CAL POLY

Aerospace Engineering Department

Getting to know

CAL POLY AEROSPACE ENGINEERING

CAL POLY
SAN LUIS OBISPO

LEARN-BY-DOING; PRACTICING AND INNOVATING AEROSPACE DESIGN

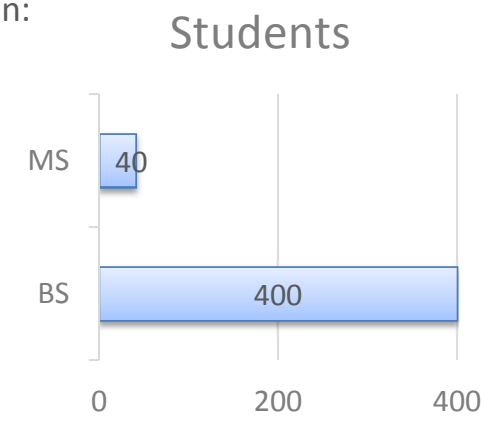
Vision - An alumni base leading and innovating the Aerospace Industry

Mission - As faculty, students and staff, we collaborate with the Aerospace Industry to build partnerships which promote excellence and innovation to serve diverse communities. We work as a team to provide an excellent Learn-by-Doing, systems and design focused engineering education; graduating Day One-ready professionals

CAL POLY
SAN LUIS OBISPO

A BIT OF HISTORY

- Founded in 1927 with an emphasis on Aircraft Mechanics and Maintenance
- Started building airplanes in 1928
- A strong tradition of Aerospace Systems Engineering and Design has been built as part of our “Learn-by-doing” motto
- Incoming Freshman:
  - 4.03 GPA
  - 1305 SAT



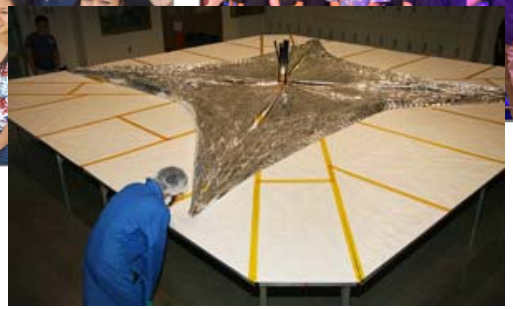
Faculty and students with the first plane built at Cal Poly SLO. The design is based on the Spirit of St. Louis

CAL POLY
SAN LUIS OBISPO

THE HOME OF CUBESAT

Photograph of CP-4 taken by AeroCube, April 17th, 2007

- CubeSat standard invented by Dr. Bob Twiggs of Stanford and Dr. Jorid Puig-Suari of Cal Poly
- The CubeSat standard in currently maintained by students and faculty at Cal Poly
  - cubesat.org



CAL POLY
SAN LUIS OBISPO

THE FUTURE OF UAS

- Applications of Autonomous Flight Initiative
- Newly hired faculty member with 15+ years industry experience
- Support for alumni and Cal Poly
- Partnering with College of Agriculture

- Goals

1. Prepare students to work in the UAS industry
2. Facilitate multi-disciplinary research using UAS
3. Research technologies for UAS
4. System assessment of UAS concepts

Dr. Aaron Drake, his team of students and the RMAX

CAL POLY
SAN LUIS OBISPO

Students and Industry Representatives at the Cal Poly SLO Aircraft and Spacecraft Design Symposium

SENIOR DESIGN AT CAL POLY

- RFP and Requirements Driven
  - A strong emphasis on systems engineering principles
- Industry-centric
  - Design projects are relevant industry projects
  - Industry sight visits and tours
- Critical Reviews
  - 3-4 industry hosted design reviews
  - Design Symposium hosted on campus in May – PDR/CDR level review

CAL POLY
SAN LUIS OBISPO

ASTRONAUTICS CURRICULUM

- Fully implemented in 2013
- Includes all topics required by ABET for an accredited BS program in Astronautics:
  - Spacecraft Attitude Determination and Control
  - Orbital Mechanics
  - Space Structures
  - Rocket Propulsion
  - Telecommunications
  - Space Environment
  - Power Systems (we added this one)

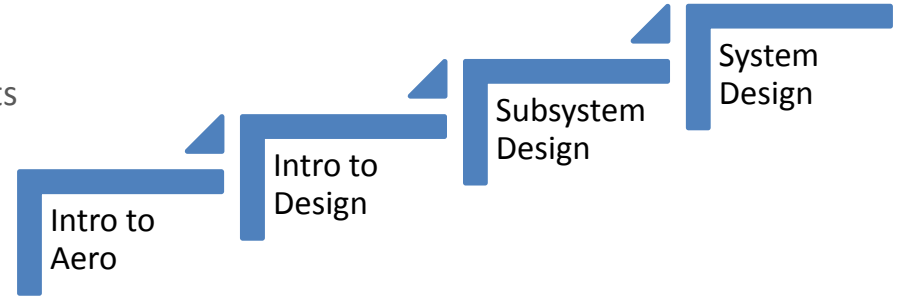
The Space Environments lab at Cal Poly, SLO

SYSTEMS ENGINEERING AT CAL POLY

CAL POLY
SAN LUIS OBISPO

NO DEDICATED CLASS

- Concepts are introduced as needed throughout the curriculum
- Freshman Year
  - Intro to Aerospace Engineering
    - Engineering Requirements
- Sophomore Year
  - Intro to Aerospace Design
    - Verification and Validation of Requirements
- Junior Year
  - Subsystem specific design strategies
- Senior Year
  - Full system design, functional teams



Systems engineering principles throughout the curriculum

CAL POLY
SAN LUIS OBISPO

Systems Engineering

PLANS AND UPDATES

As we continue to update the curriculum, we are seeking to formalize the systems engineering thread. We want to map learning outcomes associated with systems engineering to all courses offered in both concentrations with a culmination in Senior Design

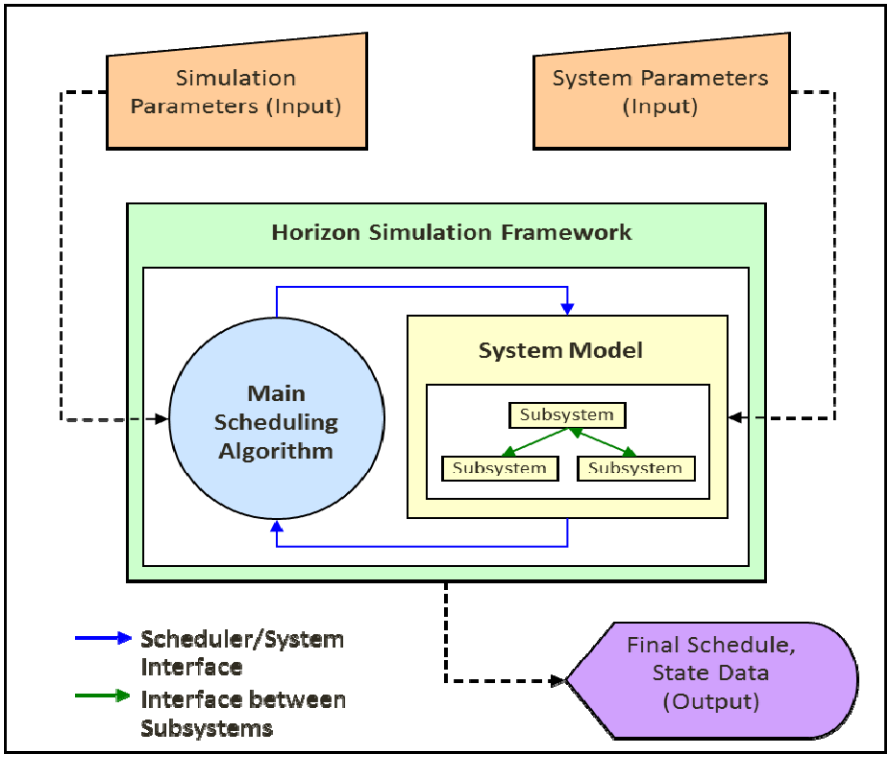
SPECIFIC ACTIONS

- Sophomore course in system integration
- Further refinement and restructuring of discipline specific design strategies
- Elective class in modeling and simulation

THE HORIZON SIMULATION FRAMEWORK

Software built for system-level requirements verification through simulation

CAL POLY
SAN LUIS OBISPO



WHAT IS *HSF*?

- A library of integrated software tools supporting rapid system-of-systems modeling, simulation and utility analysis
- Provides an extensible and well-defined modeling capability concurrent with a scheduling engine to generate operational schedules of Systems-of-Systems with corresponding state data
- Modular, Flexible with built-in utilities

FUNDAMENTAL SCHEDULING ELEMENTS

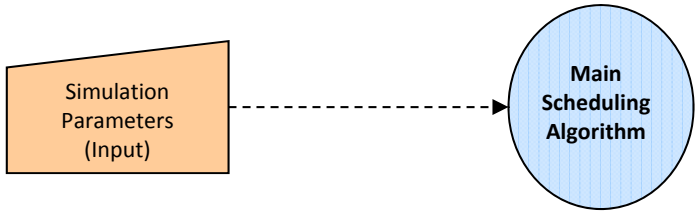
CAL POLY
SAN LUIS OBISPO

- **Four fundamental scheduling elements**
  - **Task** – The "objective" of each simulation time step. It consists of a target (location), and performance characteristics such as the number of times it is allowed to be done during the simulation, and the type of action required in performing that task.
  - **State** – The state vector storage mechanism of the simulation. The state contains all the information about system state over time and contains a link to its chronological predecessor.
  - **Event** – The basic scheduling element, which consists of a task that is to be performed, the state that data is saved to when performing the task, and information on when the event begins and ends.
  - **Schedule** – Contains an initial state, and the list of subsequent events. The primary output of the framework is a list of final schedules that are possible given the system.

THE MAIN ALGORITHM

- Contains the interface between the main scheduling module and the main system simulation module

- Guides the scheduling algorithm in discrete time steps and keeps track of the results



CAL POLY
SAN LUIS OBISPO

SELECTABLE MAIN SCHEDULING ALGORITHM

- BDESA – Big Dumb Exhaustive Search Algorithm
  - Baseline scheduling algorithm
  - Provides a recursive scheduling algorithm which allocates constrained resources with respect to a cost function
- Hybrid Genetic Algorithm
  - Recently added to the *HSF*
  - Provides improved performance over the BDESA in most cases

CAL POLY
SAN LUIS OBISPO

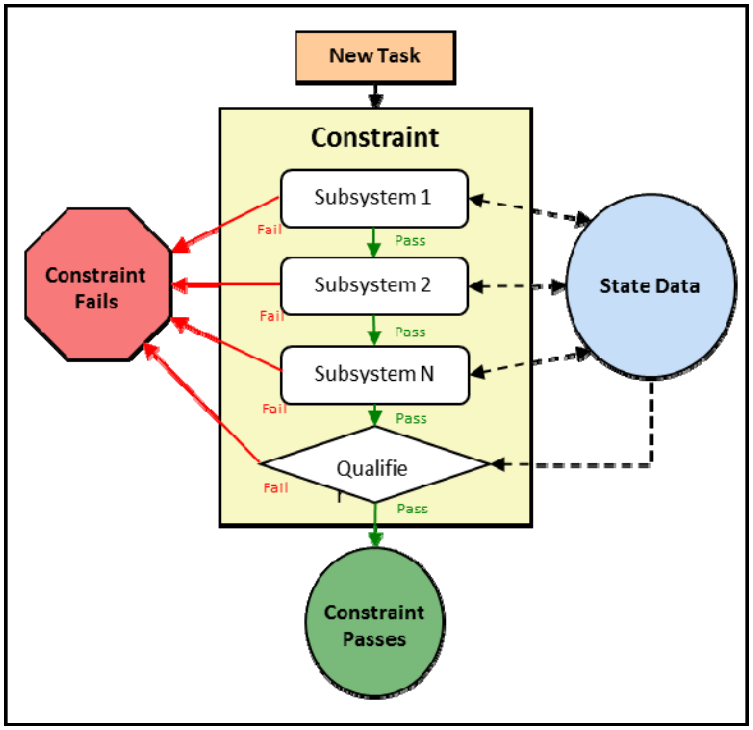
FUNDAMENTAL MODELING ELEMENTS

- **Constraint** – A restriction placed on values within the state, and the list of subsystems that must execute prior to the Boolean evaluation of satisfaction of the constraint. Also the main functional system simulation block, in that in order to check if a task can be completed, the scheduler checks that each constraint is satisfied, indirectly checking the subsystems.
- **Subsystem** – The basic simulation element of the framework. A subsystem is a simulation element that creates state data and affects either directly or indirectly the ability to perform tasks.
- **Dependency** – The limited interface allowed between subsystems. In order to keep modularity, subsystems are only allowed to interact with each other through given interfaces. The dependencies specify what data is passed through, and how it is organized. Dependencies collect similar data types from the passing subsystems, convert them to a data type the receiving subsystem is interested in, and then provide access to that data.
- **System** – A collection of subsystems, constraints, and dependencies that define the thing or things to be simulated, and the environment in which they operate.

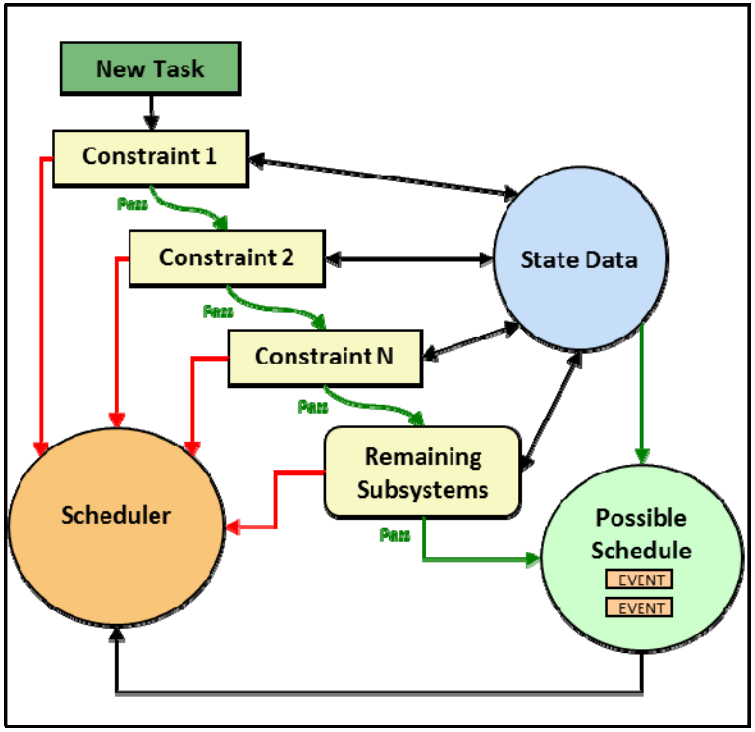
CAL POLY
SAN LUIS OBISPO

CONSTRAINT-CHECKING CASCADE

- Primary algorithm to check whether a system can perform a task
- Internal constraint process:
  - Subsystems which contribute state data to Qualifier are evaluated
  - Qualifier evaluates validity of state
  - Constraint fails if a subsystem or the qualifier fails



CAL POLY
SAN LUIS OBISPO



CONSTRAINT-CHECKING CASCADE

- Constraint-Checking Cascade:
  - Constraints are checked in user-specified order
  - Remaining subsystems evaluated
  - If any of the checks fail, no event is added to the schedule and the state is discarded
  - If all of the checks succeed, the task and state are added to the schedule as a new event
- "Fail-fast" constraint methodology

CAL POLY
SAN LUIS OBISPO

SUBSYSTEMS

- Subsystems are state transition objects
  - Describe how the subsystem goes from its old state to new state in performing the task
- Inputs
  - Old state of the Subsystem
  - Task to be performed
  - Environment to perform it in
  - Position of their asset
- CanPerform() is the main execution method
  - Code describes the governing equations of the subsystem
- Output
  - New State of the Subsystem

CAL POLY
SAN LUIS OBISPO

DEPENDENCIES

- Dependencies are the interpreters between subsystems

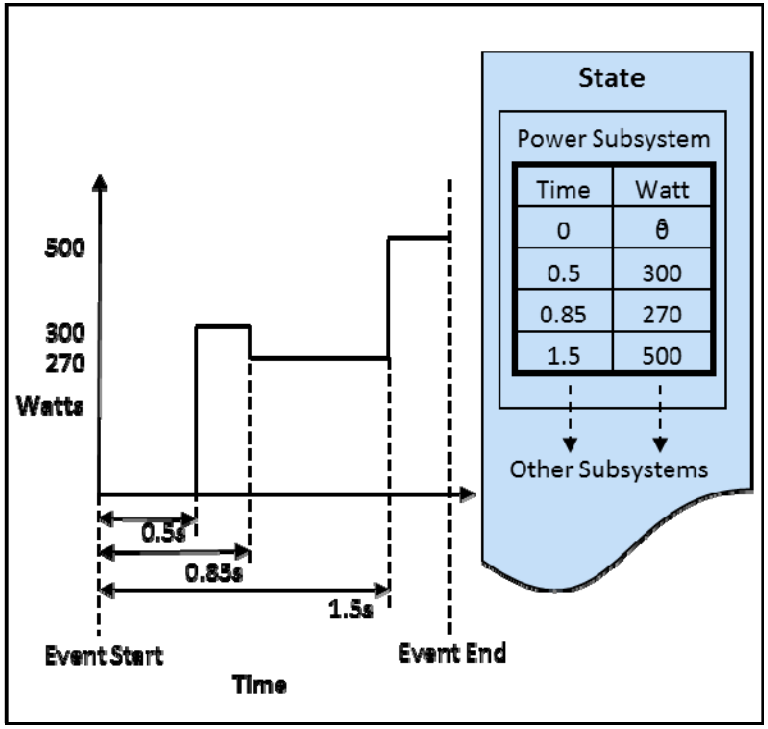
- Example: Power subsystem dependent on ADCS subsystem for power input of solar panels due to solar panel incidence angle to sun vector
  - ADCS only interested in orientation within Environment
  - Power only interested in how much power other subsystems generate or consume
  - The dependency function would translate the orientation of the spacecraft into how much power the solar panels generate

- Dependencies structured as they are to avoid “subsystem creep”
  - Information about and functions from each subsystem do not migrate into the other subsystems

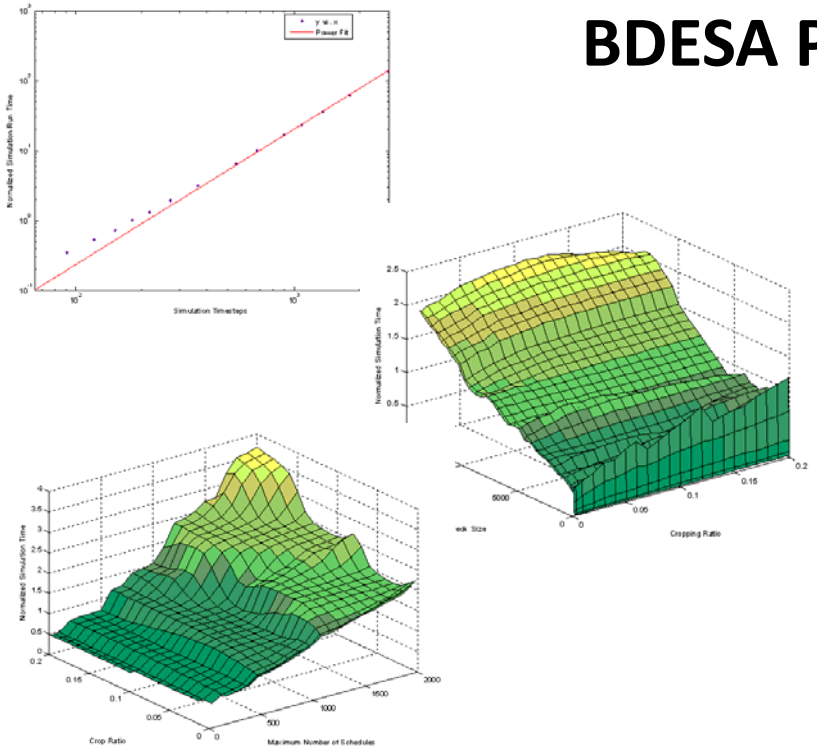
CAL POLY
SAN LUIS OBISPO

THE SYSTEM STATE

- State is unique to each event
- All the data generated over the course of the event is stored in its corresponding state
- Storage like a bulletin board
  - Only changes from previously recorded values are posted
  - Most recent saved value of the variable is also the current value



CAL POLY
SAN LUIS OBISPO



BDESA PARAMETRIC RUNTIME ANALYSIS

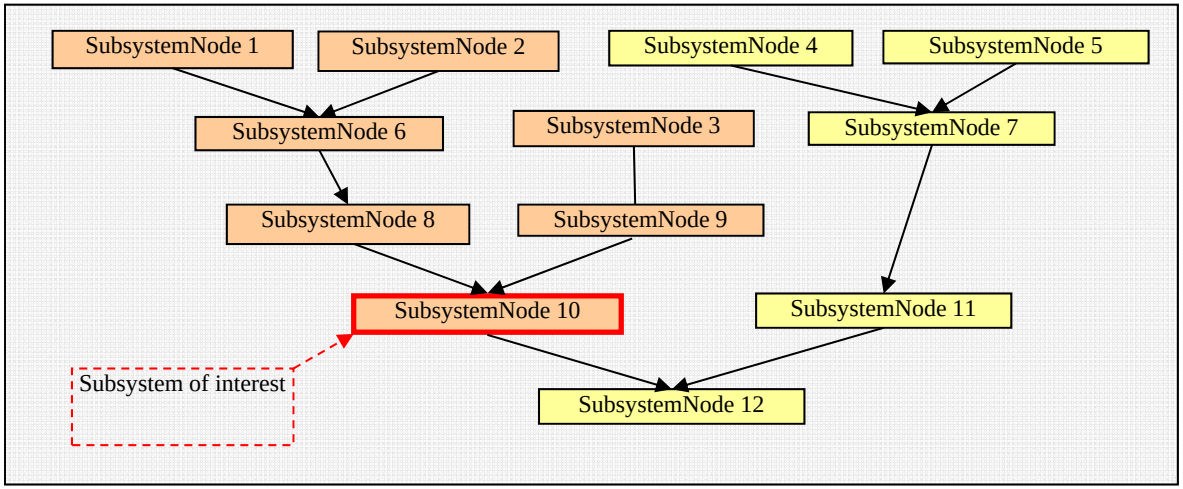
- D – Target deck size
- A – Number of Assets
- S_{max} – Maximum number of schedules allowed before cropping
- n – Number of time steps in simulation
- t_{sys} – Mean system execution time

$$O\left(D^{A} S_{max} n^{2} t_{sys}\right)$$

CAL POLY
SAN LUIS OBISPO

SUBSYSTEM DEFINITION

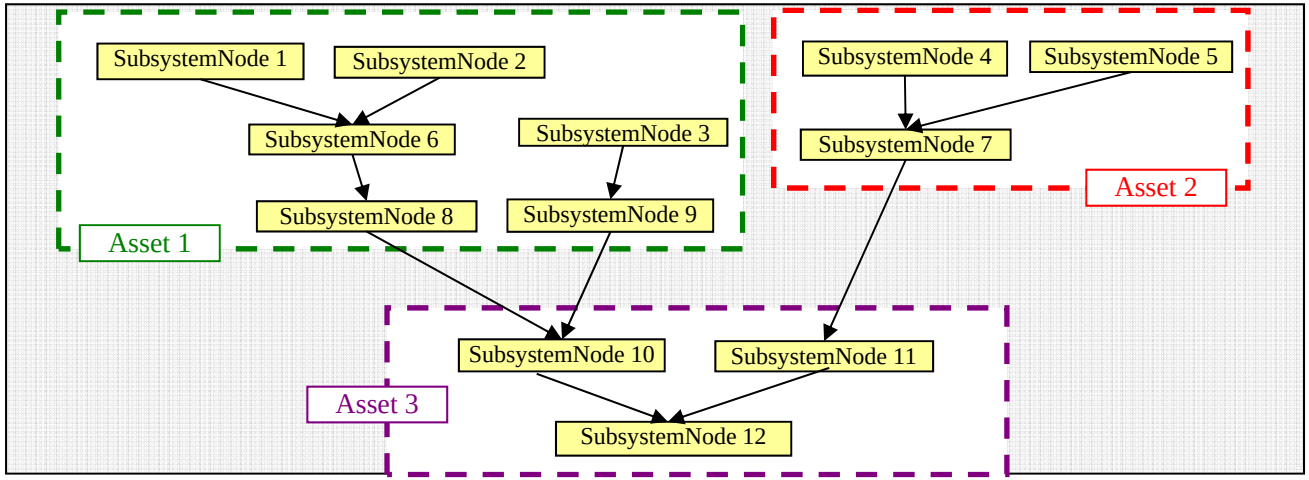
- Subsystems are encapsulated by SubsystemNodes
- SubsystemNodes solve the problem of subsystems having no hierarchical information
- SubsystemNodes point to the SubsystemNodes they are dependent on, as well as the Subsystem they represent
- Multiple SubsystemNodes can point to the same Subsystem



CAL POLY
SAN LUIS OBISPO

ASSET DEFINITION

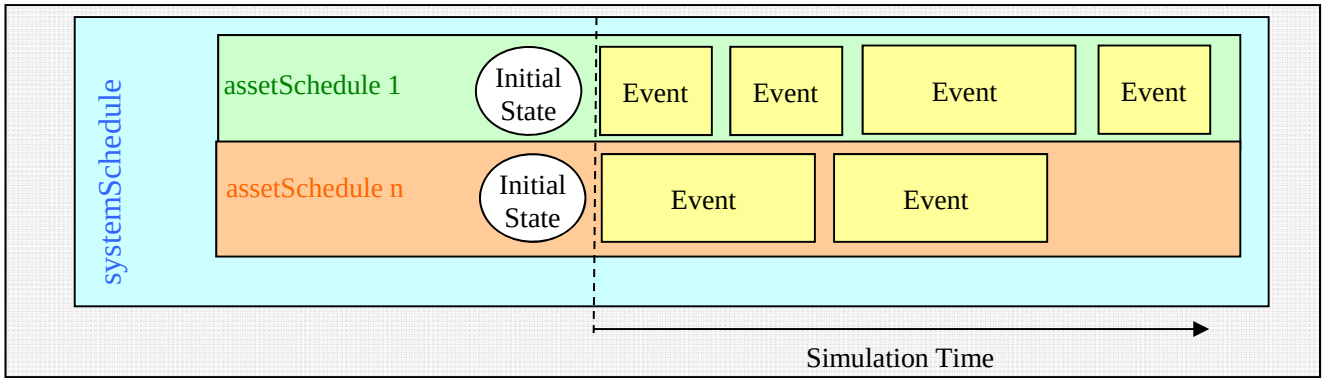
- In order to Task multiple assets, Asset objects were created
- Assets defined as: Any actor that had subsystems as members and knowable motion



CAL POLY
SAN LUIS OBISPO

SYSTEM AND ASSET SCHEDULES

- Multiple assets requires the ability to task each asset independently
- Each asset must then have its own schedule (which is called an AssetSchedule)
  - AssetSchedules hold a list of events and an initial state
- The whole system must have a unique schedule (called a SystemSchedule)
  - SystemSchedules hold a list of AssetSchedules



HSF VERSION HISTORY

CAL POLY
SAN LUIS OBISPO

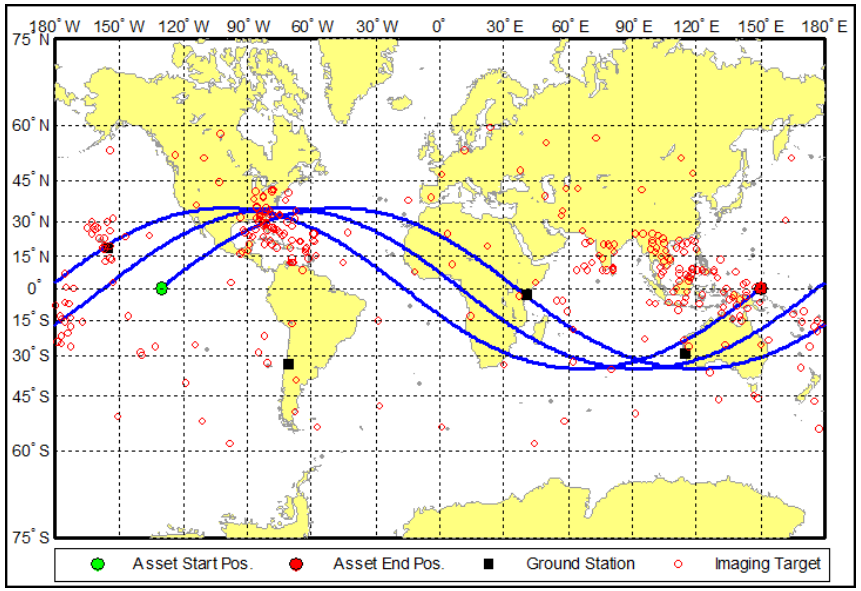
- 1.0 – Initial Release
- 1.1 – Math Libraries Added
- 1.2 – State Profile Added
- 2.0 – Multi-Asset Capability
- 2.1 – Auto Circular Dependency Check
- 2.2 – LUA Scripting
- 2.3 – Multi-core and Networking Support

THE AEOLUS CONSTELLATION

The Horizon Framework Multi-Asset Baseline Test Case

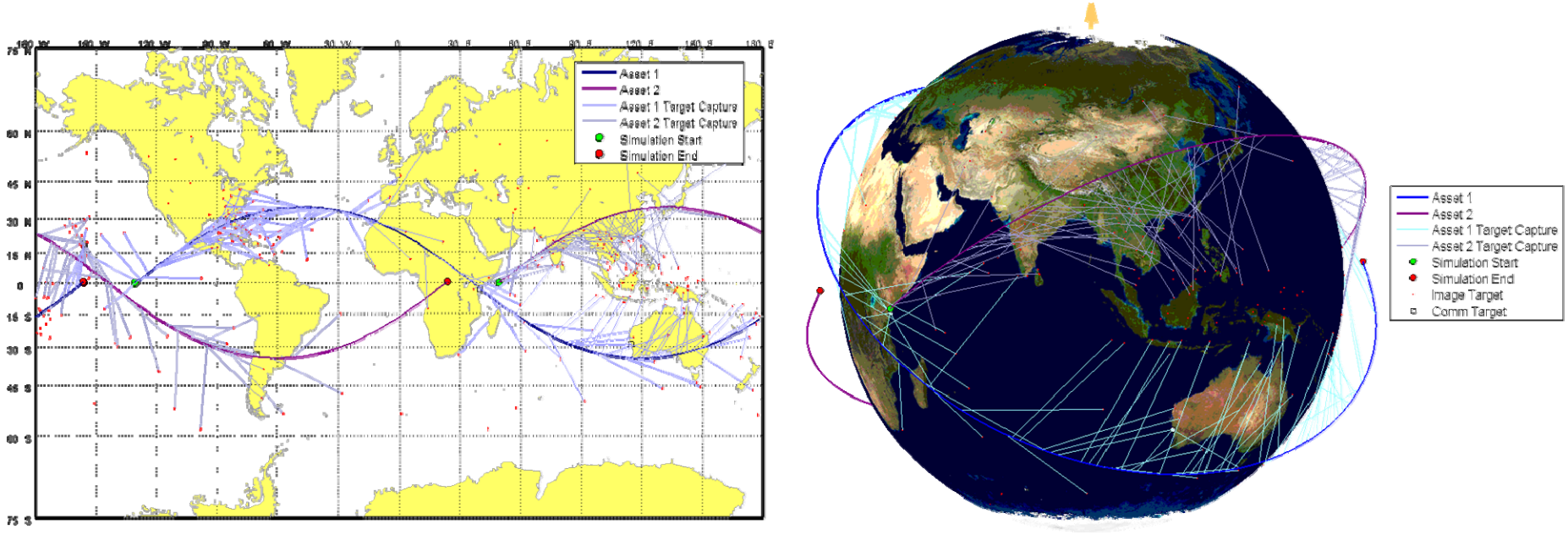
CAL POLY
SAN LUIS OBISPO

AEOLUS MISSION CONCEPT



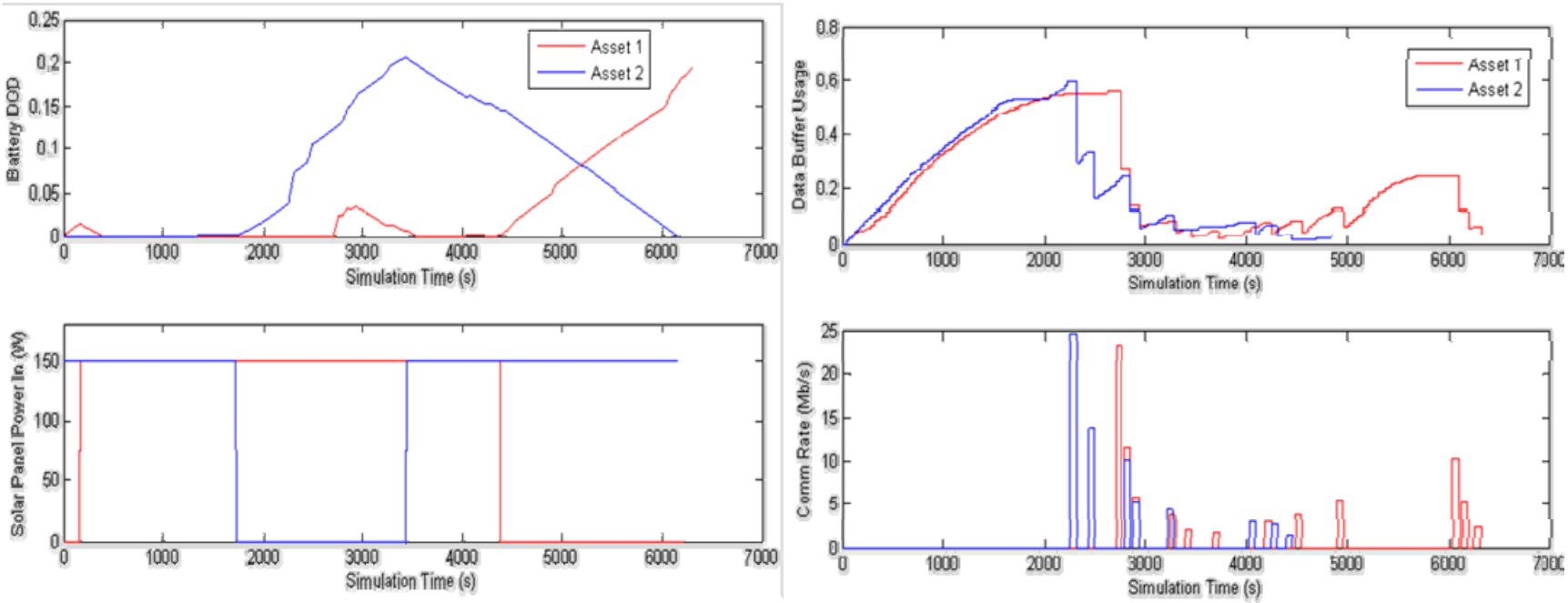
A day-in-the-life of the Aeolus constellation

- Aeolus: The Greek god of wind
- Extreme-weather imaging satellite
- Circular, 1000km, 35 degree inclined orbit
- Simulation date: August 1st 2008 for 3 revolutions
- Targets clustered into high-risk areas, including Southeast Asia and the Gulf of Mexico
- Sensor has ability to generate data while in eclipse



Matlab visualization of *HSF* simulation results

THE AEOLUS CONSTELLATION: RESULTS



Typical day-in-the-life state data

THE AEOLUS CONSTELLATION: RESULTS

CAL POLY
SAN LUIS OBISPO

FUTURE PLANS

- Drag-and-Drop Simulation Creation GUI
  - PICASSO – Version 1.2 Released
- Automatic "Sanity-Checking" for User-Specified Code
- System Model Failure Recording
  - Understanding why can't a subsystem respond to a request will help the analyst reconfigure the system or requirements
- Module Library Creation (Open Source?)
  - Underway

QUESTIONS FOR BOEING

CAL POLY
SAN LUIS OBISPO

DISCUSSION AND THOUGHTS ON:

- Model Based Systems Engineering (MBSE)
- System Modeling Language (sysML)
- Current needs from academia (or Cal Poly)
- Any projects you can share?
- What topic(s) are fundamental for systems engineers you hire?
- What topics would be appropriate for a specialized class in modeling and simulation?
- Moving forward – opportunities for collaboration?