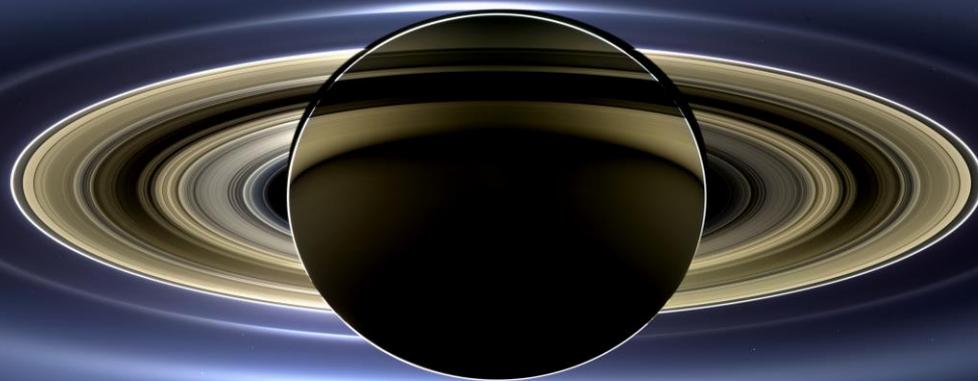




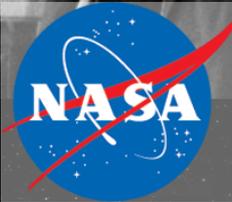
Jet Propulsion Laboratory
California Institute of Technology



Testing Through Time and Space: NASA's Twenty-Year Mission to Saturn

Presented by Andrea Connell, Software Engineer - Jet Propulsion Laboratory, California Institute of Technology

August 17, 2017



Jet Propulsion Laboratory
California Institute of Technology



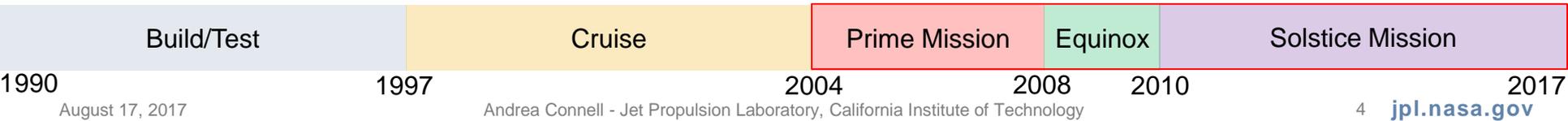


Jet Propulsion Laboratory
California Institute of Technology

Timeline



Photo Credit: NASA/JPL-Caltech



Timeline



Photo Credit: NASA



Build/Test

Cruise

Prime Mission

Equinox

Solstice Mission

1990

1997

2004

2008

2010

2017

August 17, 2017

Andrea Connell - Jet Propulsion Laboratory, California Institute of Technology

5 jpl.nasa.gov

Timeline

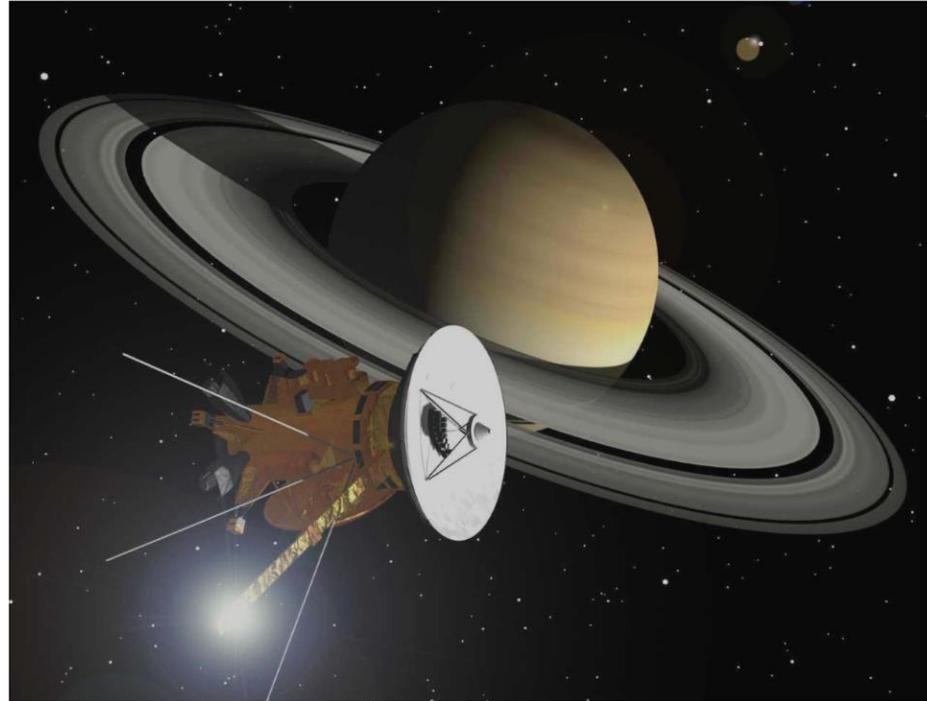


Photo Credit: NASA/JPL-Caltech/Space Science Institute



Build/Test



Cruise

Prime Mission

Equinox

Solstice Mission

1990

August 17, 2017

1997

2004

2008

2010

2017

Timeline

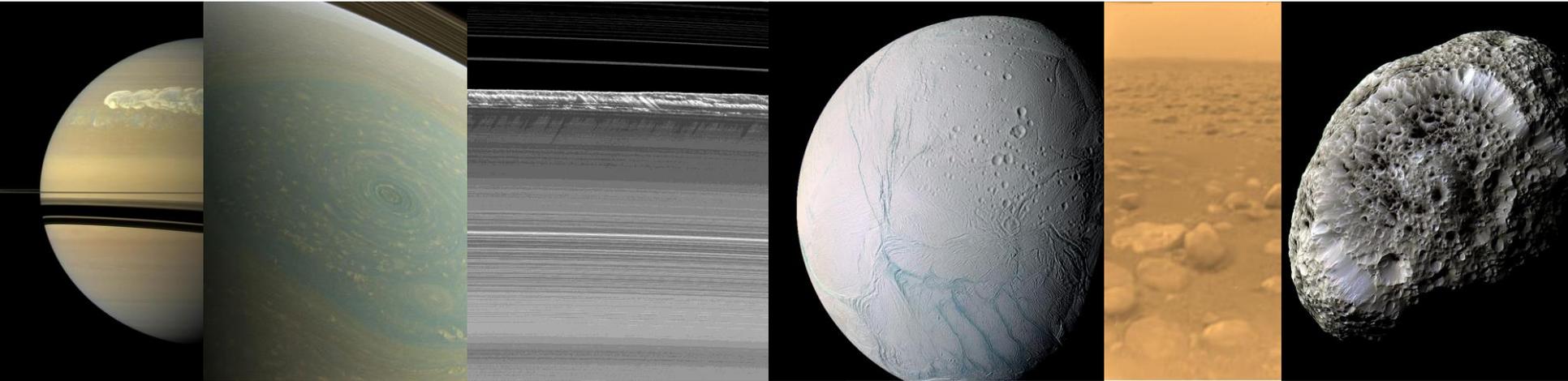


Photo Credits: NASA/JPL-Caltech/Space Science Institute, ESA/NASA/JPL/University of Arizona



Timeline

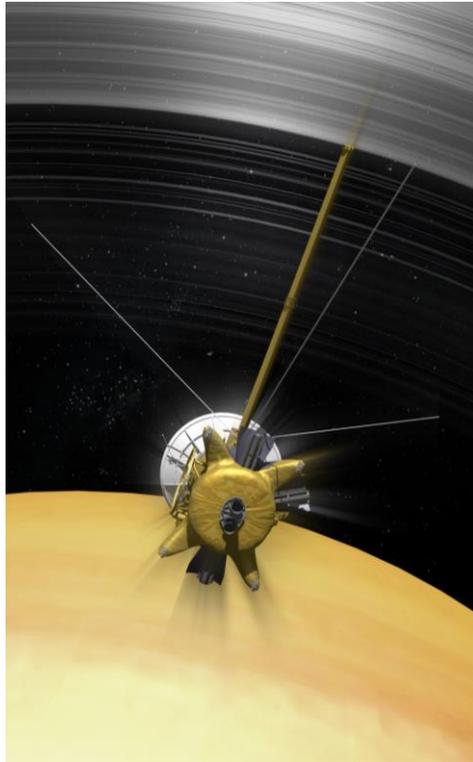


Photo Credit: NASA Jet Propulsion Laboratory



Timeline

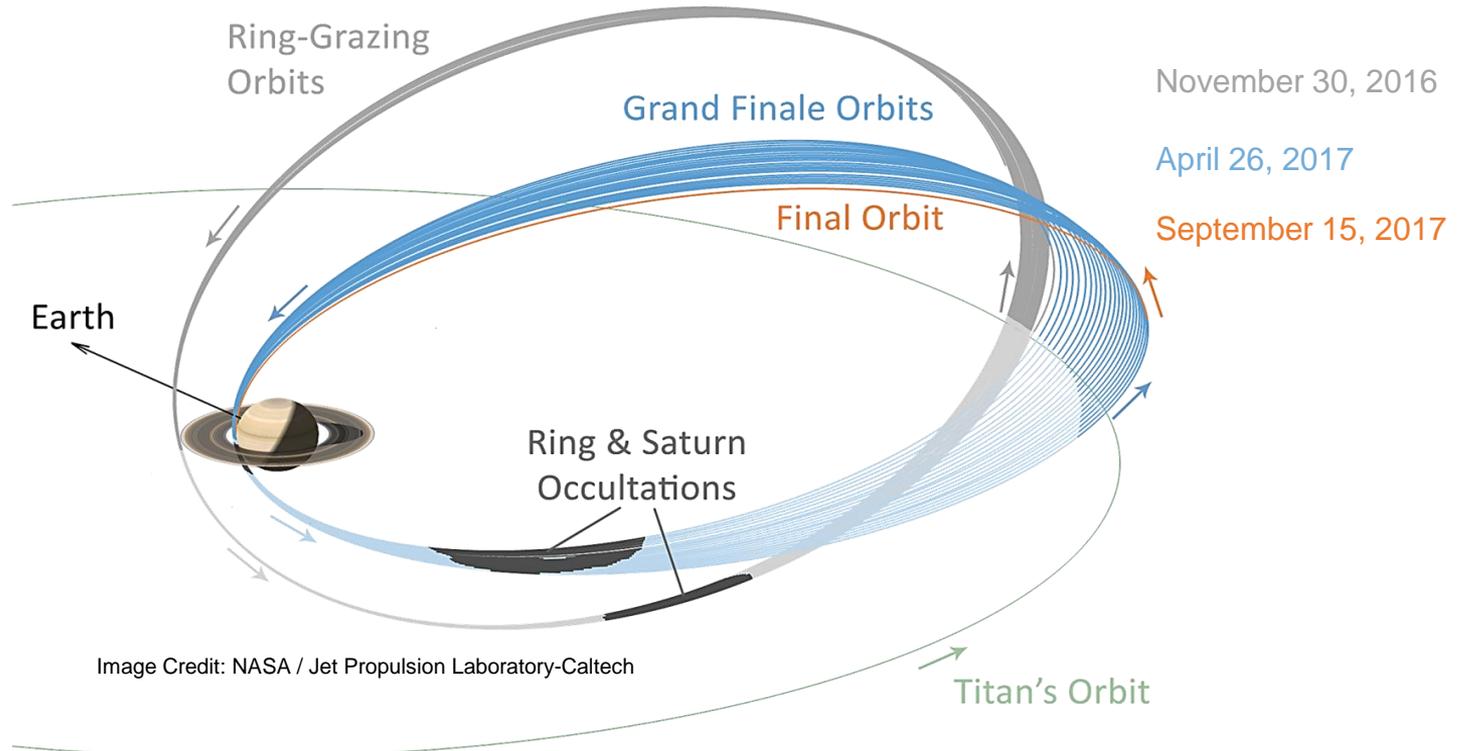
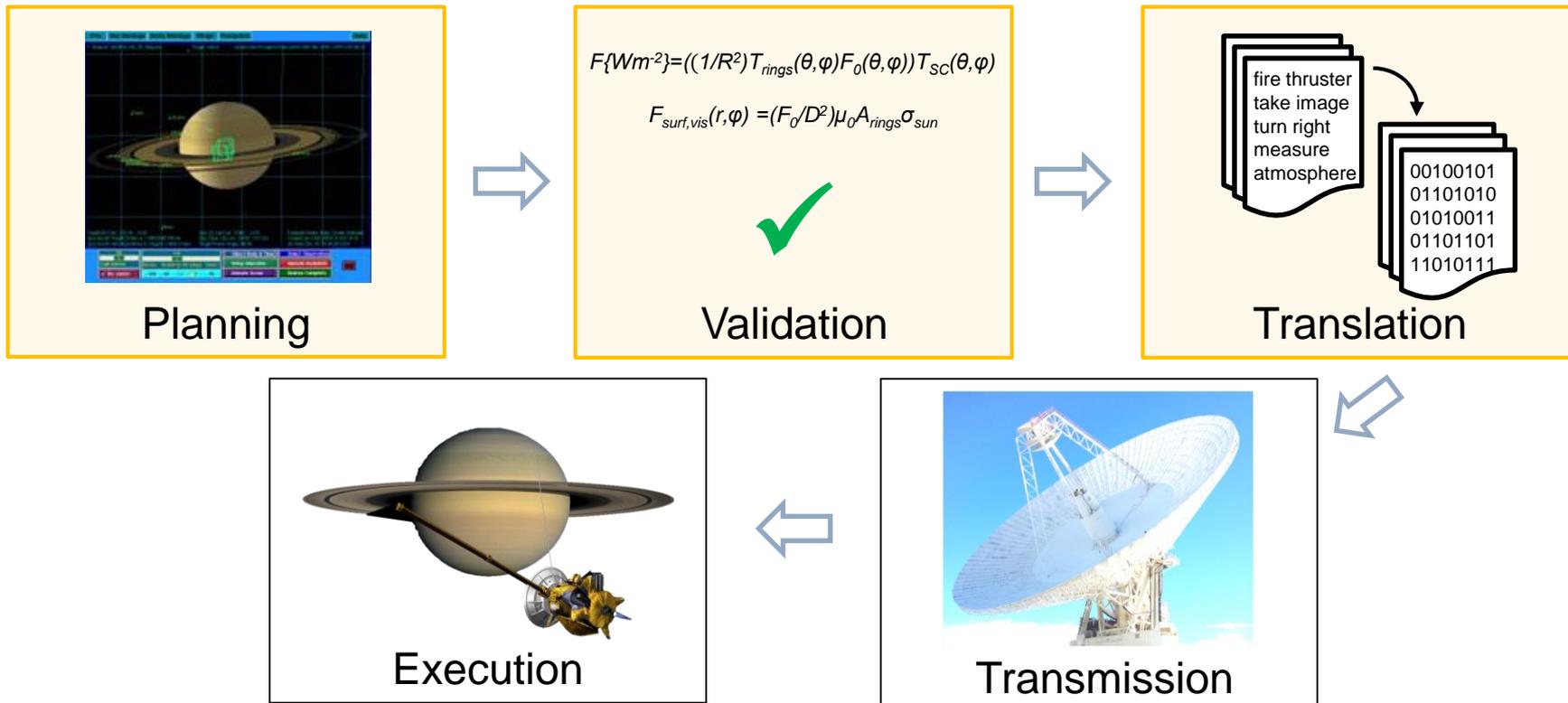


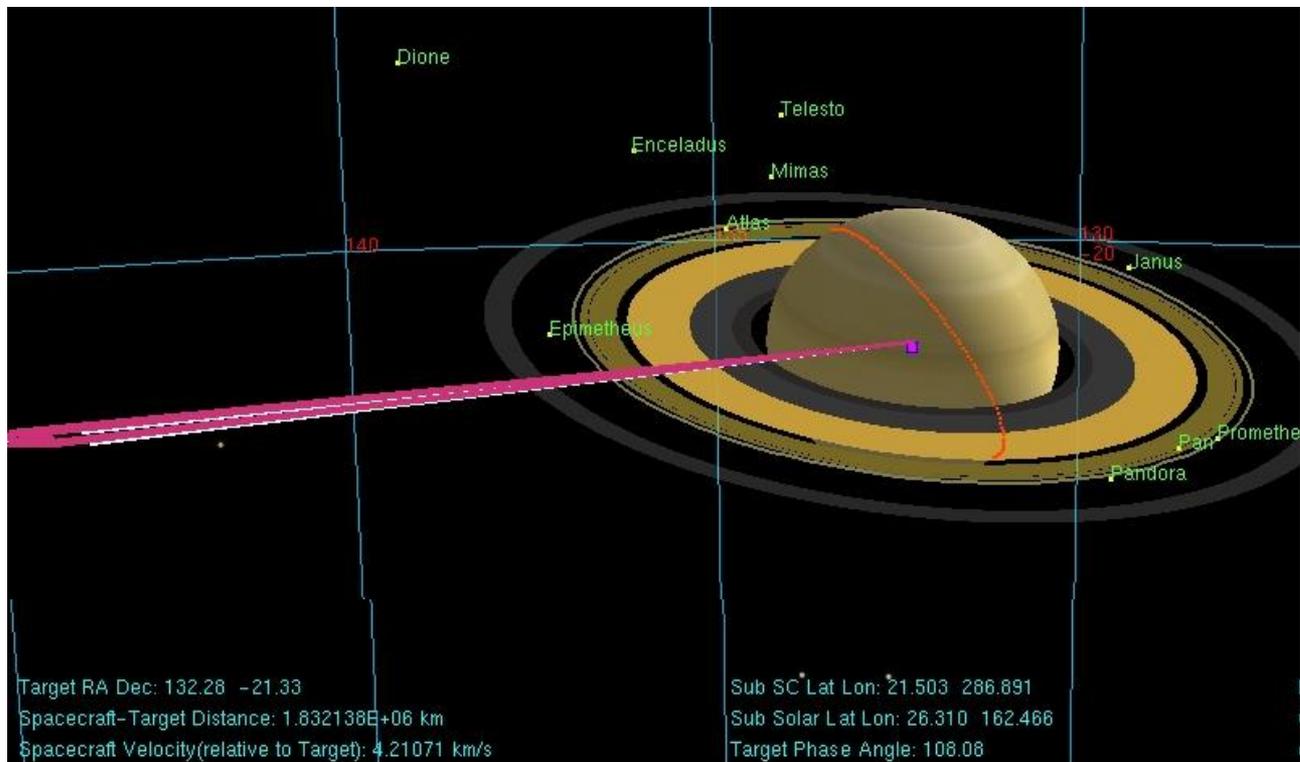
Image Credit: NASA / Jet Propulsion Laboratory-Caltech



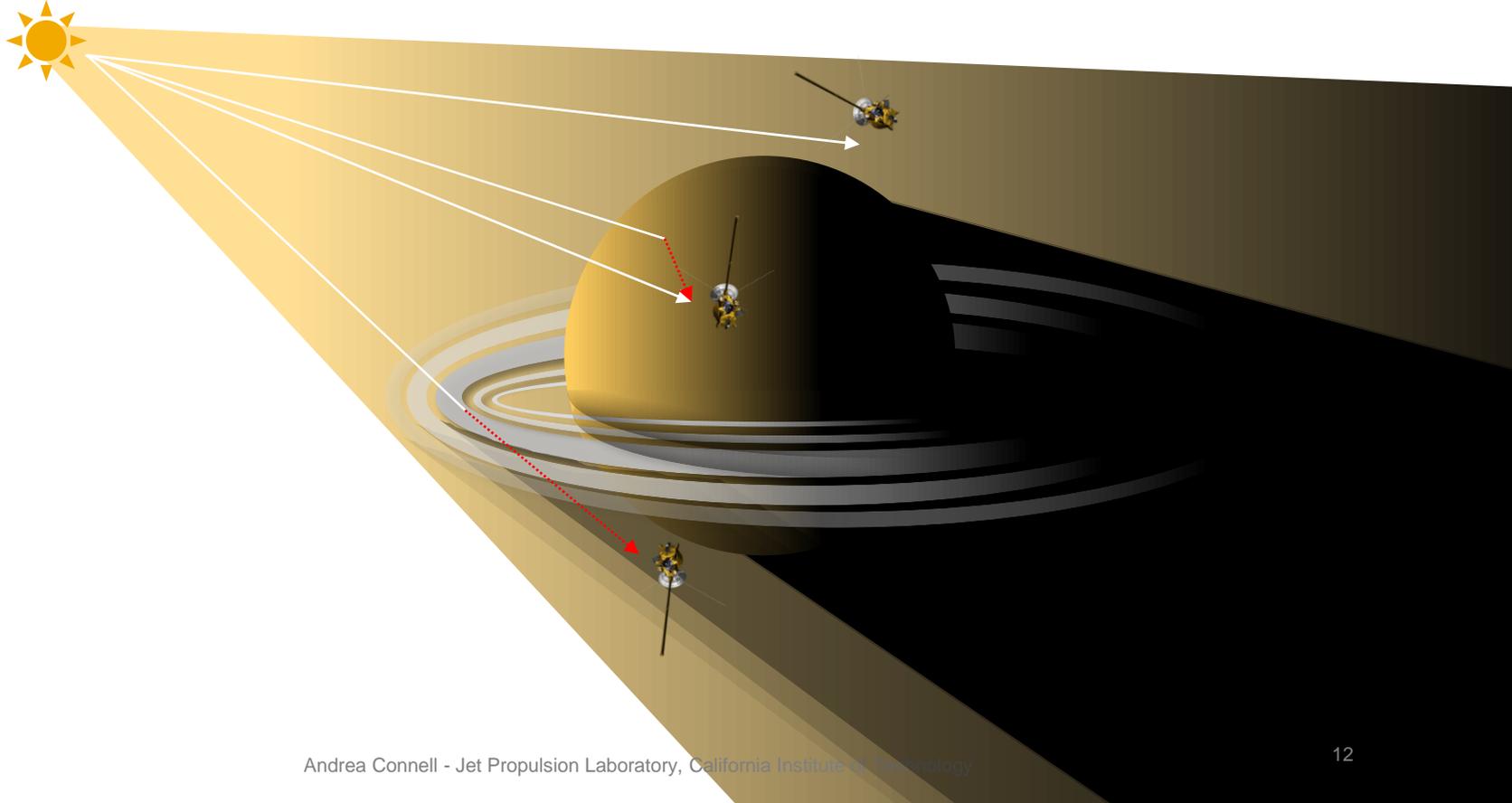
Mission Sequencing Subsystem (MSS)



Pointing and Targeting



Thermal Modeling



Challenges

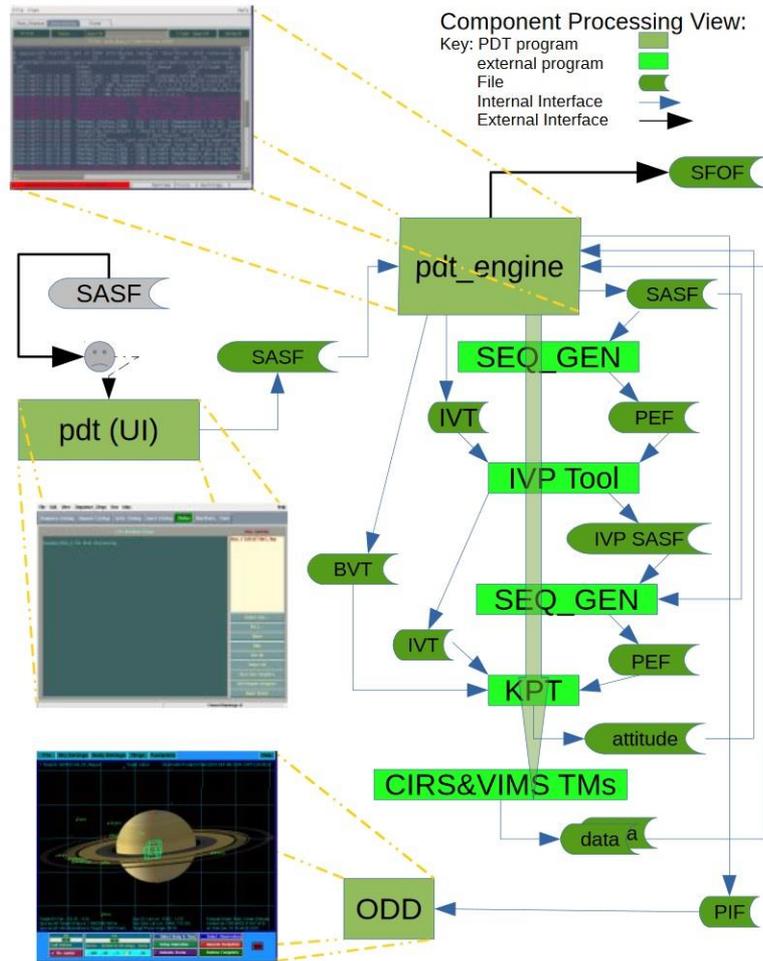
- Legacy software, limited funding, risk-adverse
- Big changes needed for Grand Finale
- Complex algorithms and models
- Randomness in output
- No GUI testing framework
- Reporting requirements

Collaboration with Team

- Previously, team had little collaboration
- Testing and integration took a long time
- Before Grand Finale changes, decided on new approach
- Scrum improved teamwork and transparency
- Developers and testers worked together
- Time to delivery decreased

Components

- Legacy software architecture
- Many components that input and output files
- Cannot execute individual functions
- Mix of Perl, C, C++, and domain specific languages



System and Integration Tests

- Components not designed for unit tests
- Refactoring software adds additional cost and risk
- We automated system and integration tests instead
- Given a specific input file, check the output file

Test Harness

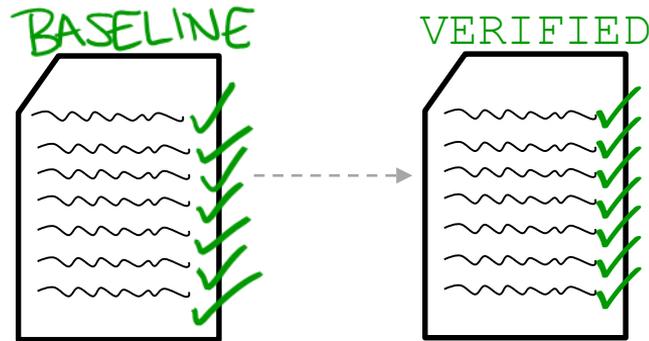
- Created in-house test framework
- Test configuration in json files
- Python and Bash scripts
 - Prepare environment and expected values
 - Execute components
 - Verify output

Test Environments

- System tests created on development machines
- Integration environments required change approval to ensure match to production environments
- Test suite executed across a variety of integration environments
- After deployment, a subset of smoke tests executed

Verification Process

- Subject matter expert inspected all output to verify
- Future executions were compared to baseline
- Able to better characterize behavior of software
- Able to better understand impact of new changes



Handling Randomness

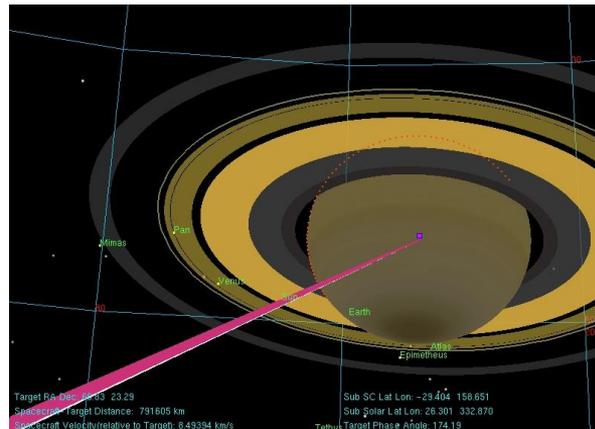
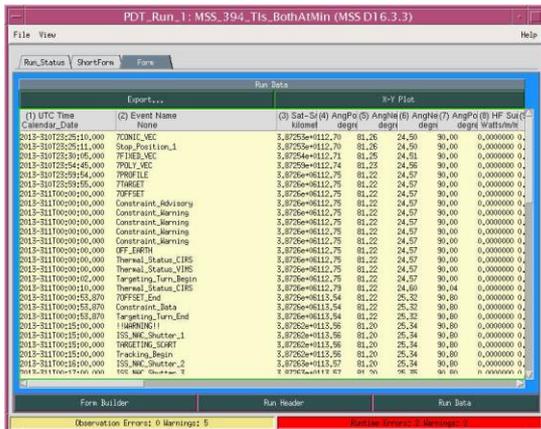
- Monte Carlo simulation generated randomness
- Amount of difference varied with geometry
- Configured acceptable tolerances by value per test
 - Percentage or exact amount
 - For example, temperatures could be within .25K

Outcomes

- Much better characterization of software behavior
- More precise and reliable tests
- More tests overall
- Anybody can run tests!

User Interface Testing

- GUI required manual clicking, a lot of person-time
- Realized schedule would slip; swarmed on testing for delivery
- Focused on how to automate for next delivery



Decouple UI from Engine

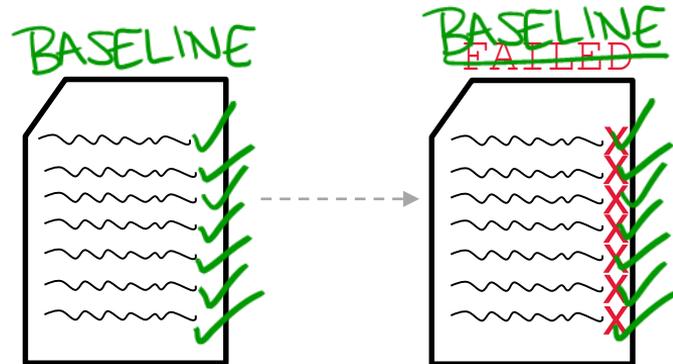
- UI interaction with Engine was not documented
- Developer volunteered for deep-dive investigation
- Luckily, found a way to decouple
- Developer helped recreate the tests for automation

Outcomes

- Test suite previously took over a week
- Could now be run in a single day!
- Still manually tested UI functionality
- Faster executions meant testing more often
- Tested a change by running all 22 final orbits with multiple configurations to compare output

Problems with Baselines

- Verification is all founded on baseline files
- No easy way to narrow down problems during failure
- When software change affects outputs, need to manually re-baseline



Reporting Requirements

- In-house test harness tracked latest status
- Produced report required for delivery
- Explained each test run in each environment
- Reviewed by stakeholders before delivery

Lessons Learned

- Even if unit tests aren't feasible, automated system tests are still beneficial
- Collaboration & commitment from entire team crucial
- Find creative ways to automate

End of Mission

- September 15, 2017: Cassini plunges into Saturn's atmosphere
- Will send data back to Earth until spacecraft incinerates
- “Blaze of Glory” is a fitting end for an historic mission
- Science and engineering legacy will be studied for decades





Jet Propulsion Laboratory
California Institute of Technology