# TAT-C: A Trade-space Analysis Tool for Constellations

Jacqueline Le Moigne,
Philip Dabney, Olivier de Weck,
Veronica Foreman, Paul Grogan,
Matthew Holland, Steven Hughes, Sreeja Nag

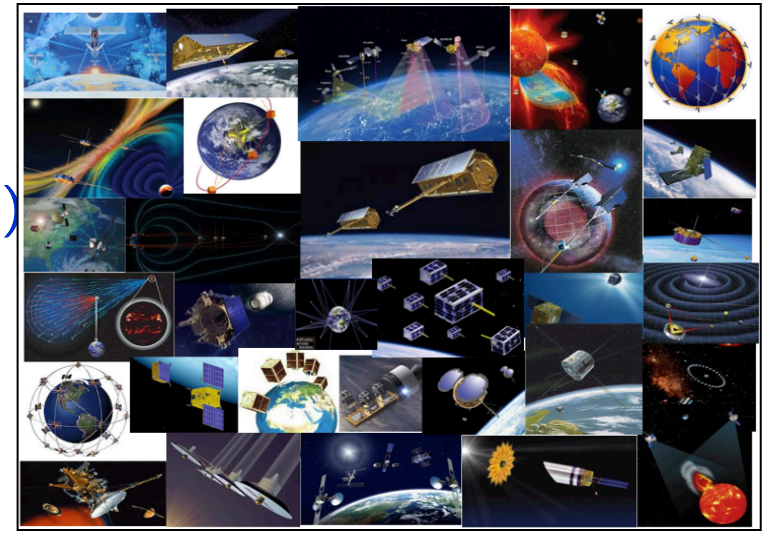
AIST-14-0053 – ESTF16
June 14, 2016

Earth Science Technology Office

# DSM Background

## What is a Distributed Spacecraft Mission (DSM)?

A Distributed Spacecraft Mission (DSM) is a mission that involves multiple spacecraft to achieve one or more common goals.



## Driver/Need

- Reduce the cost, risk and implementation schedule of all future NASA missions
- Investigate the minimum requirements and capabilities to cost effectively manage future multiple platform missions and to cost effectively develop and deploy such missions

## Previous Study

- Determine the Science relevance and the main technology challenges associated with DSM
- Collected data on 62 missions (Earth Science: *12 missions – 7 operational, 3 in development and 2 concepts)*

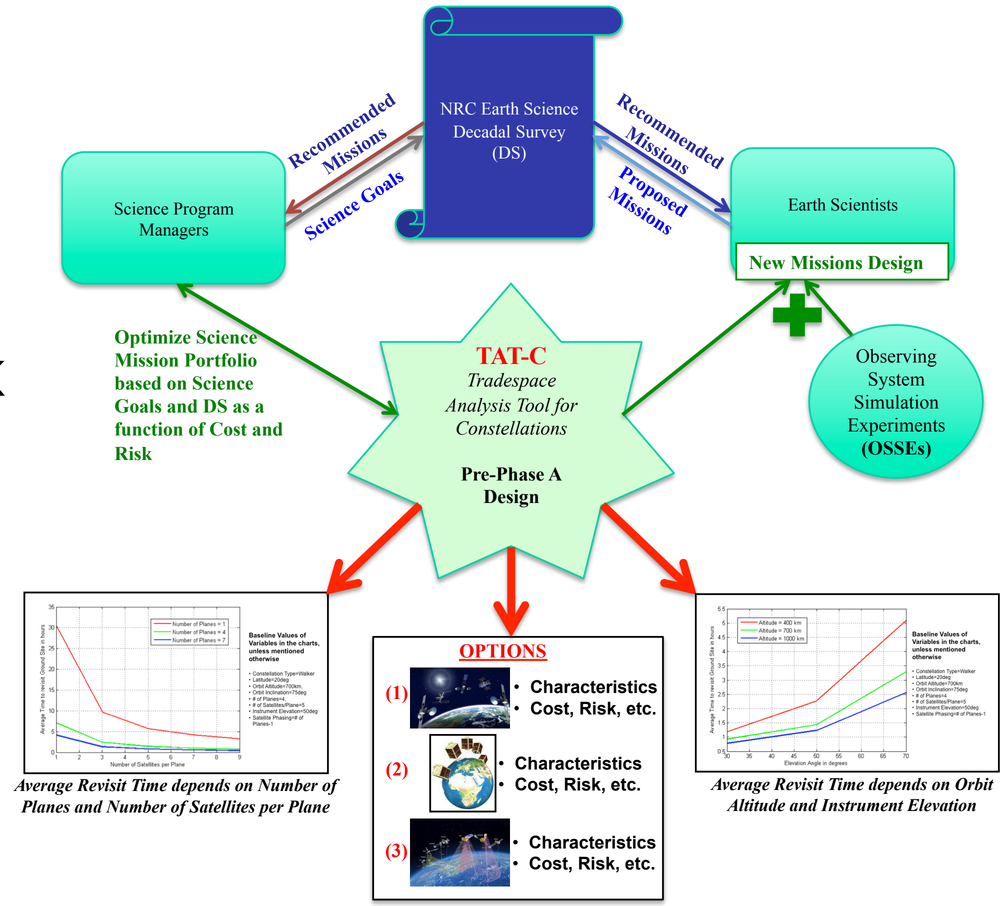
# Some Earth Science DSM Applications

- Studying the Earth Radiation Budget (e.g., CLARREO proposed as a constellation of 6 satellites) with:
  - Better radiometric accuracy
  - Multi-angle view data
- Increasing the temporal and spatial resolution (e.g., Landsat-like constellation)
  - Time series
  - Physiology of plants and phenology of vegetation
- Combining information from complementary sensors (similar to A-Train)
- Investigating various orbits

ESTO
Earth Science Technology Office

# Benefits of DSM

- Enabling New Measurements
- Improving Existing Measurements
- Mitigating Missions Risk
- Reducing Costs
- Facilitating Data Continuity
- Enabling Multi-Organization and International Cooperation

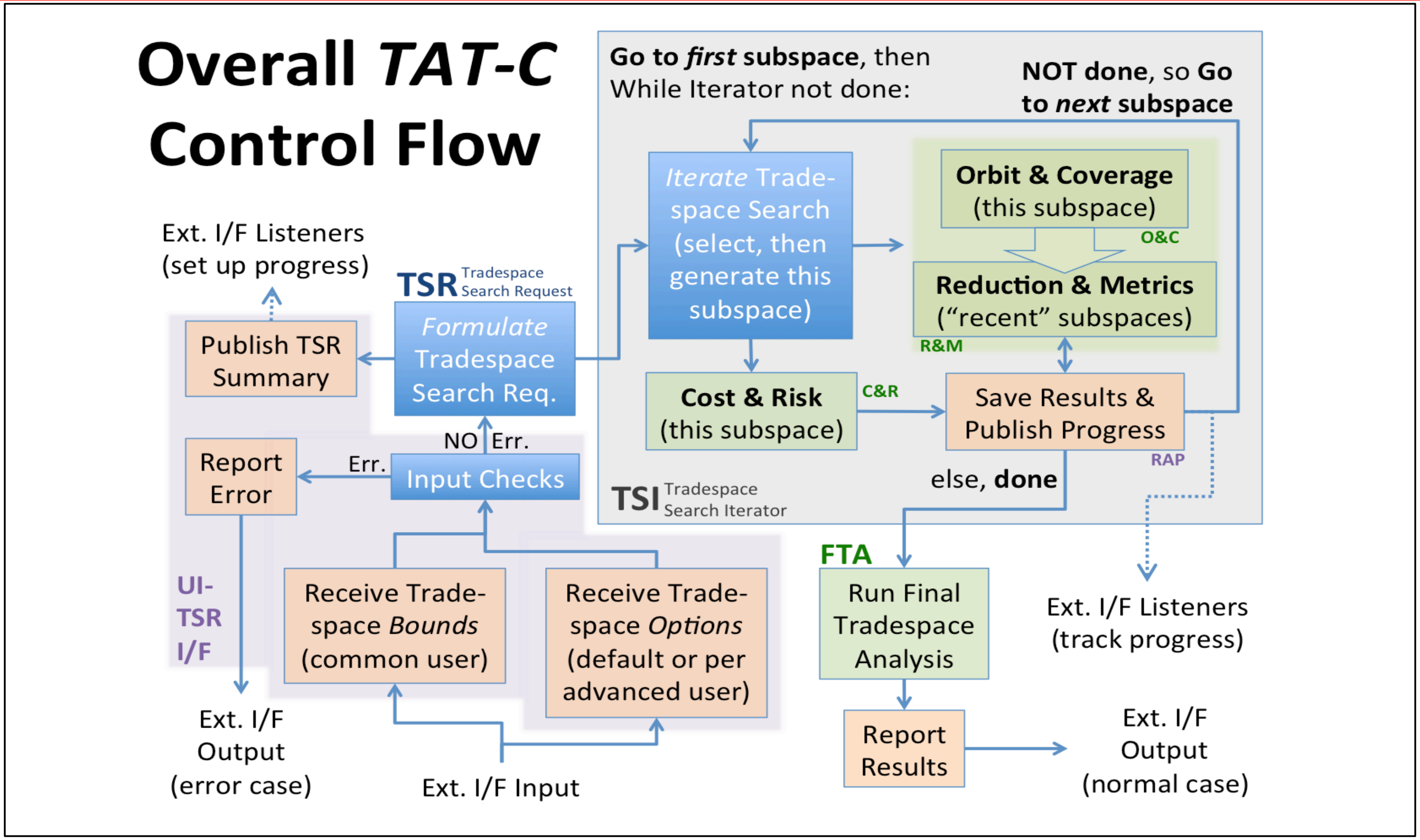


# Objectives and Overall Description

- Provide a framework to perform pre-Phase A mission analysis of Distributed Spacecraft Missions (DSM)
  - Handle multiple spacecraft sharing mission objective
  - Include sets of smallsats up through flagships
  - Explore trade space of variables for pre-defined science, cost and risk goals, and metrics
  - Optimize cost and performance across multiple instruments and platforms vs. one at a time

- Create an open access toolset which handles specific science objectives and architectures
  - Increase the variability of orbit characteristics, constellation configurations, and architecture types
  - Remove STK or any commercial licensing restrictions

# TAT-C Architecture




- Currently, TAT-C can generate architectures for 3 types of missions/payloads: imagers, occultors (to occult with fixed bodies like the Sun or moving bodies like the GPS) and pair-wise functions (bi-static radar, stereo)
- Default ground stations are DSN, NEN w/ positions, bands.
- Launch vehicles database available with cost, reliability, frequency

# Tradespace Search Iterator (TSI)

GUI Inputs

**Iterator Inputs**
- Mission Concept (e.g., Area of Interest, Mission Duration, Launch Options)
- Satellite Specs (e.g., Existing Sats, Alt/Inc ranges, Special Orbits like SSO, comm bands)
- Payload Specs (e.g., conops # - imaging, occultation, stereo, mass, power, volume, FOV, measurement time, spectral bands)
- Any constraints on the range of output values

**Iterator Outputs**
- Metrics per architecture (e.g., average, spatial and temporal distributions)
- Spatial (e.g., Spatial res, Swath overlap %, Occultation positions, coverage)
- Temporal (e.g., revisit, access, repeat)
- Angular (e.g., view zenith, solar illumination)
- Radiometric (Signal to noise fall-off)

- TSI reads user inputs given to the GUI to create iterator inputs (JSON files). Uses default values from Landsat 8 (w/ ETM+ payload) if no inputs
- TSI generates DSM architectures for a combination of variable values that satisfy iterator inputs
- A DSM architecture is a unique combination variable values (altitude, inclination, FOV, number of satellites, etc.)
- For each arch, TSI creates files and send commands to module '*Reduction & Metrics*' to compute architecture performance and to module *'Cost and Risk'* to compute architecture cost
- *Reduction & Metrics* is responsible for calling module '*Orbits & Coverage*' to propagate the orbit of every sat and compute coverage given payload specs. '*Reduction & Metrics'* integrates coverage and computes all performance metrics.

# Tradespace Iterator

**Constellation Types (Currently):**

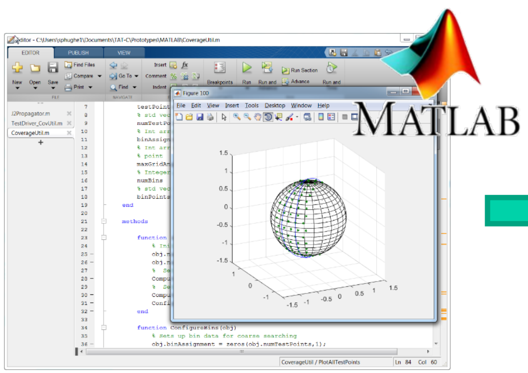
- Uniform Walker Constellations – all sats in the same alt, inc and equal sats per plane
- Ad-Hoc Constellations – all sats assumed to be located at one of the 48 current Planet Labs slots (TLEs available online)

**Constellation Types (In Development):**

- Non-Uniform Walker constellations – combinations of uniform Walker constellations over diff alt, inc, sats/plane
- Ad-Hoc Constellations where sats are launched as per the next available launch option from launch D/B
- Precessing constellation where sats are dropped off by a single LV at differential alt, inc and disperse over time to give RAAN coverage

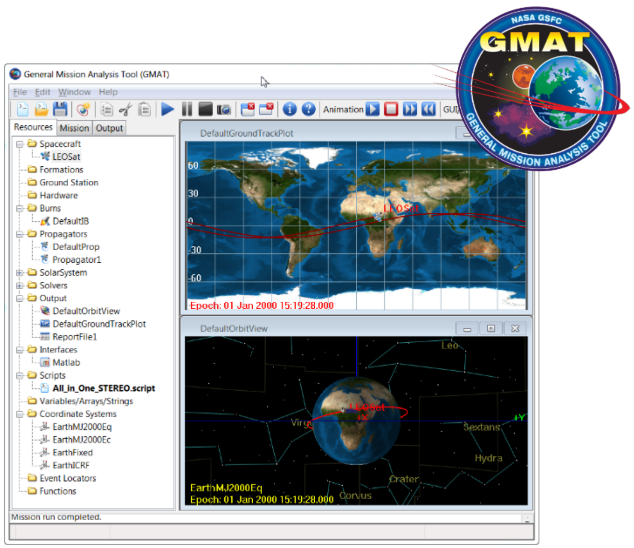
# Orbit-Coverage Module

- Purpose of Module
  - Model orbits balancing accuracy and performance
  - Compute coverage metrics for constellation/sensor set
  - Compute ancillary orbit data for performance, cost, risk
- Development Approach




Prototype Coverage Models in MATLAB

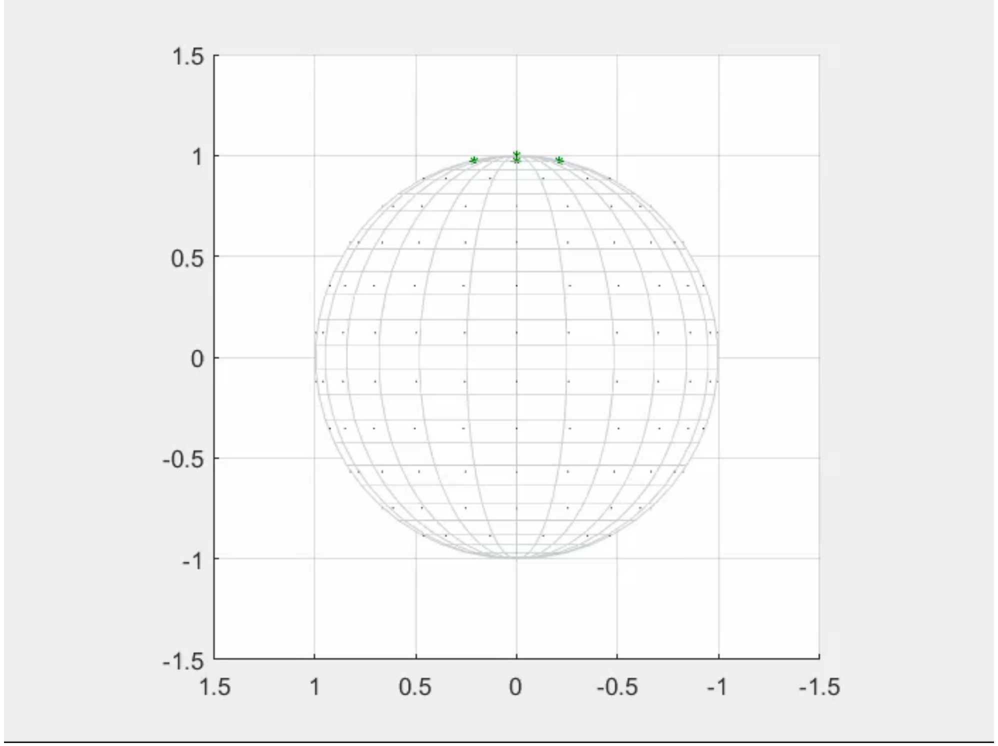

Convert to Stand-Along C++




Potentially integrate into GMAT depending upon future needs

# Orbit-Coverage Module (cont.)

- Concept:
  - TSI Manages mission architecture configuration
  - TSI Calls O&C to compute coverage and ancillary data
  - O&C returns that information to TSI/R&M for further analysis and reduction
- Capabilities:
  - Evenly spaced grid points
  - Custom grid points
  - Conical sensor
  - J2 Dynamics (fast)
  - Multiple spacecraft




Graphic Generated by Coverage Prototype

# Value, Cost and Risk Module

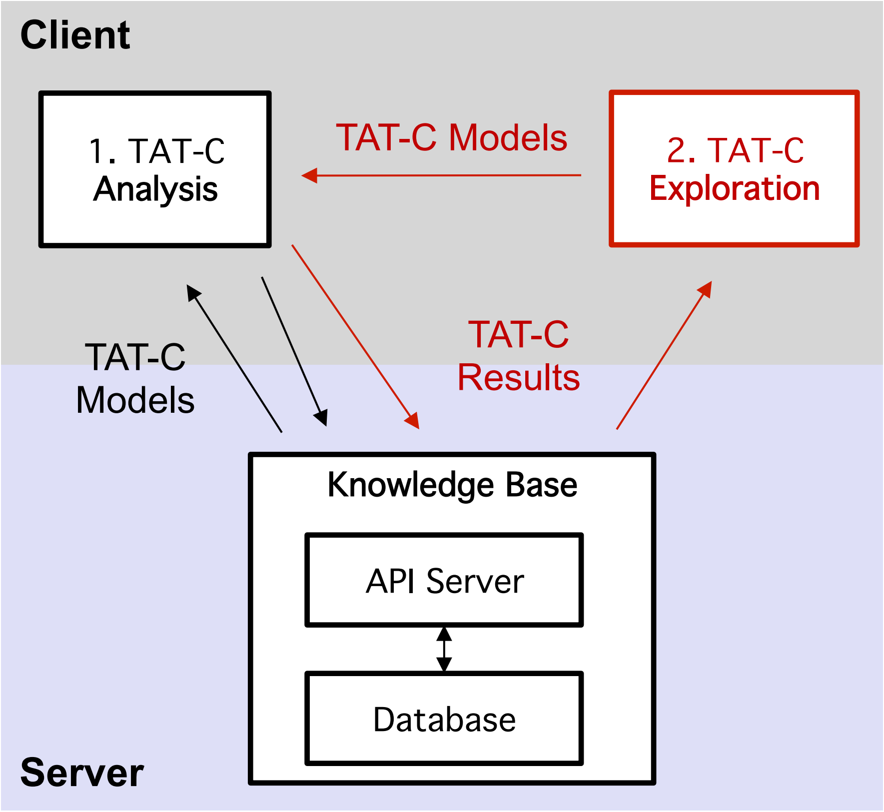
- Given a satellite constellation architecture, the VCR module will provide estimates of:
  - Value, expressed in dollars or utility
  - Cost, life cycle cost (RDT&E, manufacturing, launch, operations)
  - Risk, profile of the system technical and cost risk

- Traditionally, cost and risk estimates have been focused on individual satellites and fixed architectures.
  - It can be difficult to integrate these estimates with the decision making process
  - Previous work has sought to identify the limits of the current tools as they pertain to DSMs; we have continued and extended this effort

- VCR Module will enable trades between performance and value/cost/risk more readily
  - Current work: developing a transparent cost estimating approach that aggregates existing tools and adapts them to DSMs
  - Future work: development of risk and value elements

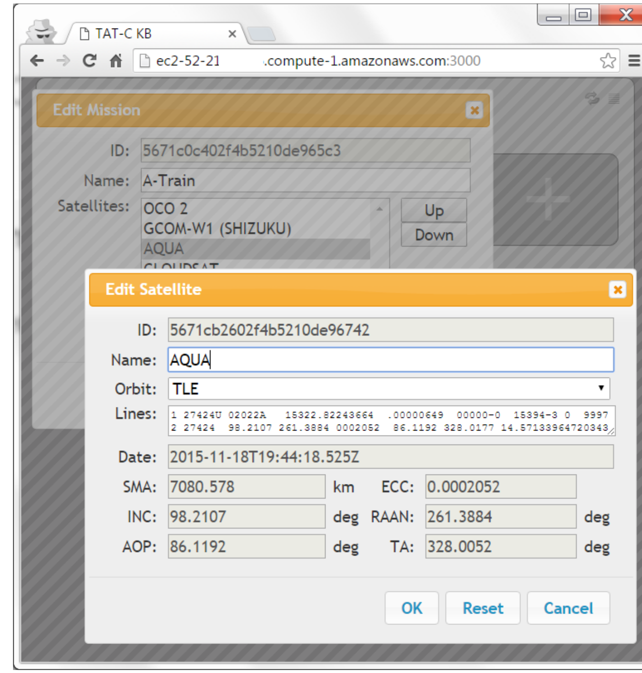
# Knowledge Base

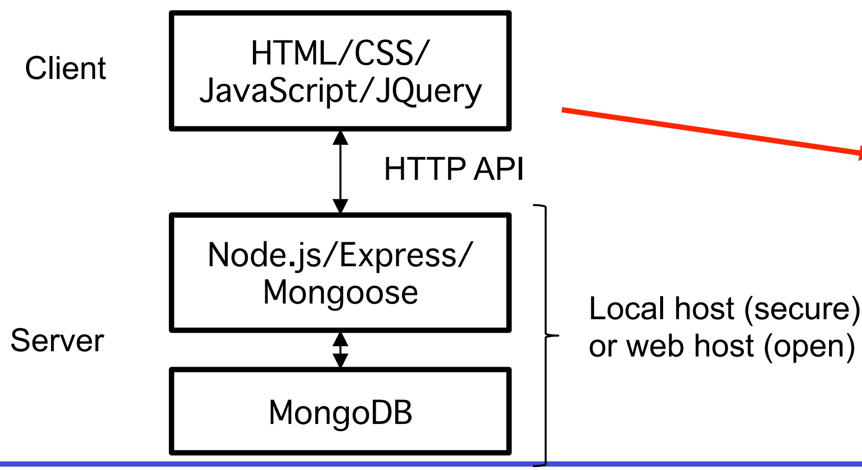
- Centralized store of structured data
- Support TAT-C tasks:
  1. Analysis: compose new mission concepts from existing models
  2. Exploration: discover new mission concepts by querying previous results



# Knowledge Base Prototype

- Demonstrate technical feasibility
  - HTTP requests (JSON)
  - Browser-based GUI client
- Layered client-server architecture
- GET or PUT models:
  - Mission > Satellite > Orbit




```
GET /api/missions
→ [ {id: "5671c…965c3"}, …,
{id: "5671d…9689d"} ]
GET /api/missions/5671c…965c3
→{ "name": "A-Train",
"satellites": […] }
PUT /api/missions/5671c…965c3
→ OK
```

# User Interface

Currently Planned TAT-C User Interface includes GUIs, CLIs and APIs.

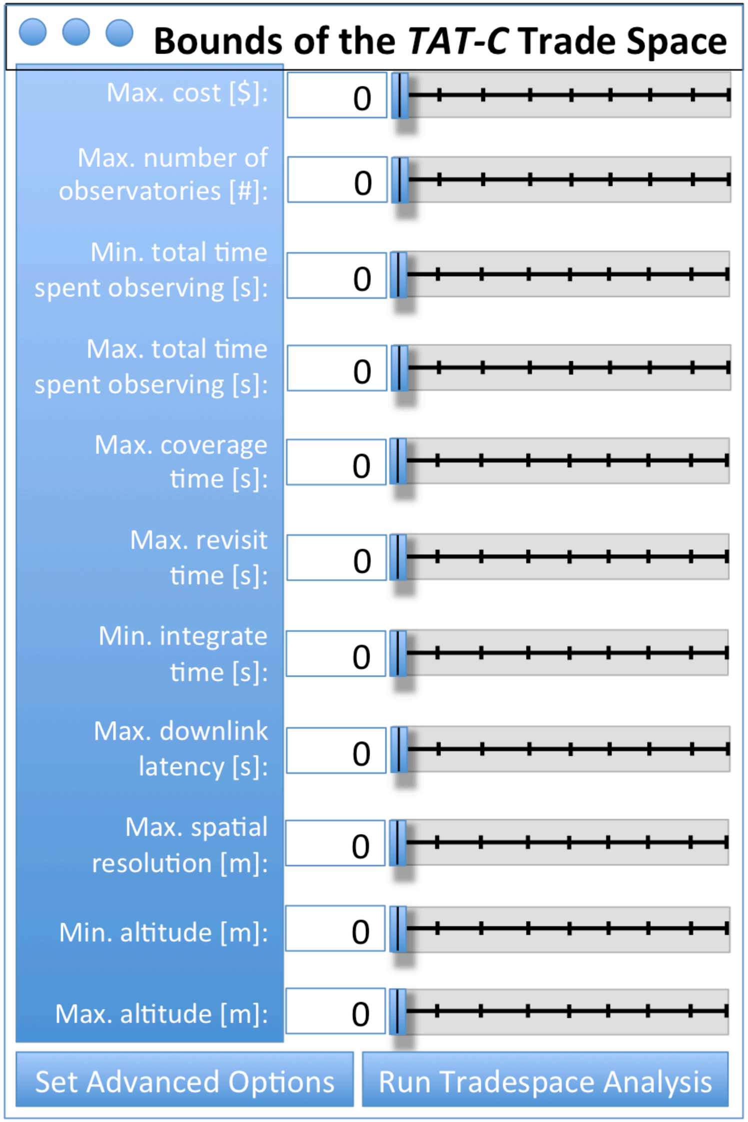
GUIs (Graphical User Interfaces) will be portable to any typical graphical computing environment, and will be designed to function like familiar "Software Wizards", walking users systematically through DSM tradespace choices, and their consequences.

- GUIs will be designed to isolate basic, required, non-expert choices from more expert options typically accessed by more advanced users.
- GUIs will intuitively blend interactive choosing with visual browsing of analysis output characterizing the results of choices.

CLIs (Command Line Interfaces) will be portable to typical command-line environments, and will be designed to enable scripting of interactions equivalent to those possible via GUIs, especially once users establish (and want to automate) their preferred workflows.

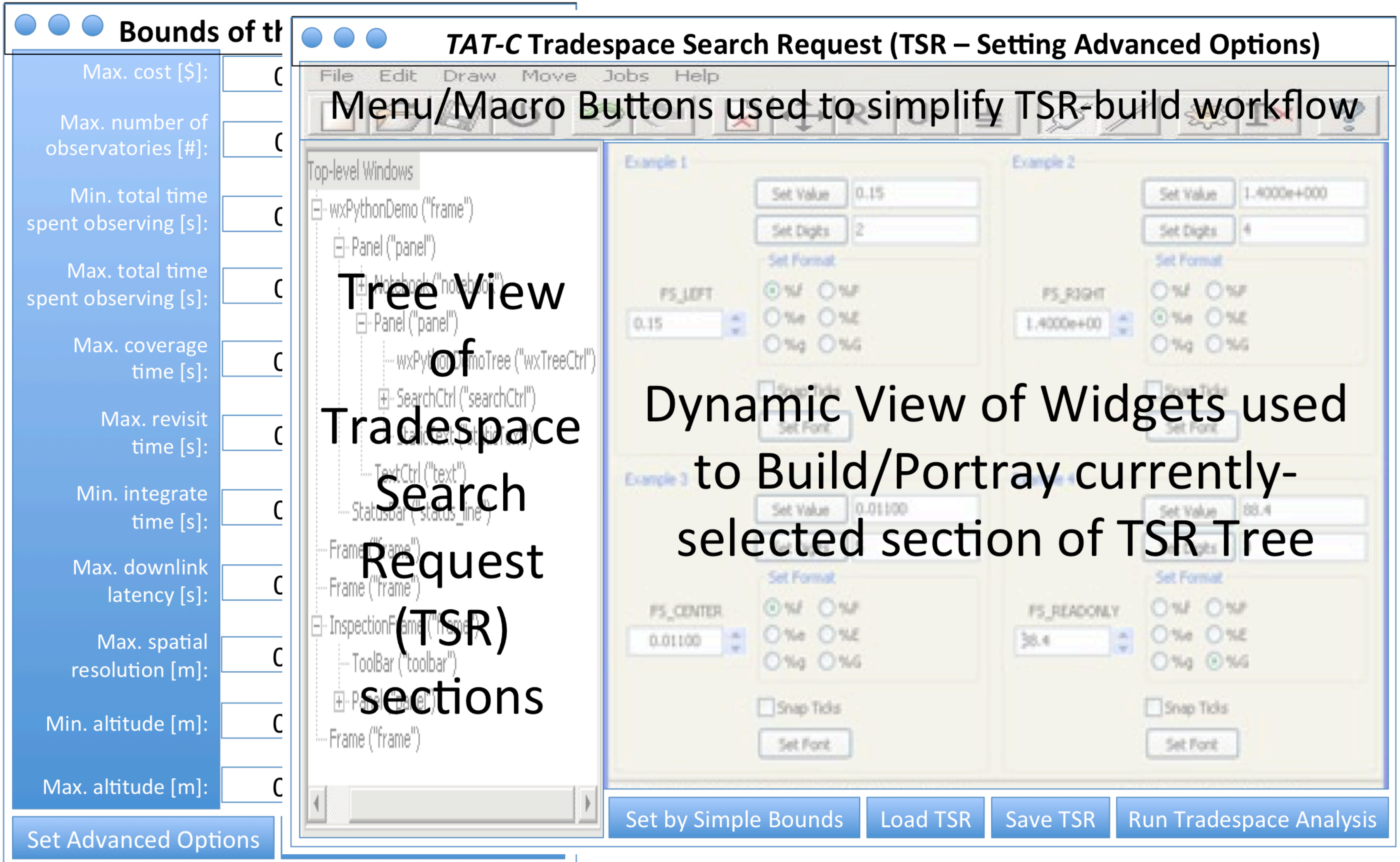
APIs (Application Programmer Interfaces) will expose internal software interfaces to skilled programmers with the expert ability to develop software applications in the "TAT-C Ecosystem".

# Graphical User Interface

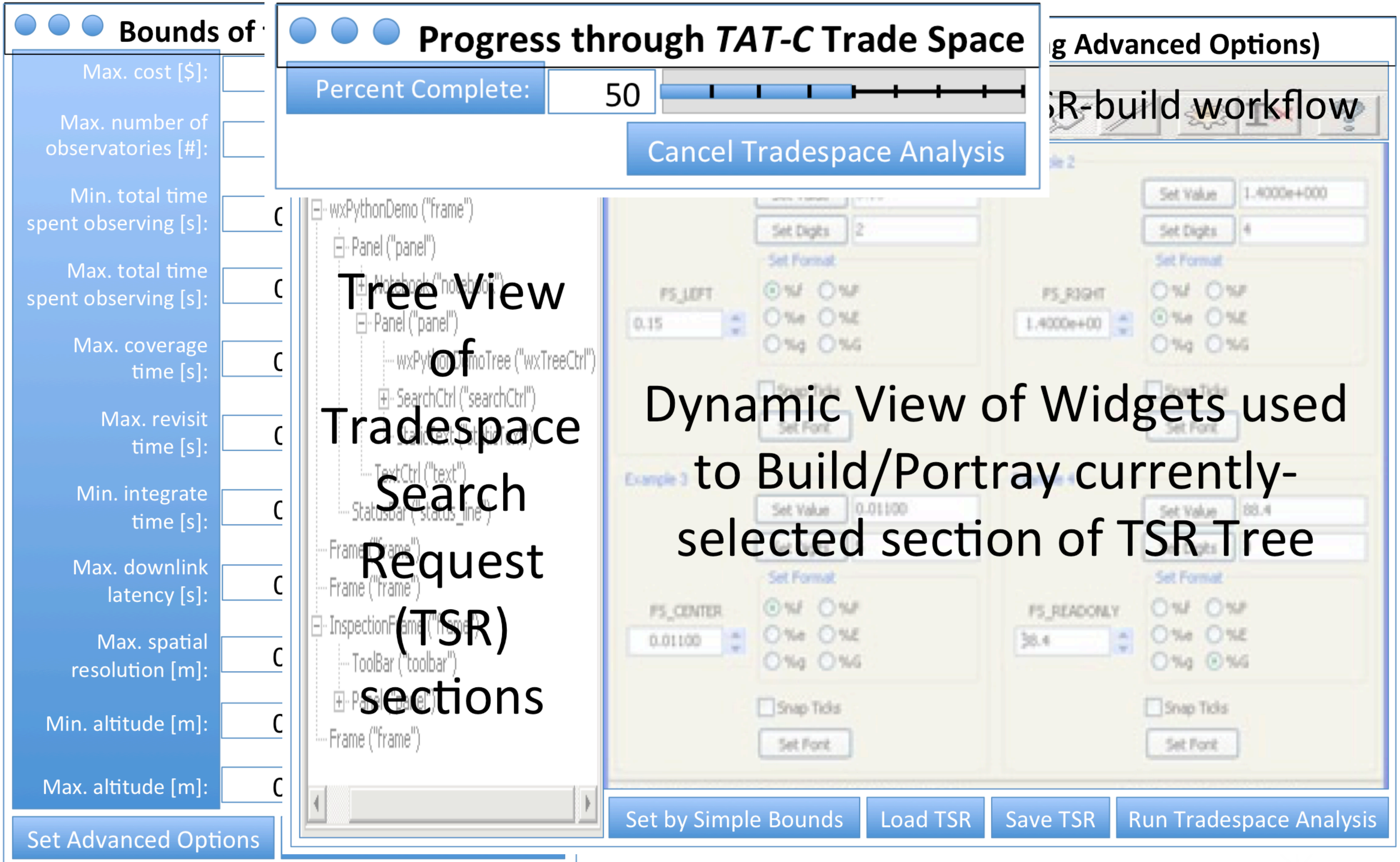



ESTO
Earth Science Technology Office

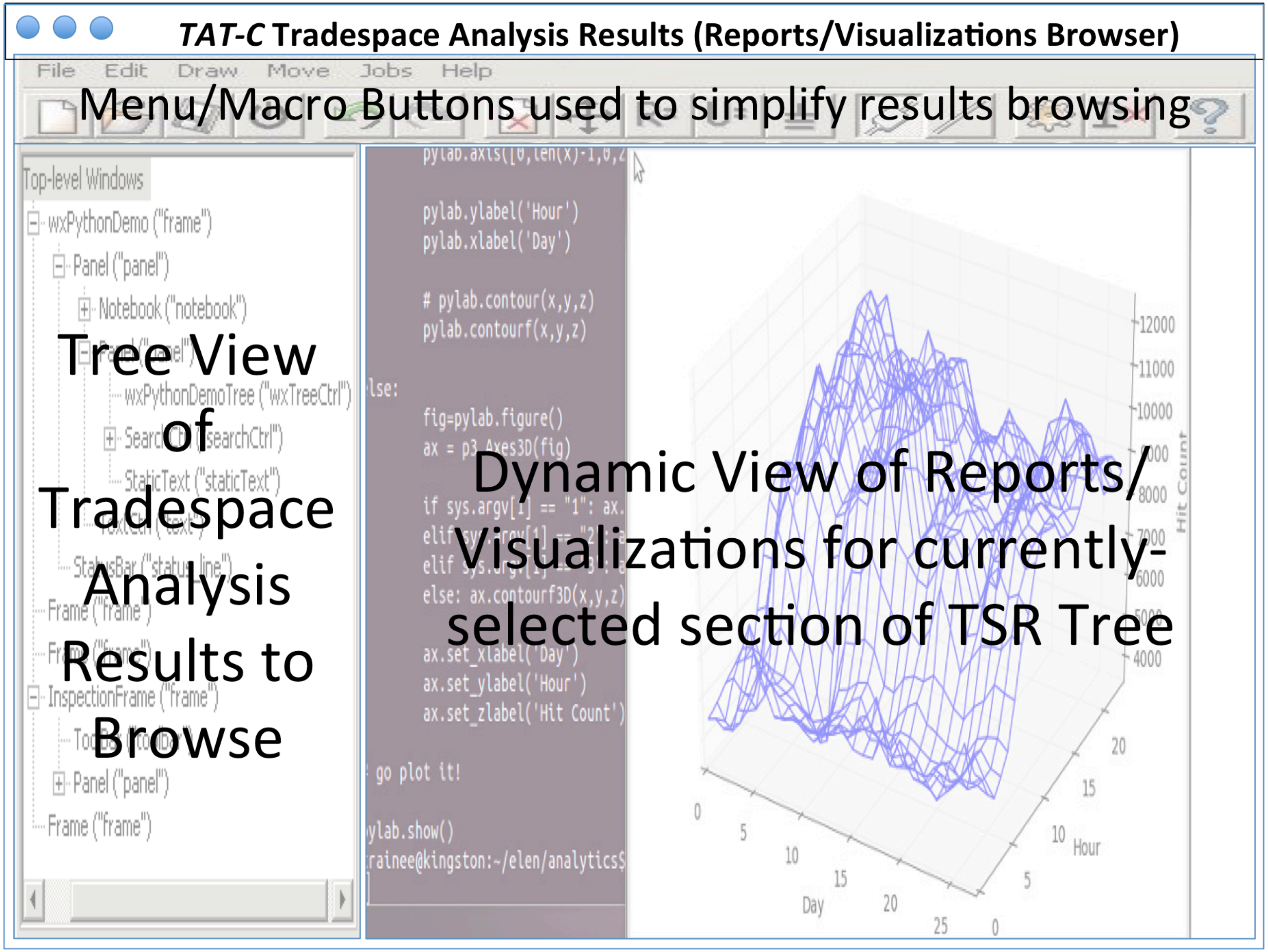
# Graphical User Interface



# Graphical User Interface

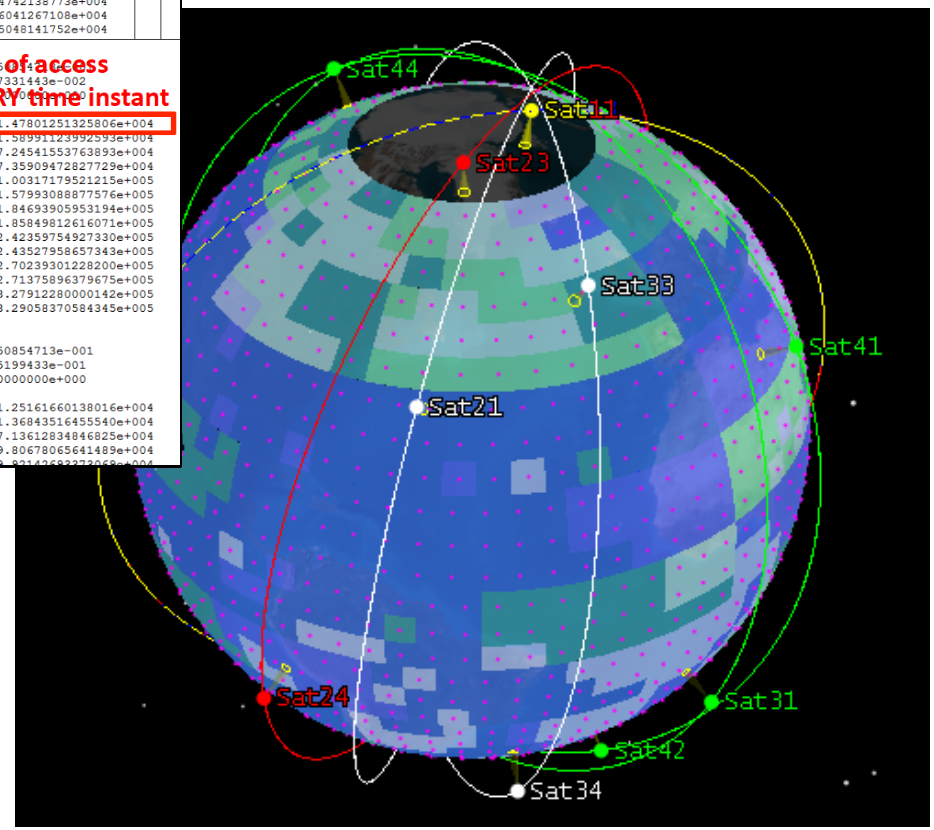


# Graphical User Interface



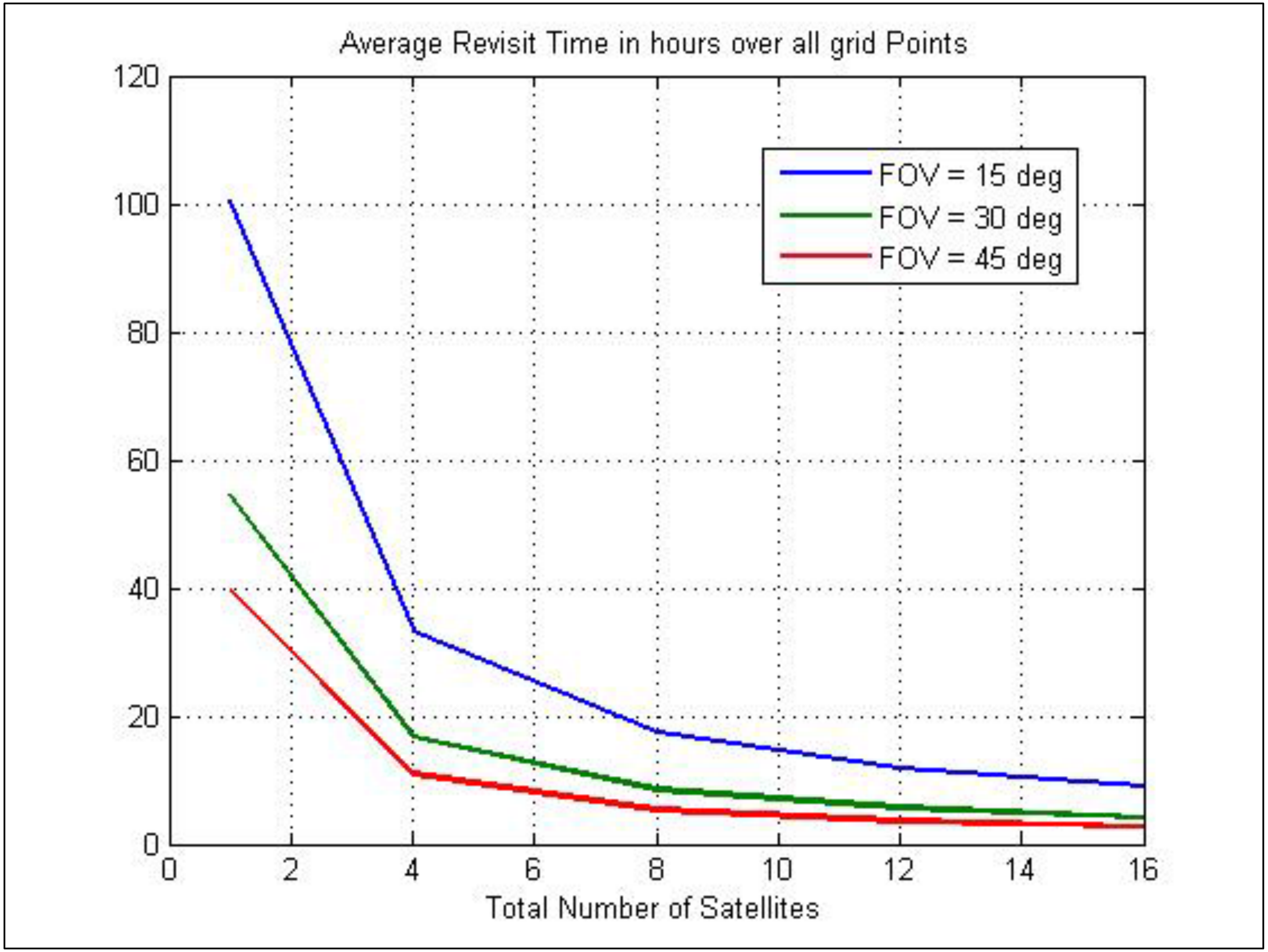
# Graphical User Interface – Proposed Outputs


*Constellation Orbits*




ESTO
Earth Science Technology Office

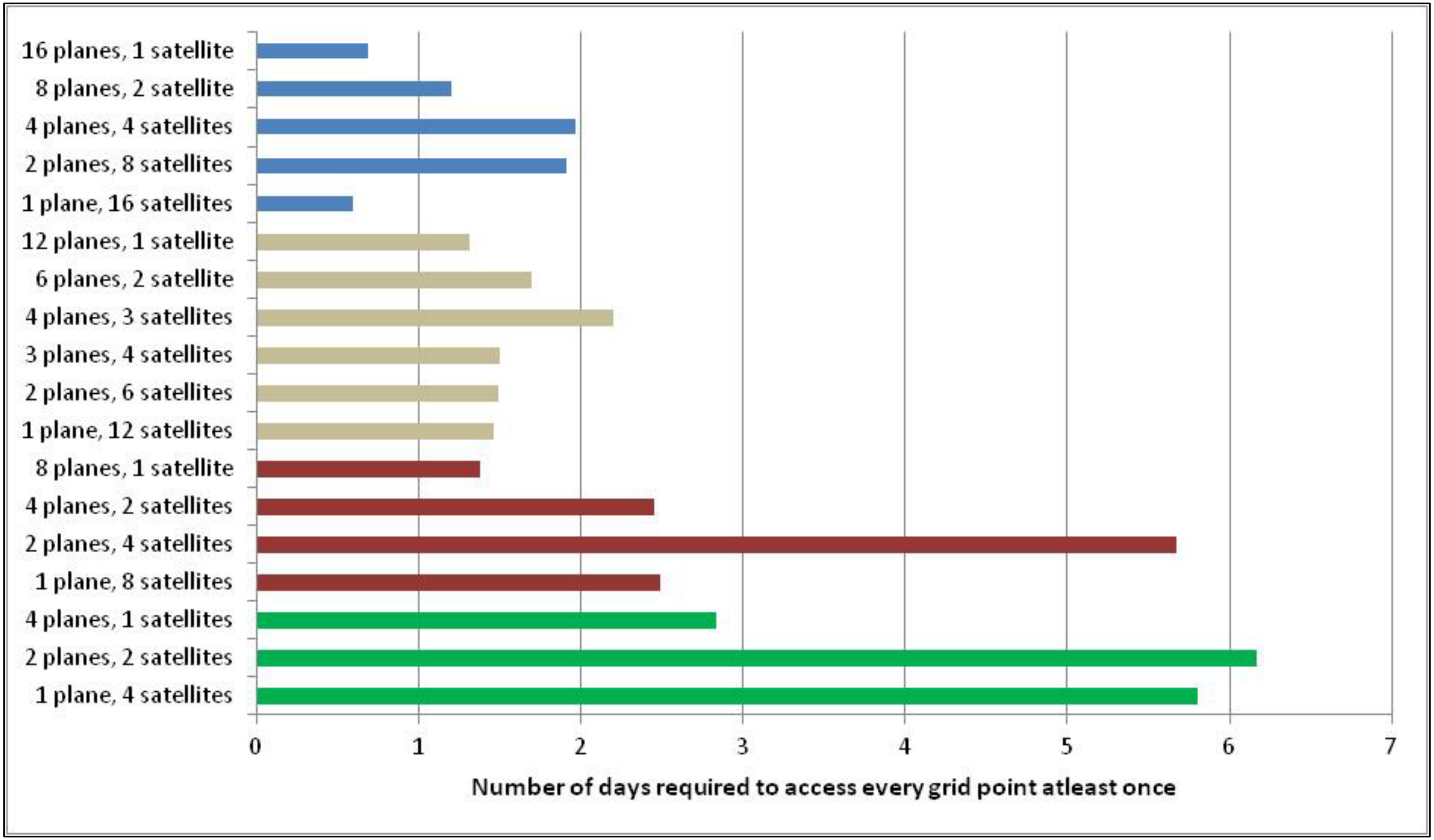
# Graphical User Interface – Proposed Outputs



# Relevancy Scenario – The Sustainable Land Imaging (SLI) Program

- **Meet the coverage requirements**
  - Minimum composite tile size for a single “image” acquisition (90 kmx90 km)
  - Global land cover and some coastal waters equivalent to ~ 500 WRS2 scenes/day
- **Maintain trending and change sensitivity:**
  - Limited range of BDRF angles for both view and solar illumination: defines useable data within a sensor’s field of regard & local overpass times, e.g. - view zenith <30 degrees and local solar time within 30 minutes of 10 am
- **Minimum number of spectral bands for relevant products**
  - Some sensors may only provide a subset of Landsat-8 (SLI baseline) bands
  - “Less capable” satellites may contribute to the SLI community’s data
  - Assessing this allows the capability “holes” to be identified
- **Spatial resolutions**
  - Can the 30m equivalent multispectral band image be derived from the raw data?
- **Assess Acquisition frequency and latency for BDRF restrictions and Band combinations**
  - Community desires a greater re-visit/acquisition frequency

# Graphical User Interface – Proposed Outputs



# Summary and Future Plans

- Definition of External Requirements and Science Requirements
  - Interviews of Stakeholders
  - Definition of Science Products and Required Visualizations
  - GUI Storyboard and Command Line Interface mockups

- Implementation
  - Definition of TAT-C System Architecture and Overall Control Flow
  - All modules version 1 ready
  - *Expect Version Prototype demo in July 2016*

- Future Plans
  - Currently, some modules in Matlab and some in C++ (TSI in Python) => All computation-intensive modules in C++ (faster speed)
  - Refined user interface and TSI
  - Improved capabilities (Type of sensors, type of constellations, etc.)
  - Risk module and knowledge base
  - Use cases testing (starting with SLI)

# Thank you!

Earth Science Technology Office

# Acronyms

- AIST: Advanced Information Systems Technology
- API: Application Programmer Interface
- AST: Architecture Study Team (for SLI)
- BAERI: Bay Area Environmental Research Institute
- CLARREO: Climate Absolute Radiance and Refractivity Observatory
- CLI: Command Line Interface
- COBRA: Complexity Based Risk Assessment Model
- CER: Cost Estimating Relationships
- C&R: Cost & Risk
- DSM: Distributed Spacecraft Mission
- ESTO: Earth Science Technology Office
- ED: Executive Driver
- FTA: Final Trade-space Analysis
- FOV: Field Of View
- GMAT: General Mission Analysis Tool
- KB: Knowledge Base
- MBSE: Model-Based System Engineering
- MIT: Massachusetts Institute of Technology
- NCIM: NASA Instrument Cost Model

# Acronyms

- O&C: Orbit & Coverage
- OSSE: Observing System Simulation Experiment
- R&D: Research & Development
- R&M: Reduction & Metrics
- SICD: Software Interface Control Document
- SLI: Sustainable Land Imaging
- SSCM: Small Satellite Cost Model
- STK: Systems Tool Kit (from AGI, Analytical Graphics, Inc.)
- TAT-C: Trade-space Analysis Tool for Constellations
- TSI: Trade-space Search Iterator
- TSR: Trade-space Search Request
- (G)UI: (Graphical) User Interface
- USCM: Unmanned Space Vehicle Cost Model
- USGS: United States Geological Survey
- VARG: Value-At-Risk-Gain
- VCRM: Value Centric Risk Management

# Generating DSM Architectures

Iterator Inputs

Loops to generate unique architectures:

- *For every FOV in 4 equally distributed points in input FOV Range:* Make 4 equally distributed points in input Altitude Range. If special orbits, altitude is constrained.
  - *For every Altitude option among the 4 equally distributed points:* Calculate the swath on the ground for the given alt and FOV. Use swath to calculate the minimum and maximum number of satellites needed to achieve max and min revisit constraints. For computed range of satellite numbers, get 10 equally distributed options in between.
    - *For every total Sat Num option among the 10 equally distributed points:* Calculate all possible Walker constellation configs with # of planes × sats/plane = total sats. Each uniform config is defined by the initial RAAN and TA of the constituent sats.
      - *For every inclination in 4 equally distributed points in input Inclination Range:* Constrain inclination inputs if there is a special orbits constraint. Incl depends on alt option, if so.
        - *For every RAAN-TA combination for a given inc, sat number, alt, FOV (unique architecture):* For each sat, Compute the initial Keplerian elems. If the sat, GS and payload combination is unique, store all their specs as a unique JSON file under the User/Mono/ folder. For the whole DSM, Store the mapping of the unique sats that make up the DSM (pointer to Mono/ location) as JSON files under a unique folder for this arch in User/DSMs/ folder.

Then, create JSON file with information for each DSM for the cost/risk module (e.g., orbital params, maintenance delta-V)