ▶ Retrospective is very valuable
 - ▶ Agile processes often encourage this, however it needs to go beyond just the core team

QUESTIONS?