Activity Planning with Resources for the Exploration of Space

Introducing APRES 2.0

John L. Bresina
NASA Ames Research Center

Activity Planning with Resources for the Exploration of Space

Introducing APRES 2.0

John L. Bresina
NASA Ames Research Center

- 1988 – 1998: NASA Contractor
- !998 – present: NASA Civil Servant
- Mission Involvement
  - **MER:** GDS/MOS Tactical Activity Planner
  - **MSL:** GDS only
  - **LCROSS:** GDS/MOS Activity Planning & Sequencing Lead
  - **LADEE:** GDS/MOS Activity Planning & Sequencing Lead
  - **HelioSwarm**: GDS/MOS since Proposal/Phase A

APRES Project Intro

- *Parent Project:* The AES Autonomous Systems and Operations (ASO) Project
- ASO Organizational Responsibility
  - *Responsible Mission Directorate:* Exploration Systems Development Mission Directorate (ESDMD)
  - *Lead Center:* Ames Research Center (ARC)
  - *Responsible Program:* Mars Campaign Office
- ASO Program Management
  - *Program Director:* Christopher L Moore
  - *Program Manager:* Greg Chavers
  - *Project Manager:* Jeremy D. Frank
- APRES Systems
  - **APRES Prototype:** Available in the NASA Software Catalog (entry ARC-18869-1) since August 2023
  - **APRES 2.0:** Started work at beginning of November 2022, and is still a work in progress
    - **In use on HelioSwarm Mission (Phase A & B)**
- Objectives
  - Design and implement the Ames Intelligent Systems Division's next generation mixed-initiative mission planner for ground operations
  - Design to support all missions, and in particular, multi-spacecraft missions; e.g., swarm missions
- APRES 2.0 Team
  - *Project Lead & Design Lead:* John Bresina
  - *Software Development Manager:* Hassan Eslami
  - *APRES Service Team:* David Smith, Elif Kurklu, Vijayakumar Baskaran; *along with support from J Benton*
  - *APRES User Interface Team:* Deep Tailor, Hassan Eslami

APRES 2.0 Architecture

APRES 2.0 Interface
Browser-based, using the REACT framework

Spring Boot API with JSON content

APRES 2.0 Data Store
Directory structure with JSON and ANML files

APRES 2.0 Service

- ANML Parser
- Project Validator
- Episodic Plan Simulator (EpSim)
- Constraint Checker
- STN Violation Resolver
- Mutex & Threat Analyzer
- File Manager

ANML Editor
Browser-based

Service Components

- ANML Parser
  - Validates model and creates parse data structure used by Service components
  - Creates Activity Dictionary used by User Interface and supports the creation of initial assignments and default configuration files.
- Project Validator
  - Determines if loaded projects have valid syntax and if all associated files are compatible with the domain model
  - Also performs semantic checks (e.g., activities must be contained within the plan boundaries)
- Constraint Checker
  - Detects if user's temporal constraints are inconsistent
  - Detects temporal constraints that are not satisfied in the plan
- Episodic Plan Simulator (EpSim)
  - Simulates plan, creating numeric and state chronicles
  - Detects and reports violations
  - Determines process triggering and termination
- STN Violation Resolver
  - Determines action rescheduling resolutions, using a Simple Temporal Network & greedy search
- Mutex & Threat Analyzer
  - Supports STN construction
- File Manager
  - Retrieves and saves files in the APRES Data Store
  - Maintains integrity of the APRES Data Store

Planning Project

- Project information
  - Author
  - Last modified timestamp and last valid timestamp
  - Plan start and end timestamps
  - Action start time precision (Milliseconds, seconds, or minutes)
  - Associated file references (model, initial & timed assignments, configuration)
- Activity Plan
  - Actions and processes
  - User-defined temporal constraint (unary and binary)
- Simulation Information
  - Violations detected
    - variable-bounds
    - unsatisfied-condition
    - violated-condition
    - inconsistent-effect
    - duplicate actions
    - temporal-constraint
    - inconsistent-constraints
  - Numeric chronicles
  - State chronicles
- Resolution Suggestions
  - Rescheduling of action instances

ANML II Summary

- ANML II is a very expressive language, enabling more **accurate domain models** and **plans**
  - APRES 2.0 is only using a subset of the ANML II language; e.g., we currently do not support decomposition, conditional effects, quantification, existential temporal qualifiers
- ANML II allows the definition of **processes** which autonomously occur due to world states.
  - *For example, charging of the battery from the solar panels, or the operation of survival heaters that automatically power on and off based on temperature sensor readings.*
- ANML II allows the specification of a rich set of **flight rules**
  - *For example, the battery state-of-charge must remain above a specified minimum, or the instantaneous power draw must remain below a specified maximum.*
- ANML II allows the specification of **predicted behaviors** over time
  - *For example, eclipse periods, periods when a spacecraft is in view of a particular Deep Space Network Station, and range of spacecraft from Earth*
- ANML II allows the specification of a rich variety of activity/resource effects, including the following:
  - Absolute or incremental
  - Discrete or **continuous**
  - Resource consumption or production of reusable or consumable resources
- ANML II allows functional fluents; e.g., thrustersActive(sc) where sc is a variable whose value is a member of the set of spacecraft
  - Greatly reduces the size of the domain model, especially for multi-spacecraft missions

Demonstration Model

- Small, simplified model based on a *swarm mission*
- *Multi-spacecraft:* one hub s/c and three node s/c (n1, n2, n3)
- *Communication Policy:*
  - The nodes only communicate with the hub
  - The Ground only communicates with the hub
  - The hub can only communicate with one node at a time
  - The hub can communicate with the Ground and a node simultaneously
- *Two science instruments per spacecraft:* MAGa and MAGb
- *Five telemetry sources per spacecraft: s/c bus HK, MAGa HK, MAGa Sci, MAGb HK, MAGb Sci*
- *Three DSN stations (one per complex):* DSS34, DSS45, DSS55
- *Numeric resources modeled:*
  - Total (instantaneous) Power
  - Data to be downloaded on each spacecraft
  - Data received on the ground
- *Resource Flight Rule:* Total Power must not exceed 400 Watts
- There are a number of mutual exclusion flight rules, including:
  - The hub cannot communicate to a node or the Ground when its thrusters are active
  - A node cannot communicate to the hub when its thrusters are active

Demo Activity Types

- Action Types
  - Hub_Unload_Momentum
  - Node_Unload_Momentum
  - **Ground_Comm**
  - **Crosslink**
  - Node_Maneuver
  - Hub_MAGa_PwrOff
  - Hub_MAGa_PwrOn
  - Hub_MAGb_PwrOff
  - Hub_MAGb_PwrOn
  - Node_MAGa_PwrOff
  - Node _MAGa_PwrOn
  - Node _MAGb_PwrOff
  - Node _MAGb_PwrOn
- Process Types
  - Hub_MAGa_Operate
  - Hub_MAGb_Operate
  - Node_MAGa_Operate
  - Node_MAGb_Operate
  - **Comm_Xmit**
  - **CL_Xmit**
  - nodeDevData
  - hubDevData

ANML Action Example


```
action Ground_Comm (Station dss, PosReal commDur ::("default", "14400.0"), PosReal
            GCSdataRate ::("default", 100.0) ::("units", "kbps")  )
            ::("legend", "Ground Comm") :("keyParams", "dss")
{       duration := commDur ;
        [start] DLrate := GCSdataRate ;
        [all] {  TotalPower :uses GCSpwr ;
              GroundComm_Active == false := true ==> := false ;
              inview(dss) == true ;
              allocated(dss) == true ;
               hubThrust_Active == false ;
               radioGCS == Rx := RxTx ==> := Rx ;
             }
}
```

Annotations (in blue) are used by the UI

[start] TotalPower :+= GCSpwr;
[end] TotalPower :-= GCSpwr ;

radioGCS must initially equal Rx; at the action start it is set to RxTx and this value must be maintained until the action end; at which point it is set to Rx.

ANML Process Example


```
process Comm_Xmit () ::("legend", "Data Xmit")
  {
  [start] {
            GroundComm_Active == true;
            hubDataTBDL >= DLrate * minXmitDur ;
             }
  [all] {
            hubDataTBDL :-= #t * DLrate ;
            receivedData :+= #t * DLrate ;
             }
 (all] {
            GroundComm_Active == true;
            hubDataTBDL >= 0.0 ;
            }

  }
```

Process is **triggered** when all of these start conditions are **true**

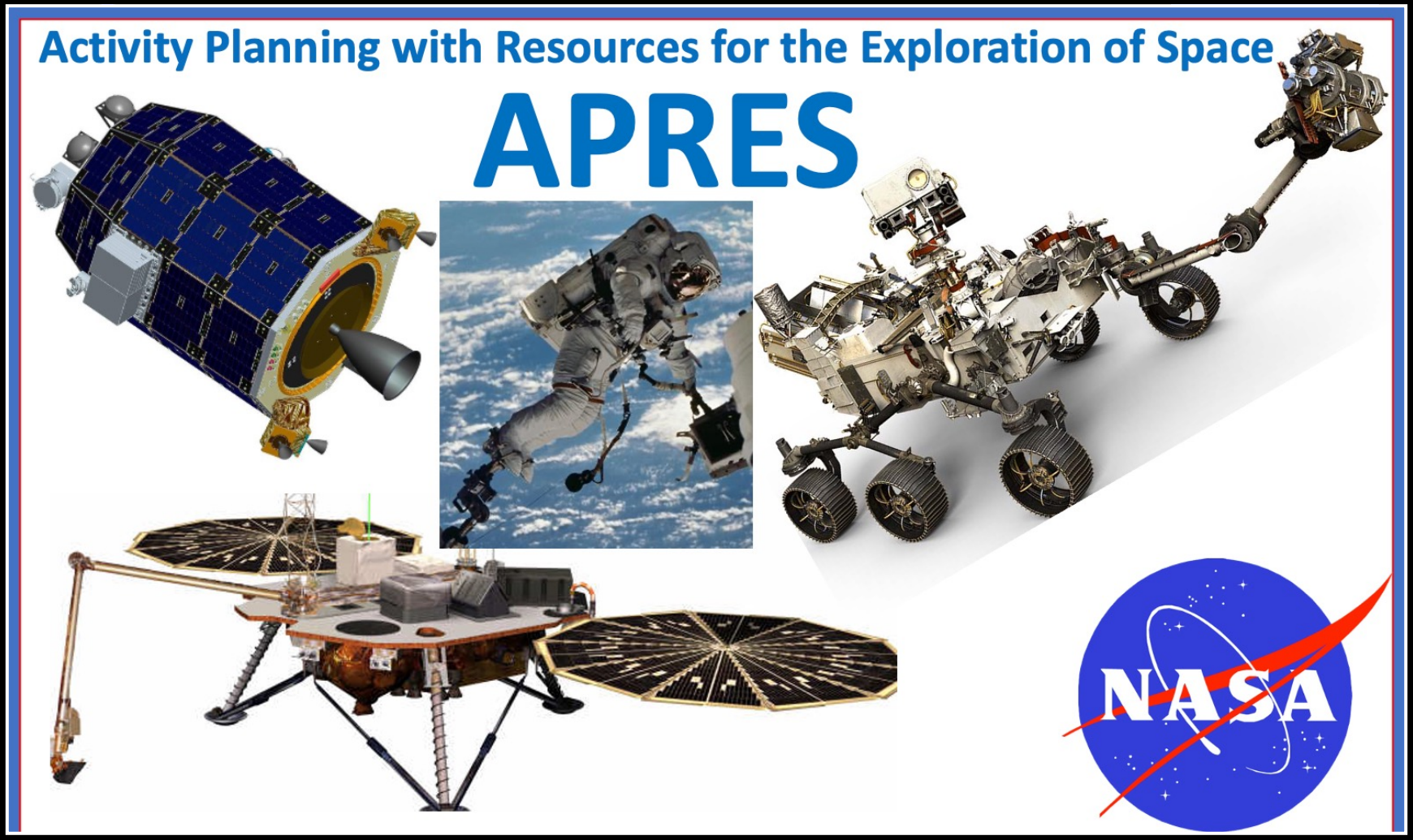
Continuous effects

Process is **terminated** when any of these conditions is **false**

Data Generation Process


```
process hubDevData () ::("legend", "Data Generate")

{

   [all] { hubDataGen(hubBus, HK) + hubDataGen(MAGa, HK) + hubDataGen(MAGa, SCI)
            + hubDataGen(MAGb, HK) + hubDataGen(MAGb, SCI) > 0;

          hubDataTBDL :+= #t * ( (hubDataGen(hubBus, HK) * DataRate(hubBus, HK)) +
            (hubDataGen(MAGa, HK) * DataRate(MAGa, HK)) + (hubDataGen(MAGa, SCI) *
            DataRate(MAGa, SCI)) + (hubDataGen(MAGb, SCI) * DataRate(MAGb, SCI)) ) ;

            }
}
```

- *This process determines the data generated on the Hub s/c based on all of its active telemetry streams (housekeeping, HK, and science, SCI)*
- *The hubDataGen functional fluent has the value 1 when the specified telemetry stream is active and 0 if it is inactive.*



Demonstration of APRES 2.0

Configurable aspects of UI

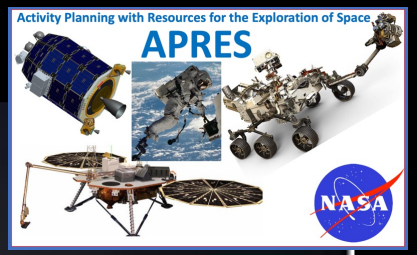
- Defaults for aspects not explicitly specified
- Color of action/process
  - Index: AD type or reusable Color Function w.r.t. **key** parameters
    - One or more action/process parameters can be designated as "**key**" parameters
  - Color for each **key** parameter value
  - Example: nodeColor(nx), where nx is the name of the node parameter in model, with values n1, n2, n3
- Colors for a state chronicle
  - Index: Fluent name or Regular Expression
  - Color for each state value
- Color of numeric chronicle
  - Index: Fluent name or Regular Expression
- Whether to show labels for a state chronicle
- Legend of action/process
  - Index: AD type or Regular Expression
- Hide Boolean
  - Legends
  - Numeric Chronicles
  - State Chronicles
- Layout Order
  - Legends
  - Numeric Chronicles
  - State Chronicles

Enhancements Since Demo

- Developed new concept of “observations”
  - Used in place of “Initial/Final Assignments”
  - The effect of an observation occurs immediately and not an epsilon later as for assignments
- Integration of Plan Transition
  - Supports transition between successive overlapping tactical plans
  - Transition time is the end time of last Ground Contact before next planning session
  - Currently, telemetry values for model variables are based on the observations from simulation
  - Initializes the new planning projects with suffix of previous plan
- Import of externally generated Timed Data, used for automatic generation of Timed Assignments
- Import of externally generated subplan (actions and constraints)
- Plan Table View (an alternative to the timeline view)

Future Enhancements

- Multi-user capabilities
  - Enforces user roles (e.g., Manager, Editor, User)
  - Enforces that a plan can only be edited by one authorized user at a time
  - Allows multiple users to watch a plan editing session
  - *Capabilities have been designed and partially implemented*
- Violation resolution based on mathematical programming/optimization
  - Exploring different approaches, but currently focusing on Mixed-Integer Linear Programming employing the open-source SCIP s/w
  - This approach will resolve a wider scope of violations due to its larger (and more disjunctive) solution space
  - It can be extended to handle **numeric** (e.g., resource) violations
  - We plan to investigate its use in some limited plan optimization
  - *Design and implementation are in progress*



Questions?
Thank you!