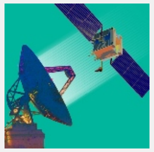
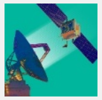


GSAW 2020
March 2-5, 2020
Renaissance Los Angeles Airport Hotel
Session 3: Development Methodologies

Is Structured Agile an Oxymoron?

Tales from Implementing and Executing Agile in a US Government Environment

Theresa Beech and Sharon Orsborne, NASA Goddard Space Flight Center, Software Engineering Division
Jay Bugenhagen, Arctic Slope Technical Services
Jay Czarnecki, Telophase Corporation

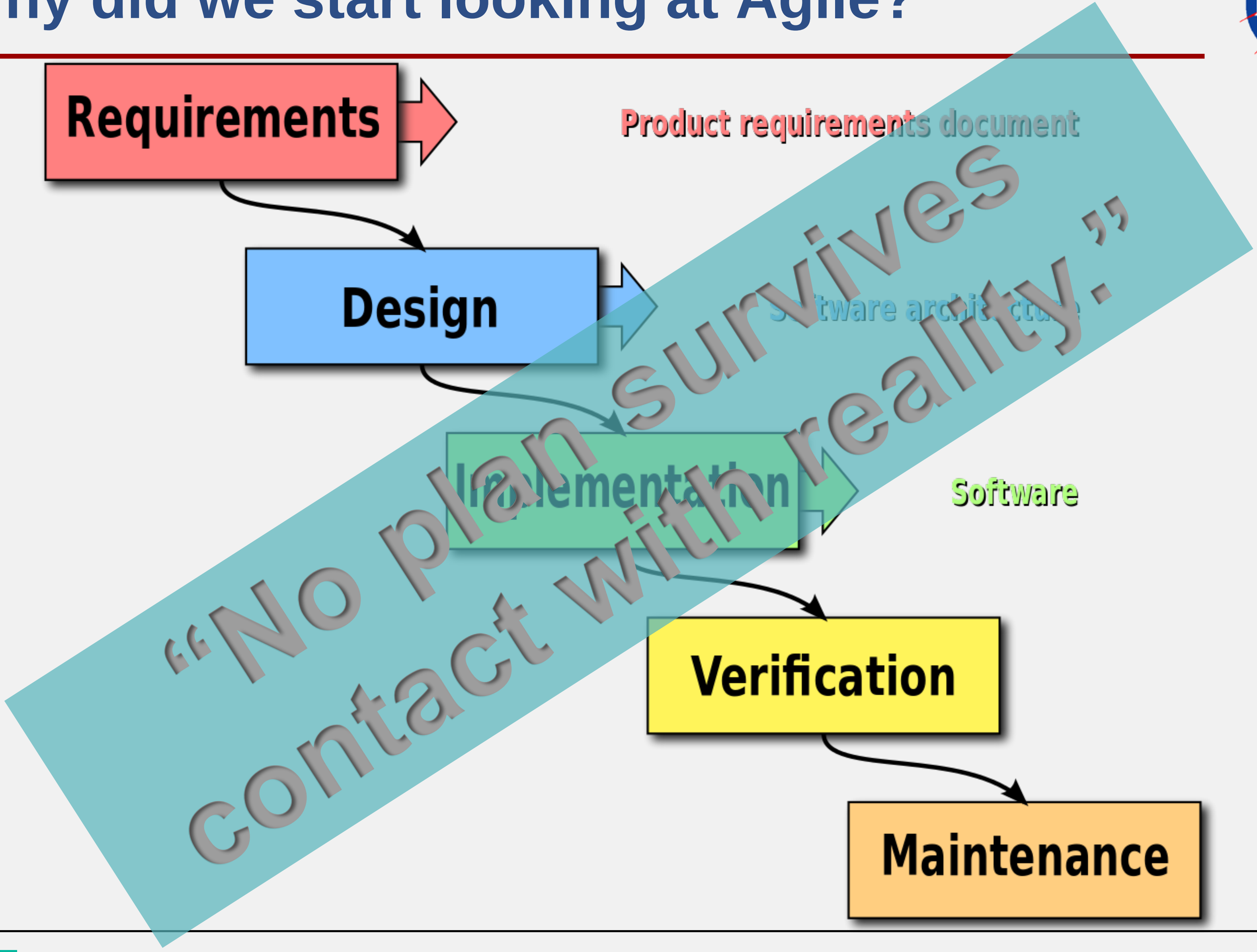


Overview of our development

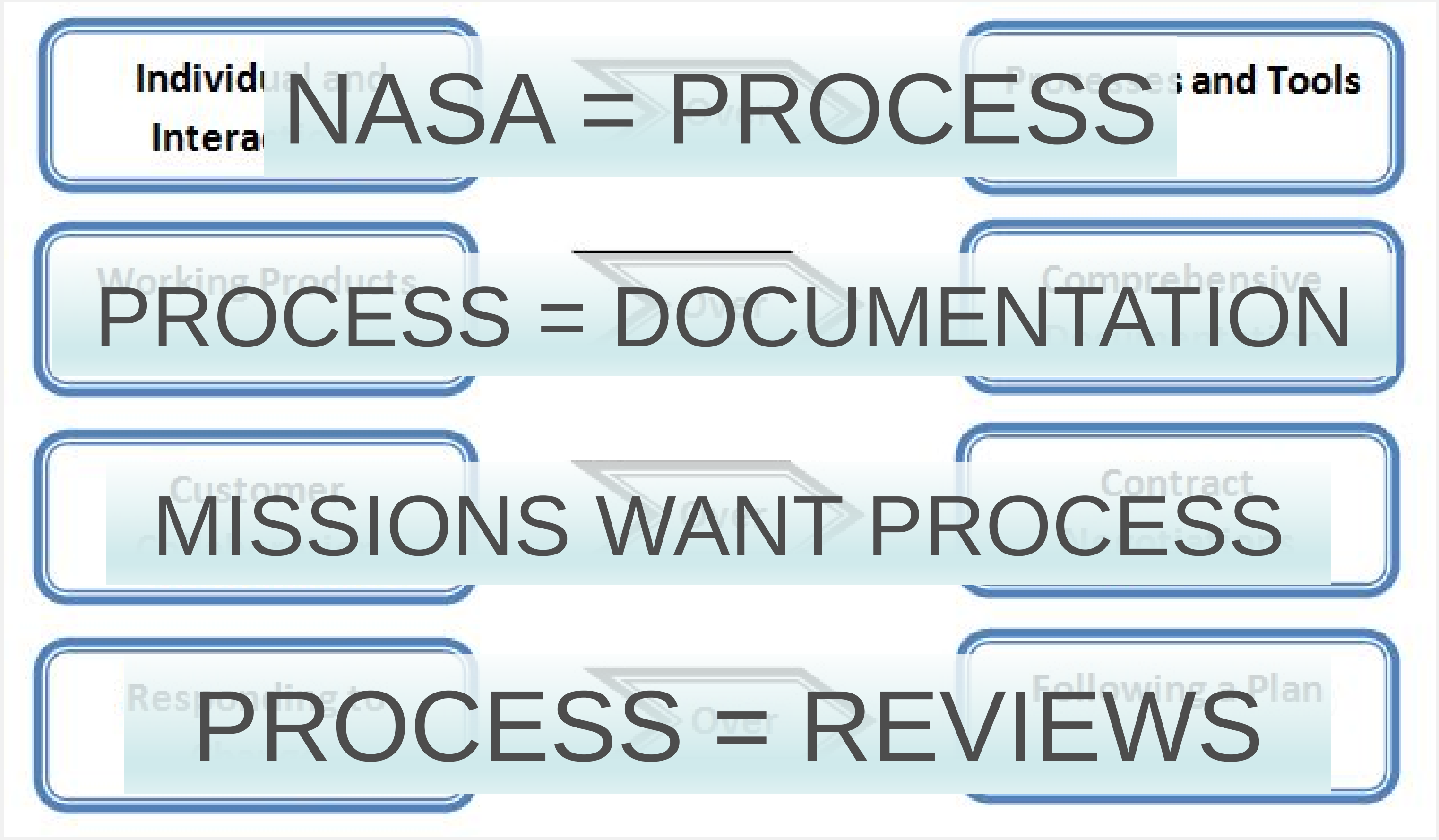
- About 16/17 FTEs (~ 20 people) develop satellite ground system SW
  - Mix of civil servants and contractors
- Support dozens of satellite missions across multiple space agencies (NASA, NOAA, DoD)
- Operational since 2005
- SW is a broad spectrum of:
  - Older SW in maintenance & sustainment mode
  - SW with on-going enhancements
  - New SW mostly well-bounded
  - New SW of unknown scope/implementation
  - Class B (mission critical), C and D software
- OMG C2MS standard compliant (C2MS evolved from our message standards)
- On-going mission support
- Software technologies include:
  - C, C++, C#, Java, Python, webservices, …
  - SQL, Elasticsearch
  - Multiple middlewares: ActiveMQ, RabbitMQ, Webspheres, OpenDDS, internal (Bolt)

GMSEC's mini-MOC: Big Bertha

Why did we start looking at Agile?



The Agile Manifesto & Our Problem Space



NASA = PROCESS

PROCESS = DOCUMENTATION

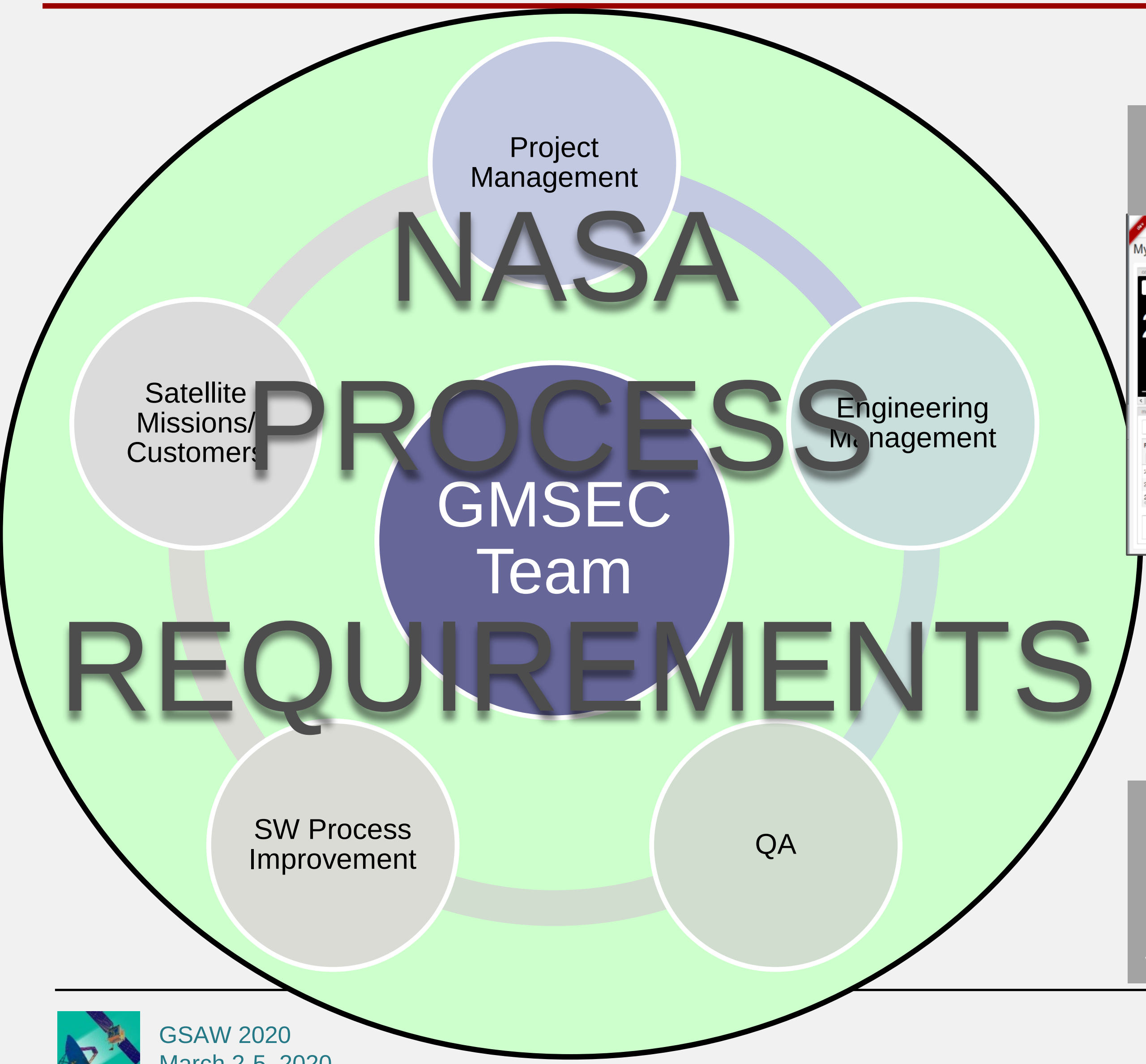
MISSIONS WANT PROCESS

PROCESS = REVIEWS

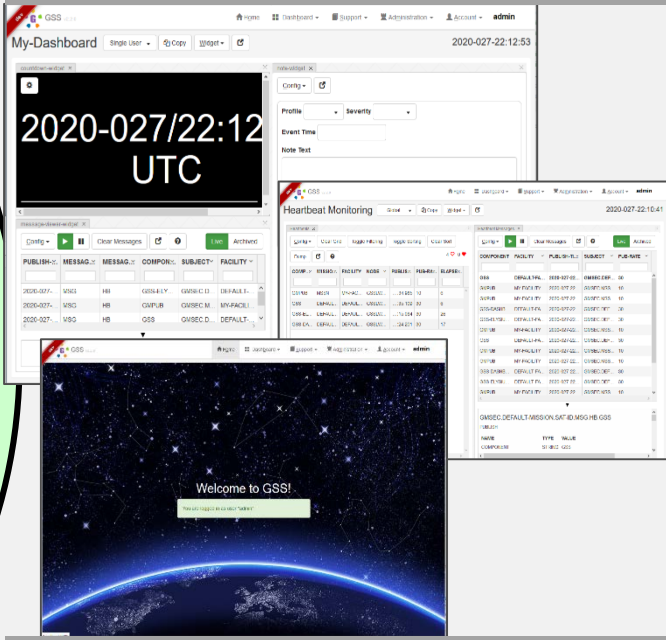
Sounds great, but how do we do this at NASA?

How did we start?

Our environment & stakeholders….



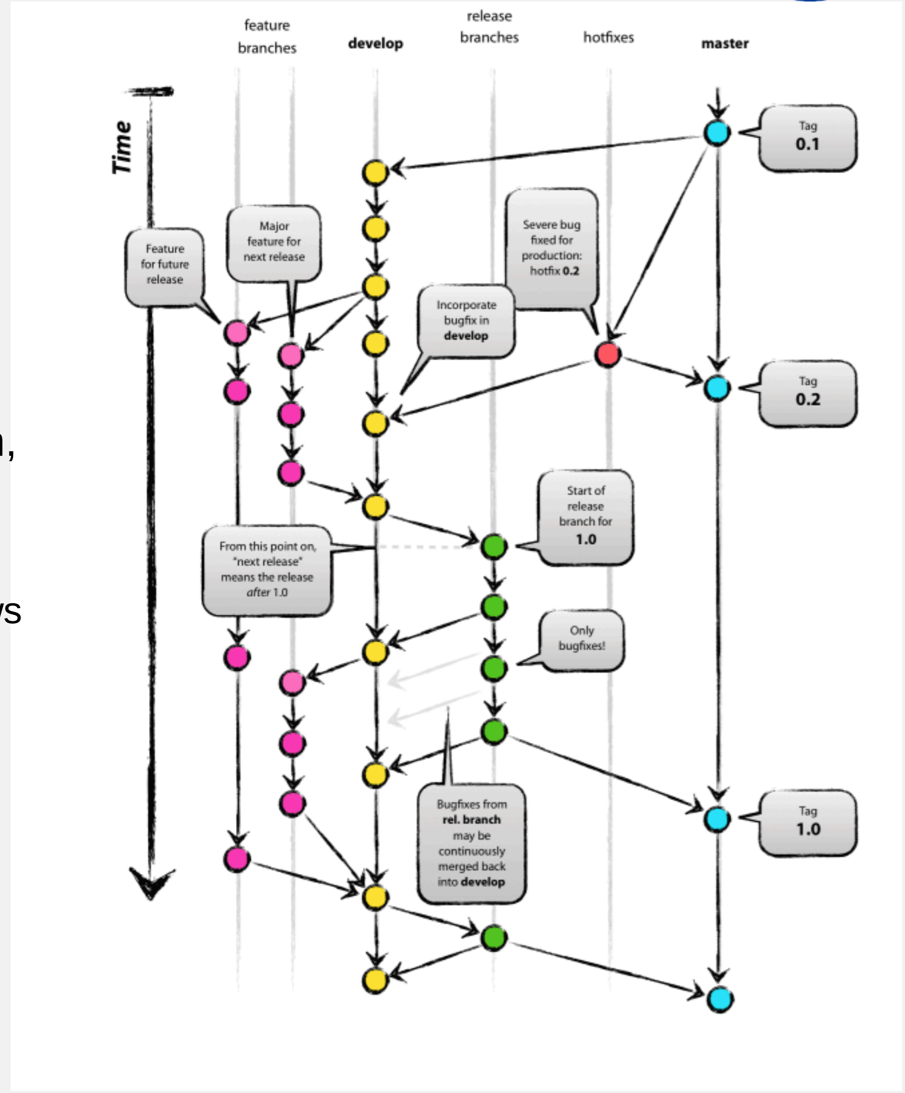
2015/6: Agile prototype of new component GSS



2018: Extended Agile to all GMSEC software with mods to meet NASA process reqs → ***STRUCTURED AGILE***

What have we done?

- **Structured Agile**
  - Team sub-divided into smaller teams
  - Team meetings in person + telecons
    - Monthly sprints
    - Daily stand-ups (max 15-20 mins)
    - Weekly Engineering Peer Reviews (EPR)
    - Ad hoc meetings as needed
  - **Frequent communication** with SPI team, management and missions/customers
  - **Reduced formal reviews > 50%**
    - Mthly sprint mtgs substitute for some reviews
  - **Automate, automate, automate:**
    - Component testing **reduced from 15 manmonths to 1 manmonth** per release
  - **Minimize manual documentation:**
    - **> 2/3 generated automatically**
    - Standardized component test plan
  - Independent QA audits unchanged


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

How to tailor Agile to be more structured

- **Continuous team discussion on processes:** everyone participates
  - Process discussions occur monthly to figure out what is working, what isn't, and what we need to do to meet the NASA requirements
- **Close working relationship with the SW Process Improvement (SPI) Team** to understand the NASA reqs, and tailor our processes to meet it
  - Invited to all meetings except stand-ups
  - **3 reviews:** RCR, TRR and RRR
  - **Framework/tools maintain/house our QA artifact repository**
    - SW Reqs → SW tests RTM, Test Procs, & Test Reports in Robot
    - DRs and Ers logged in Jira ➔ auto-generate VDDs and readmes
    - Code review process tracked in Git at merge requests
    - Jenkins used for continuous integration testing nightly, Gradle is our build environment, Izpack is installer
    - ***Complicated to get QA buy-in***

EXAMPLE in progress: bidirectional trace

SW Reqs ⟺ SW Design ⟺ Code

- **Waterfall answer = more documentation, *lots* of documentation**
  - Not acceptable, so how to do something Agile?
- **On-going team discussions**
    - Can we ignore this? What do we want to do? From nothing to …
    - How to make it useful for us? Old vs. new developments
    - How to minimize the "check the box" cost?
    - **Heated discussions with SPI listening in**
    - Informal briefings to eng mgmt
  - Approach decided upon:
    - **Two pilot projects started:**
      - MagicDraw linked to Jira and Git
      - Javadoc and javacomments

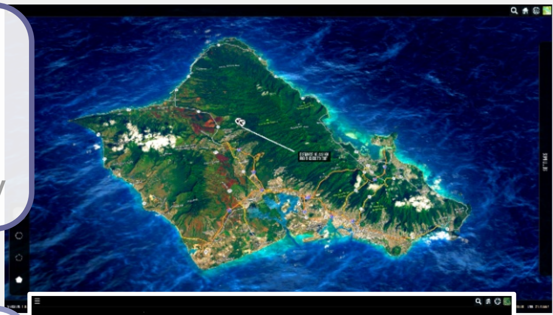
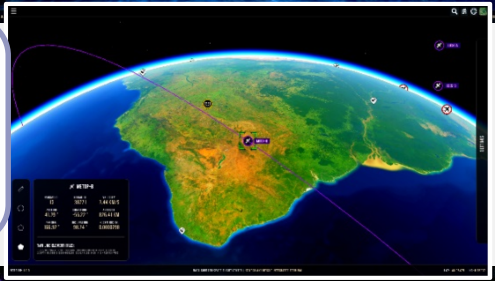
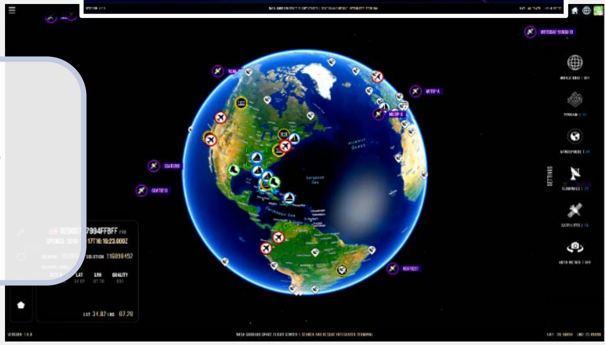
Mission EXAMPLE: Search and Rescue (SAR) Intelligent Terminal (SAINT)

- Objective: develop a prototype for a distributed beacon tracking visualization system for use during human launch & landing

Challenges
- New customer, new application, new mission type
- Unclear requirements
- Complex database with unhelpful documentation
- Customer not used to custom SW development, Agile dev

Approach
- Agile prototype starting with mapping beacons
- Re-engineering of database
- Weekly customer meetings, with periodic management meetings including sprint demos

Current status
- Expansion of prototype to full operational development
- Support of database passed to team

What is our Structured Agile outcome?

Structured Agile

- More robust SW testing
- Better team communication & cohesion
- Good customer feedback
- Increased SW development tempo
- NASA Process Policy Compliance
- More Team Lead headaches….

More: Robust SW, SW drops, SW functionality
Better: Team environment
+ NASA Process Policy Compliance!!!

Lessons Learned

GUIDING PRINCIPLE: The process must help and not hinder the team

COMMUNICATE

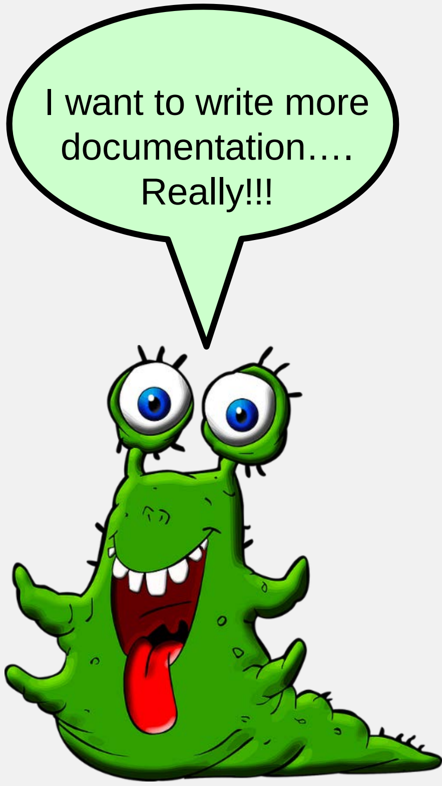
- **Talk through all issues, then talk more.**
- Environment of trust & emotional safety is essential for productive conflict
- Continuous stakeholder communication, again and again and again

AUTOMATE

- **Pick your tools well**
- Continuous automated testing & notification
- Minimize manual document generation
- Link tools (Jira → Git, Git → Robot → Jenkins → Gradle → IzPack)

RESPOND

- **Technical excellence at all levels** → autonomy to respond as needed
- Flat team of SW dev + sys eng
- Quick response to mission/customer needs
- Use sprint outputs for customer feedback



…said no engineer ever…