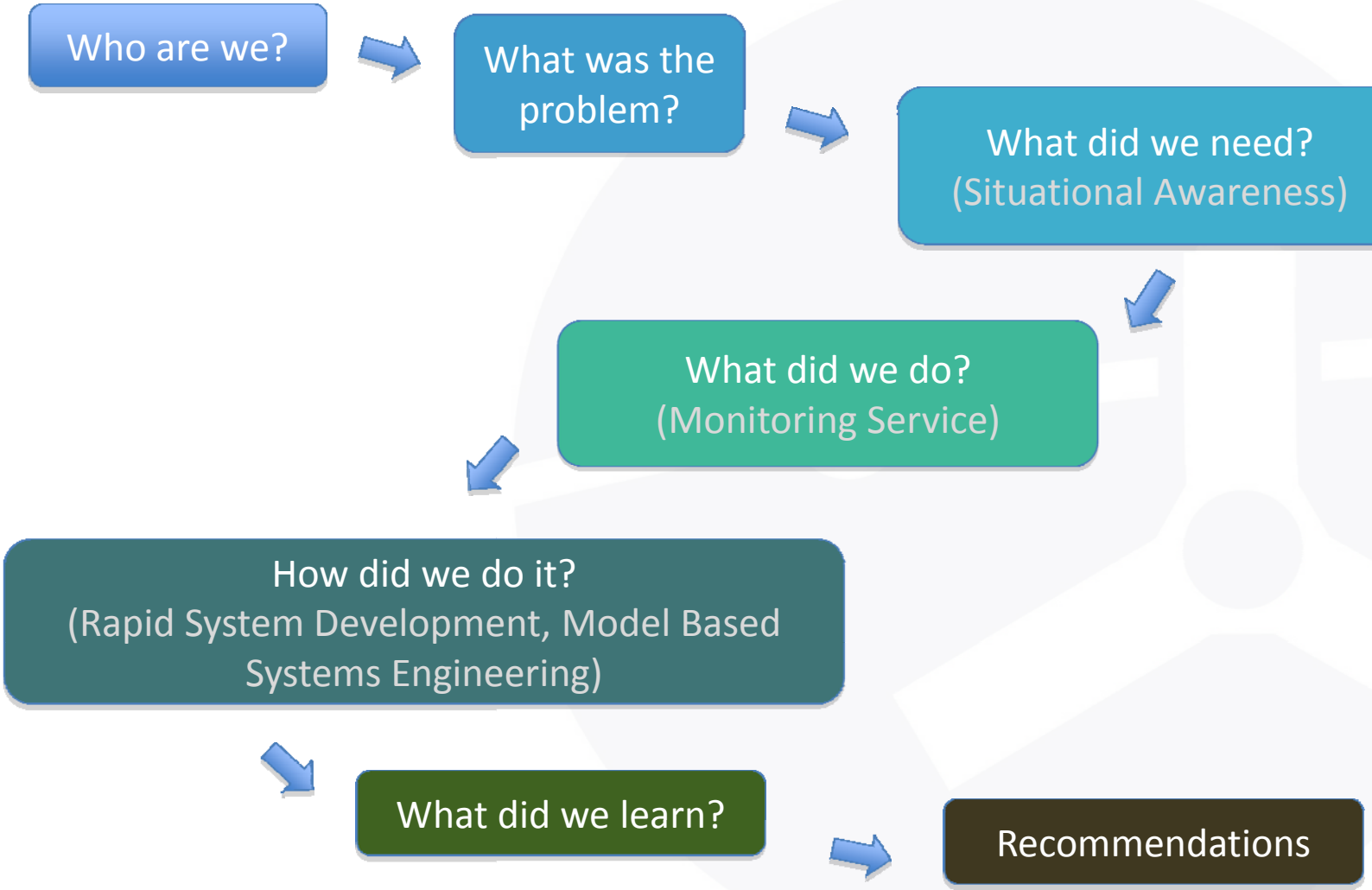
Juno Instrument Data Pipeline Monitoring a.k.a. “Where’s My Data?"

Maddalena Jackson

Marla Thornton

Jet Propulsion Laboratory, California Institute of Technology

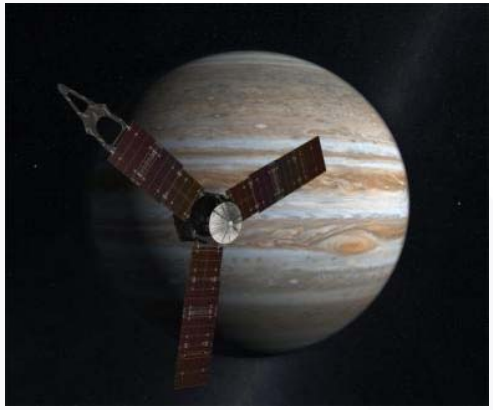
Overview


Who are we?
What was the problem?
What did we need? (Situational Awareness)
What did we do? (Monitoring Service)
How did we do it? (Rapid System Development, Model Based Systems Engineering)
What did we learn?
Recommendations

Mission Background

Salient Features

- *Juno is a Category 1, Class B mission*
- *First solar-powered mission to the outer planets*
- *Eight instrument payload to conduct gravity, magnetic and atmospheric investigations plus an E/PO camera*
- *Polar orbiter spacecraft launches in August 2011*
  - *5 year cruise to Jupiter, JOI in July 2016*
  - *1 year operations, EOM via de-orbit into Jupiter in 2017*
- *Elliptical 11 day orbit swings below radiation belts to minimize radiation exposure*
- *Key Juno partners: SwRI, JPL, ASI, LM-Denver and GSFC*



Instruments

GRAVITY SCIENCE & MAGNETOMETERS (ASC, FGM)

Study Jupiter's deep structure by mapping the planet's gravity field & magnetic field

MICROWAVE RADIOMETER (MWR)

Probe Jupiter's deep atmosphere and measure how much water (and hence oxygen) is there

JEDI, JADE & WAVES

Sample particles, electric fields and radio waves around Jupiter to determine how the magnetic field inside the planet is connected to the atmosphere and magnetosphere – especially the auroras

UVS & JIRAM

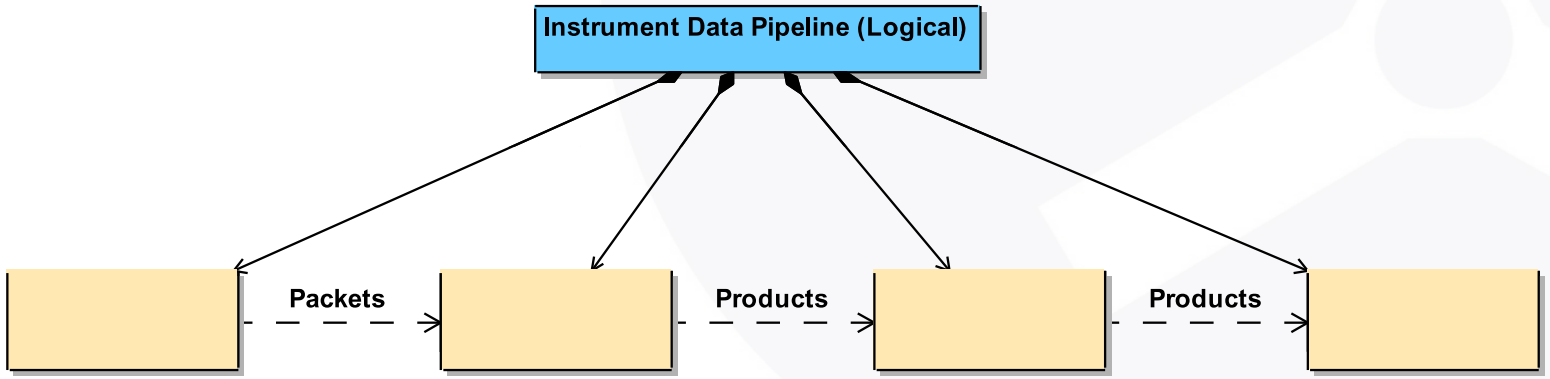
Take images of the atmosphere and auroras, along with the chemical fingerprints of gases there, with ultraviolet & infrared cameras

JUNOCAM

Take spectacular close-up, color images

Background

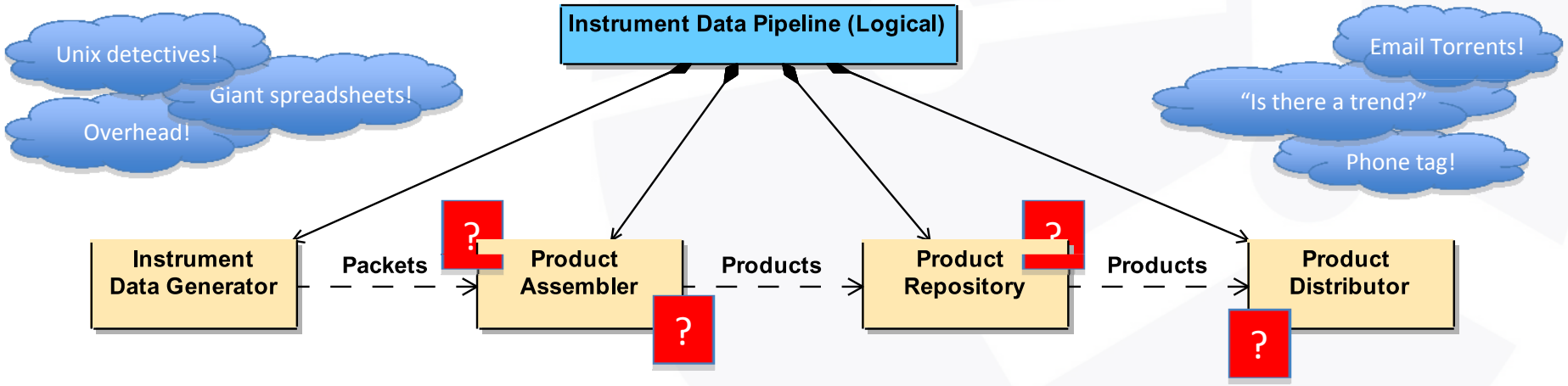
- CCSDS File Delivery Protocol (CFDP)
- Push data to Instrument Teams
- Instrument Data Pipeline
  - From ground receipt to delivery to Instrument Teams
  - Multi-mission Components
  - Mission Data Delivery Latency Requirements
- Testing revealed:
  - Mission Data Delivery Latency Requirements not met



Problem

- Great until there's a problem
  - No visibility
  - Different results! (Testbeds)
  - Is this going to happen again?
  - How do I know when something's wrong?

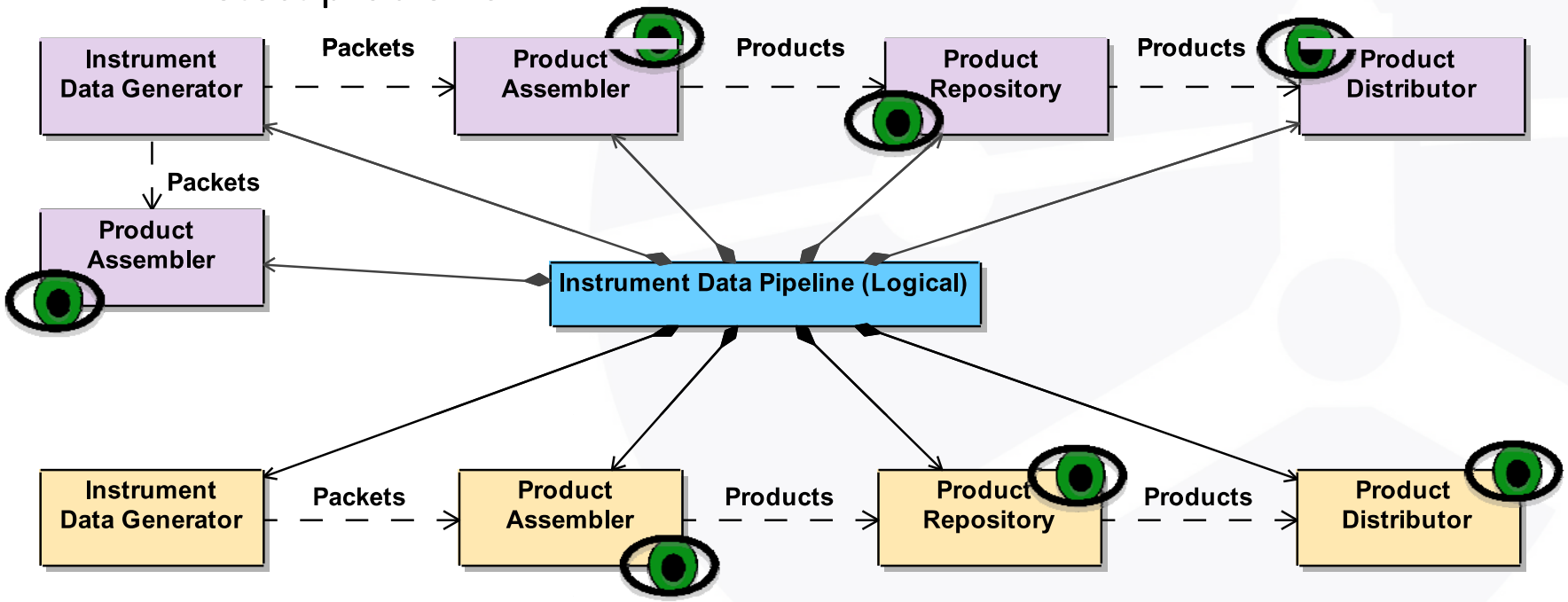
Where is the data supposed to be?
Where is it really?
Why isn't it there?
When was it supposed to be there?



We need to know status in real-time! (Situational Awareness)

Situational Awareness

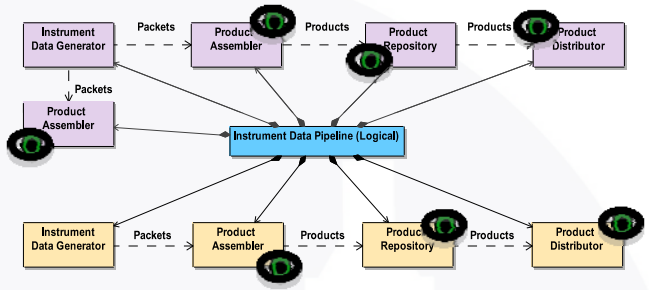
- Need monitoring (eyes) at every point in the pipeline
  - Multiple pipelines (testbed vs. ops)
  - Analyze performance (throughput, latency)
  - Compare products
  - Detect problems



- Let's automate this and do it before we need it!

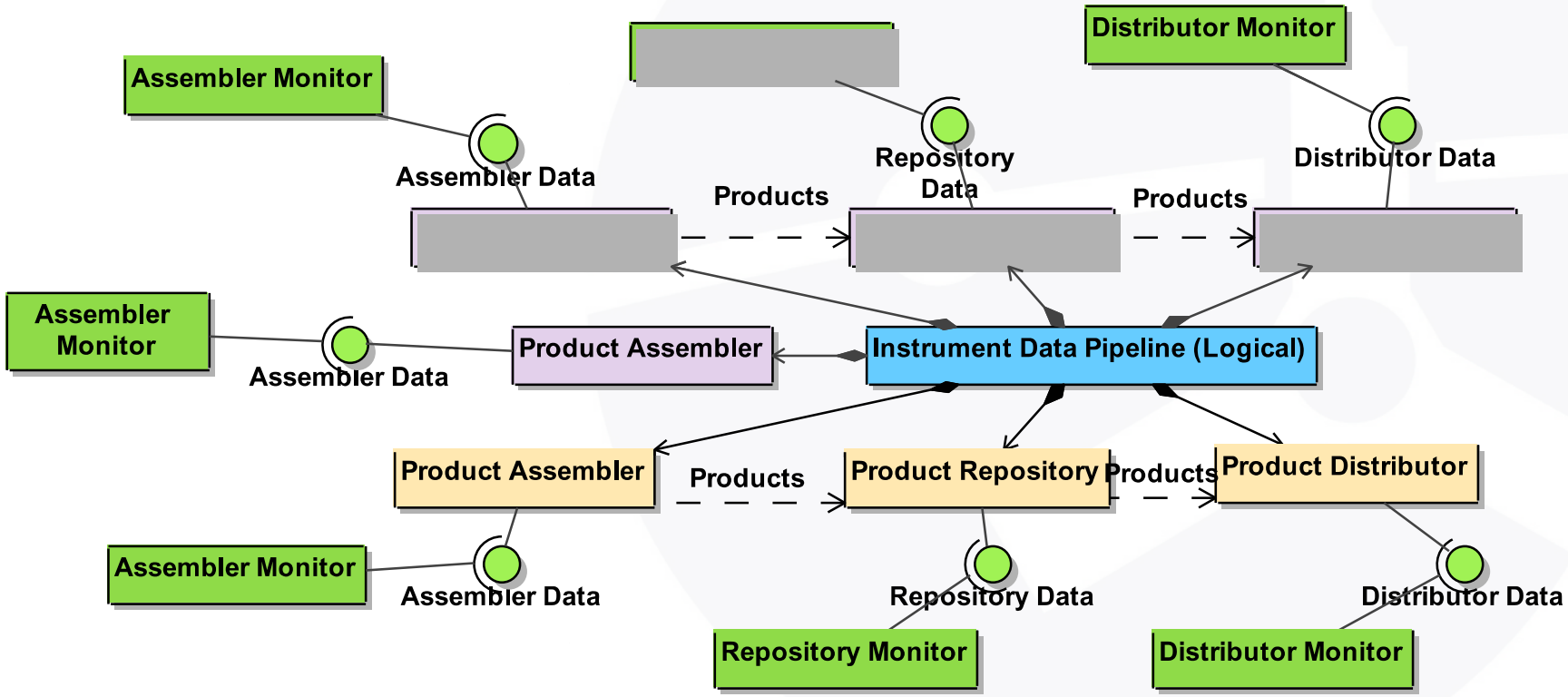
Monitoring Service

- What do need the service to do?
  - Function: Collect data
    - Modular (one monitor per IDP node)
    - Distributed (runs at the node)
    - Read-only (no effect on IDP)
  - Function: Display data and analysis
    - Display by current test, instrument, IDP component
    - Remote access (website)
    - Automated performance analysis
- Implies…
  - Standard agent-web-server interface
  - Automation (point is to save time)
  - Extensible (transition from ATLO to Operations)
  - Test-Like-You-Fly (TLYF)

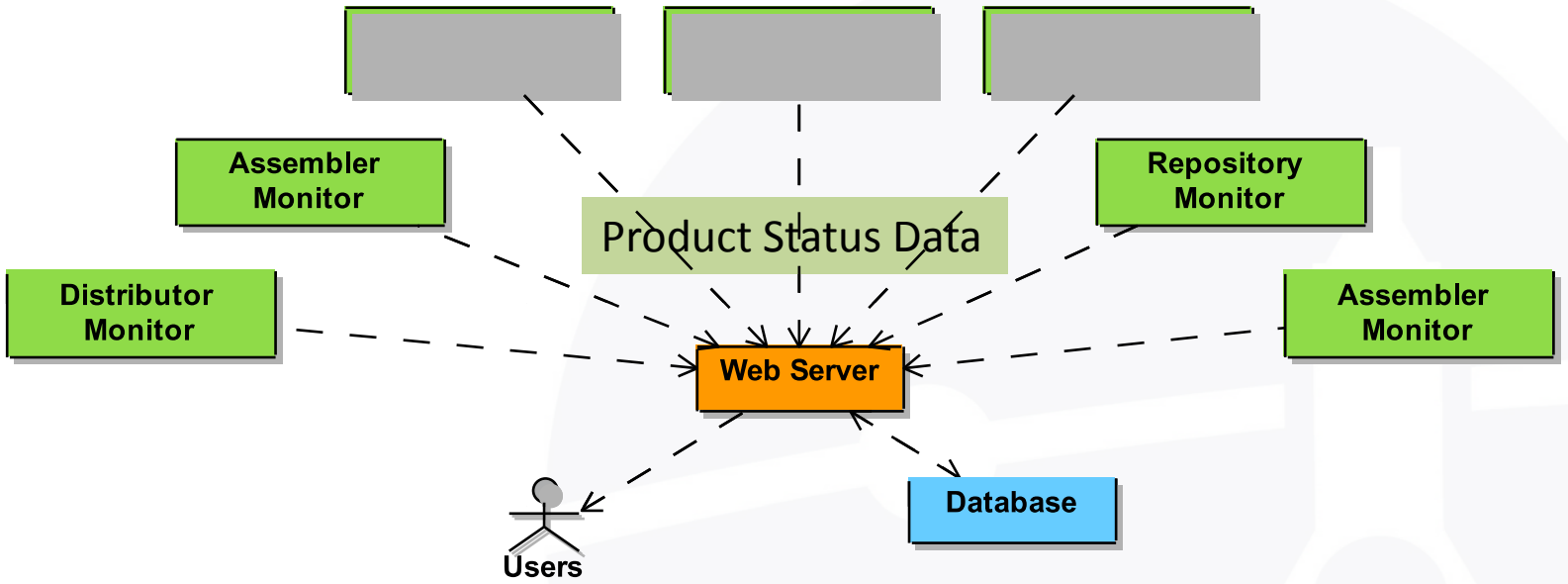


Monitoring Parameters

- What should the software monitor?
  - Size, name, arrival time, completion time
  - Provides knowledge of location, enables comparison, and determination of bottlenecks



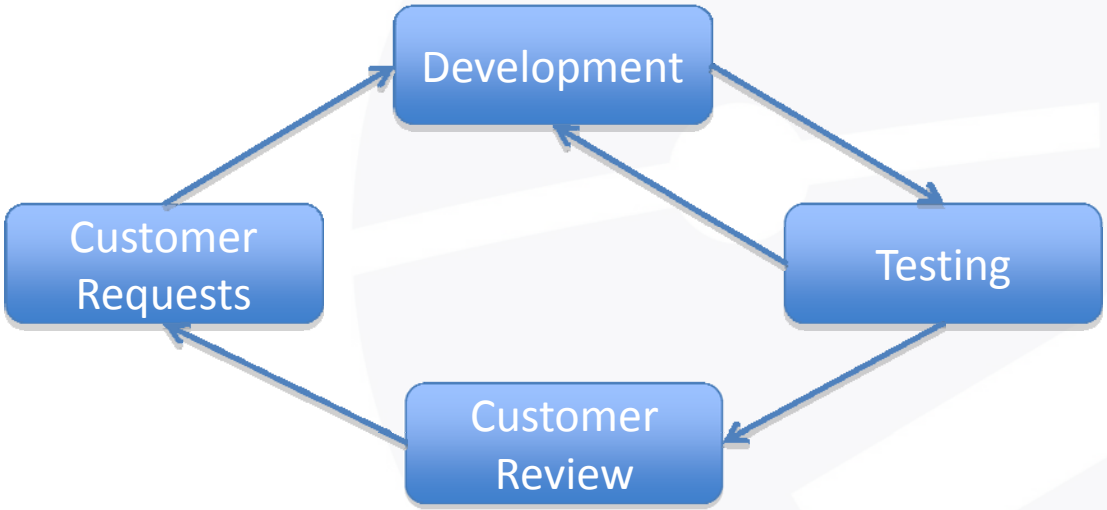
Services



- Plug-and-play monitoring “agents”
- Standard HTTP interface
- Open-source web server / database framework
- Register of products in real-time
- Accessible to team members from any location
- Shows latest location of products and product data
- Shows latencies and bottlenecks

Rapid System Development

- Needed functional system ASAP
  - Ready to support Juno Thermal-Vacuum Testing
- Rapid development cycle



- Team:
  - Senior Programmer (Developer, Tester)
  - Mission Expert (Developer, Tester)
  - Mission Personnel (Users & Customers)

Impact of Rapid System Development

- Knowledge Transfer
  - Software experience ←→ Mission experience
- Test-Like-You-Fly
  - Constant testing in operations environment
- Quick customer feedback
  - Iteration, refining needs -> different software design
  - Refactoring, re-design induces some overhead
- Agile development
  - Final product highly optimized for customer
  - Accommodates evolving knowledge of pipeline
  - Reverse-engineer some requirements, test cases, and documentation
    - No time for full ‘waterfall’ systems engineering process at each iteration

Product delivered – what next?

- Institution requires capture of:
  - Requirements
  - Design
  - Implementation
  - Testing
- 5 year cruise – information ‘archive’
  - Who will operate it when we get to Jupiter?
- Model-based Systems Engineering (in SysML)
  - Standard views
  - Supports description/design of those factors
  - Visualize dependencies
  - Gold-source / centralized SE capture
  - Can be used for future design

MBSE - Process

- Requirements - spreadsheet
- Captured components
- Captured component logical/functional design
  - Discovered areas of weakness
  - Discovered redundancy
  - Discovered inconsistencies
- Captured Requirements in Model
  - Analyzed discrepancies between requirements and design
  - Improved requirements
- Captured implementation
  - Link to requirements
- Captured testing
  - Analyze testing completeness
  - Link to requirements
  - Link to functions
- Generated Artifacts

Design Document
Test Plans
Operating Procedures
Implementation Descriptions
Requirements Documents

Conclusions

Situational Awareness
- Tedious by hand
- Easy with software

Service Orientation
- Simple (for distributed monitoring)
- Flexible
- Extensible
- Robust (modular)

Rapid System Development
- Knowledge Transfer
- Leverage open-source frameworks and libraries
- TLYF

MBSE
- Standard artifacts
- Central gold-source of systems engineering
- Reveals weaknesses
- Analysis of Systems Engineering dependencies

Recommendations

- Situational Awareness
  - Analyze current (tedious) processes →Define metrics
- Service Orientation
  - Standardize external interfaces (with GDS systems)
- Rapid System Development
  - Clear roles/expertise
  - Codify development/delivery cycle
  - Integrate functional design
  - Document all feature/functionality requests
  - Invest time in documenting use-cases /concept of operations
- MBSE
  - Involve earlier in the process (functional)
  - Capture requirements and rationale in real-time
  - Keep an up-to-date capability / function / subsystem mapping
  - Document range of TLYF events as test cases
  - Create/maintain procedures in real-time

Backup Slides

“Solution”

- How do we do it by hand?
  - Unix Detectives
    - Can we find the last known location of the file?
  - Call people if we don’t have access to parts of the pipeline
    - Did you see this product go through?
  - Write things in a notebook
    - Product ID, arrival time, file size, build time, etc.
  - Email (add to the torrent)
    - Here’s a report of the problem, file it away somewhere!
  - Get vague impressions over time
    - I think it gets slower when this instrument is turned on… hmm!
  - TEDIUM
    - Put things in a giant spreadsheet and try to find a trend?
    - Repeat the time-consuming process