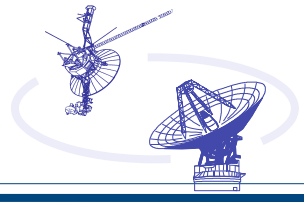
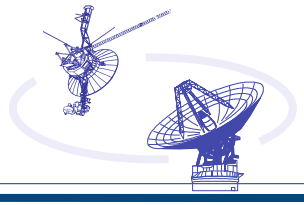


VML 3.0 Sequencing Technology for GN&C:

Utilizing advanced sequencing to simplify navigation operations and enhance science return

Dr. Christopher A. Grasso
Blue Sun Enterprises
christopher.a.grasso@earthlink.net
(720) 394-8897

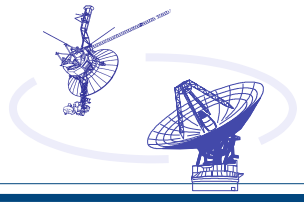
Introduction



- Spacecraft GN&C, operations, and science collection depend on sequencing
- Interpreted VML scripts provide compact specification for real-time behavior
  - much smaller and cheaper than direct low level flight software solutions (e.g. C, C++)
  - execute in a "safe sandbox" to prevent common coding mistakes and operator errors
  - easily changed and updated, highly visible
  - require less review and scrutiny than flight software
- Deep space mission issues
  - distance drives need for updateable on-board automation
  - light speed communications delay requires on-board responses to local conditions
  - minimal opportunity for ground intervention during critical phases
  - science activities depend on local conditions (e.g. target data)
- Sequencing architecture profoundly affects spacecraft operation and science collection
- Sequencing is a *cross-cutting* capability

Sequencing technologies enable resilient spacecraft operations

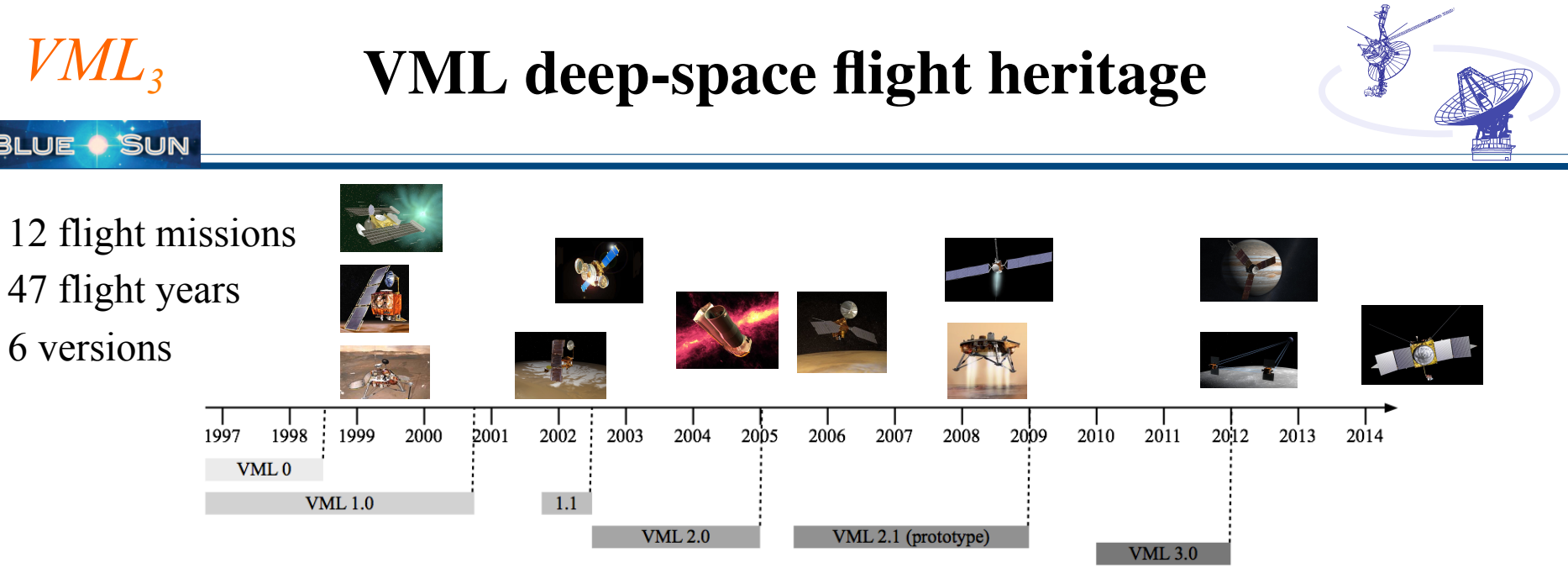
Spectrum of sequencing

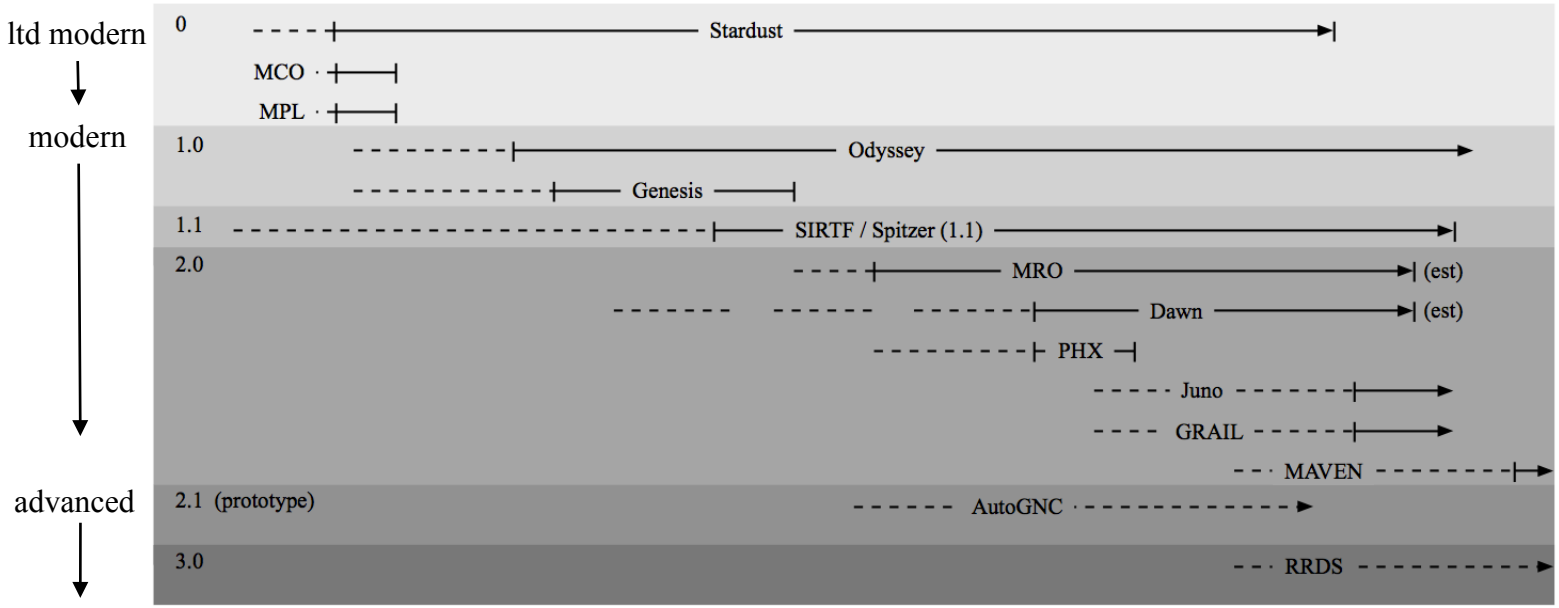


- Traditional sequencing: time-ordered commands
  - simple ground expansion of all actions
  - large product size requires frequent uplinks, high data rates
  - little or no on-board decision-making capability
  - proven largely insufficient for deep space missions
- Modern sequencing: language constructs akin to workstation languages
  - logic guides activities
  - timing determined by conditions present
  - reusable elements reduce required uplink, compatible with infrequent / low bit rate contacts
- Advanced sequencing: distributed, coordinated decision-making
  - reactive
  - decision-making constructs (e.g. state machines)
  - fault detection / response
  - replanning capability

Sequencing technologies enable robust spacecraft operations

VML deep-space flight heritage



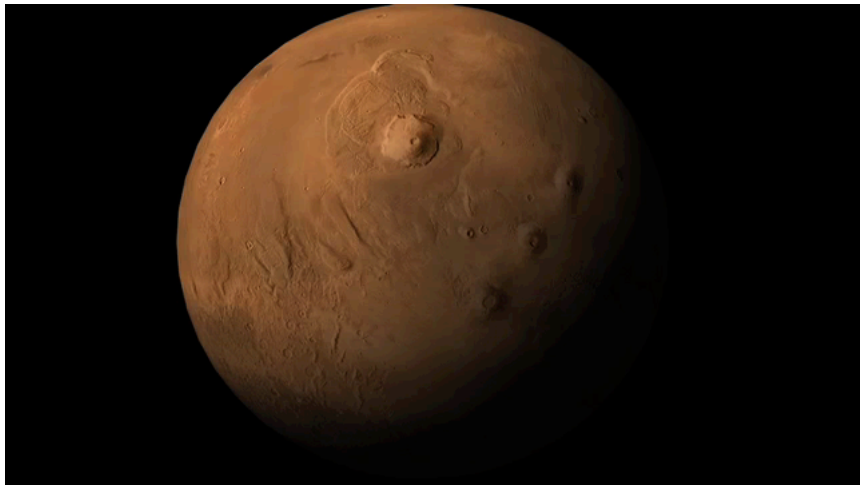


Evolution, not revolution

Complex sequencing: Entry, Descent and Landing

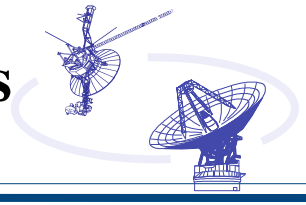
BLUE SUN

- 601 commands, 4 days, 17000 kph
- Hit ellipse 60 km x 20 km after trip of 470 million km
- Must work perfectly once to land Phoenix on Mars
- Throw away parts of the vehicle: cruise stage with solar arrays, X-band, star trackers
- No direct-to-earth communications after sep: use UHF relay via Odyssey and MRO
- Accommodate late reboot of spacecraft up to 900 seconds before entry
- Shift all activities relative to atmospheric density

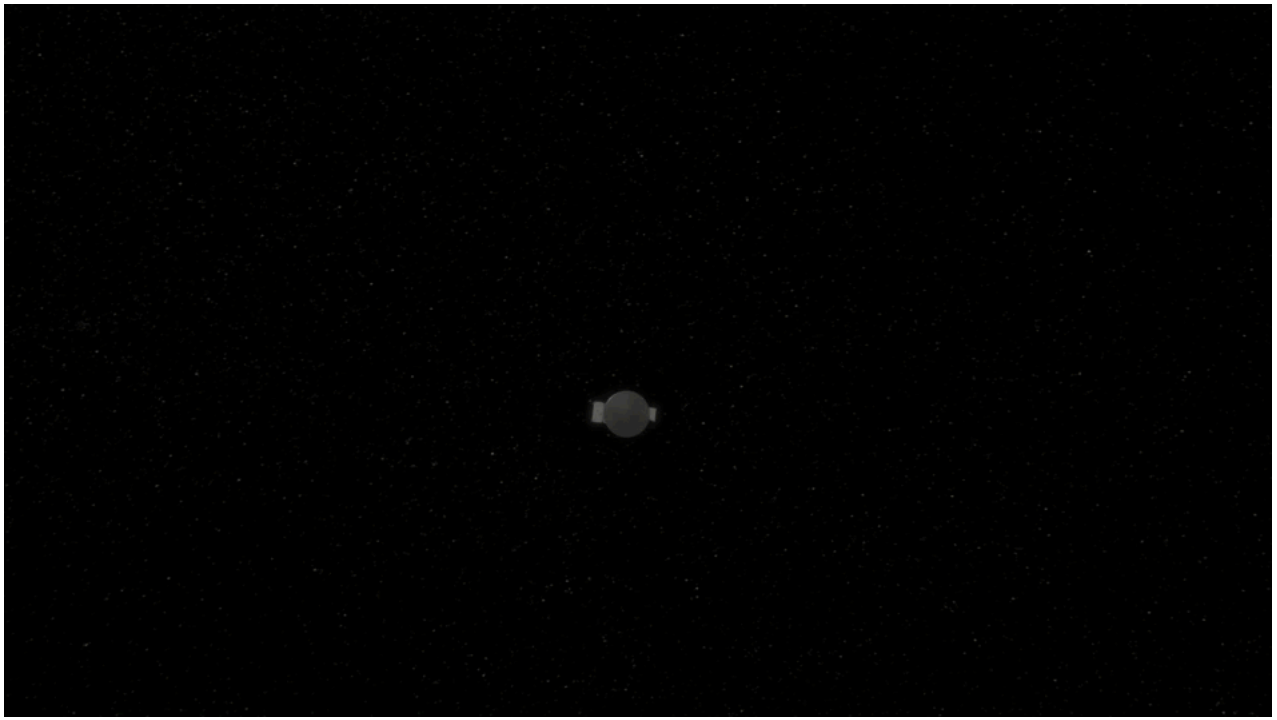


"Land or Die"

EDL sequencing highlights: 4 days of activities



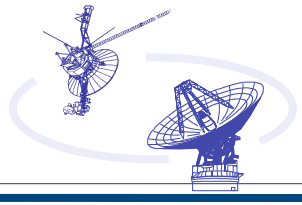
- Deactivate fault responses
- Configure thermal
- Determine acceleration bias
- Blow cruise stage
- Slew to entry attitude
- Activate hypersonic ACS
- Deploy parachute
- Prepare engines for firing
- Blow heat shield
- Deploy legs
- Turn on radar
- Drop out of backshell
- Detect touchdown
- Start landed activities



Phoenix EDL visualization

Many actions in short period of time

Commands and sequencing



- *Commands* are directives to the spacecraft with some meaning, e.g.

```
CCD_TAKE_PICTURE "narrow", 5.0
```

- Command execution timing
  - *absolute*: commands tied to absolute time, one-shot, exact timing
  - *relative*: commands tied to time offset from preceding command, reusable / shiftable
- *Sequencing* issues commands from an on-board store
  - logic and commands cause spacecraft to behave in a desired fashion
- *Calculations* are expressions evaluated within sequence to yield a result

```
gv_ccd_power := gv_ccd_voltage / gv_ccd_current
```

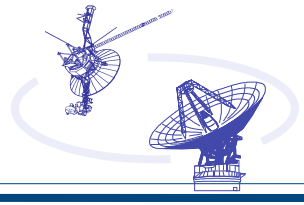
- *Conditionals* and *loops* are paths through sequence driven by truth calculations

```
if gv_ccd_power < 2.0 then ...
for i := 1 to 12 do ...
while gv_ccd_power > 2.5 do ...
```

- *Events* are conditions driven by the environment with timing that can't be predicted
  - results in *reactive sequencing*
- *Automation* is the use of reusable onboard components to perform repeated tasks

Sequencing orders on-board commands using time, logic, and events

Basic sequencing examples

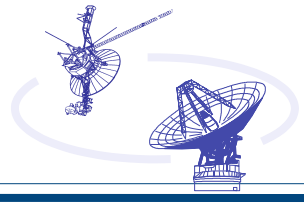


Take a 5 second exposure on a CCD [traditional]:

```
A2015-072T03:32:11.1 issue ccd_expose "narrow", 5.0
```

Absolute time simplest method, large number of statements, inflexible

Basic sequencing examples



Take a 5 second exposure on a CCD [traditional]:

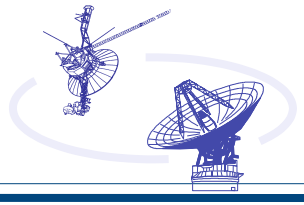
```
A2015-072T03:32:11.1 issue ccd_expose "narrow", 5.0
```

But first check if the power is on [traditional / modern]:

```
A2015-072T03:32:11.1 if gv_ccd_power > 2.0 then
A2015-072T03:32:11.1     issue ccd_expose "narrow", 5.0
A2015-072T03:32:11.1 end_if
```

Conditional check: on-board safety

Basic sequencing examples



Take a 5 second exposure on a CCD [traditional]:

```
A2015-072T03:32:11.1 issue ccd_expose "narrow", 5.0
```

But first check if the power is on [traditional / modern]:

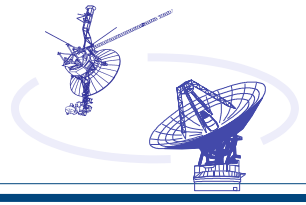
```
A2015-072T03:32:11.1 if gv_ccd_power = 2.0 then
A2015-072T03:32:11.1     issue ccd_expose "narrow", 5.0
A2015-072T03:32:11.1 end_if
```

Better yet, take a set of images on a CCD all in a row [modern]:

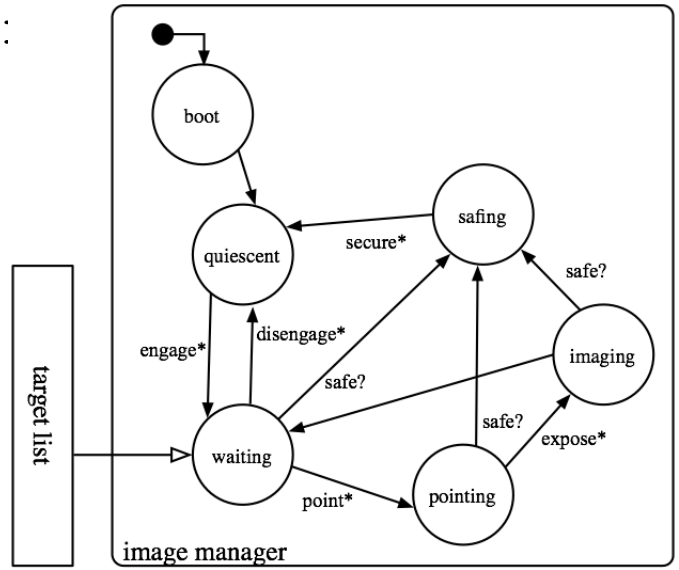
```
A2015-072T03:32:11.1 call take_pictures "narrow", 5.0, 4
      block take_pictures
          input field
          input duration
          input num
          declare i := 0
      body
          if gv_ccd_power > 2.0 then
              for i := 0 to num do
                  issue_dyamic #ccd_expose, field, duration
                  delay_by duration
              end_for
          end_if
      end_body
```

Automate action with reusable routine that accepts parameters, develop once

Reactive sequencing



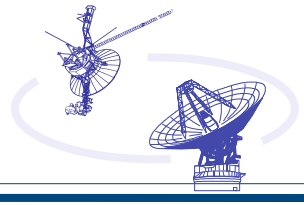
Trigger imaging with state machine [advanced]:



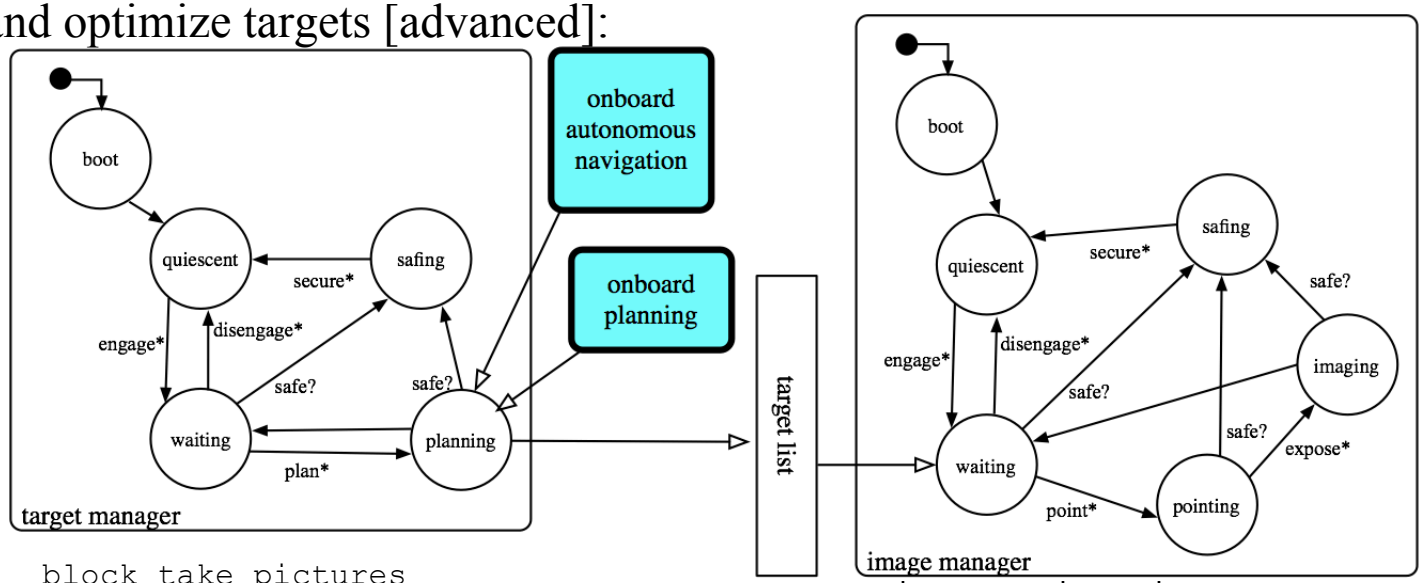

```
block take_pictures
    input field
    input duration
    input num
    declare i := 0
body
    if gv_ccd_power > 2.0 then
        for i := 0 to num do
            issue_dyamic #ccd_expose, field, duration
            delay_by duration
        end_for
    end_if
end_body
```

Onboard trigger of reusable block with targets

Reactive sequencing



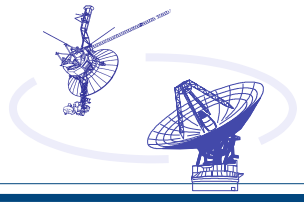
Replan and optimize targets [advanced]:




```
block take_pictures
    input field
    input duration
    input num
    declare i := 0
body
    if gv_ccd_power > 2.0 then
        for i := 0 to num do
            issue_dyamic #ccd_expose, field, duration
            delay_by duration
        end_for
    end_if
end_body
```

Onboard replan for targets of opportunity

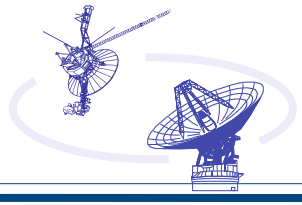
State-based approach to complex activities



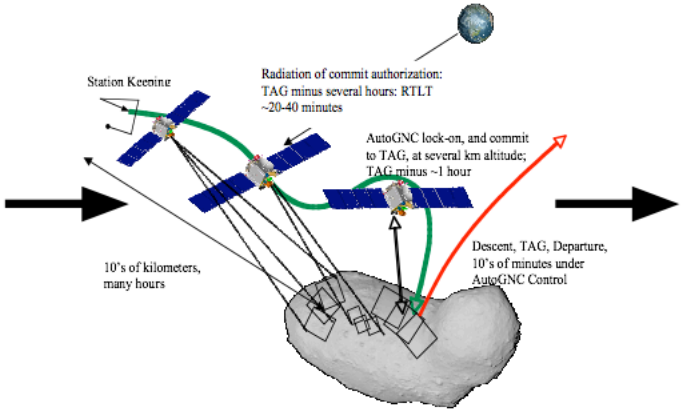
- EDL: Six parallel state machines
  - mainline
  - sideline
  - communications
  - uplink verification
  - CPU utilization
  - imaging
- Mainline progresses through 27 substates, others follow using signals from mainline
- Centralized catch-up logic, counter represents progression through substates
- States implemented as blocks spawning blocks, substates occur within blocks when signals sent
- Signal transmit: global variables set with event time
- Signal receive: WAIT on global variable

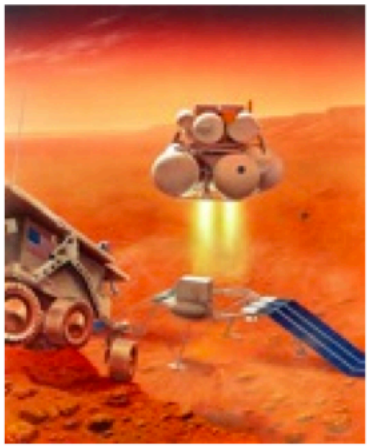
Concise division of labor, concise coordination points

Autonomous navigation



- Phoenix EDL involved very rigorous, demanding sequencing using VML 2.0
- Techniques developed EDL included state machines with one-way synchronization
- State machine concepts expanded to two-way synchronization for autonomous comet / asteroid sampling operations using AutoGNC and VML 2.1
- Sampling architecture is being further enhanced to enable autonomous rendezvous and docking for a potential Mars sample return mission using VML 3.0



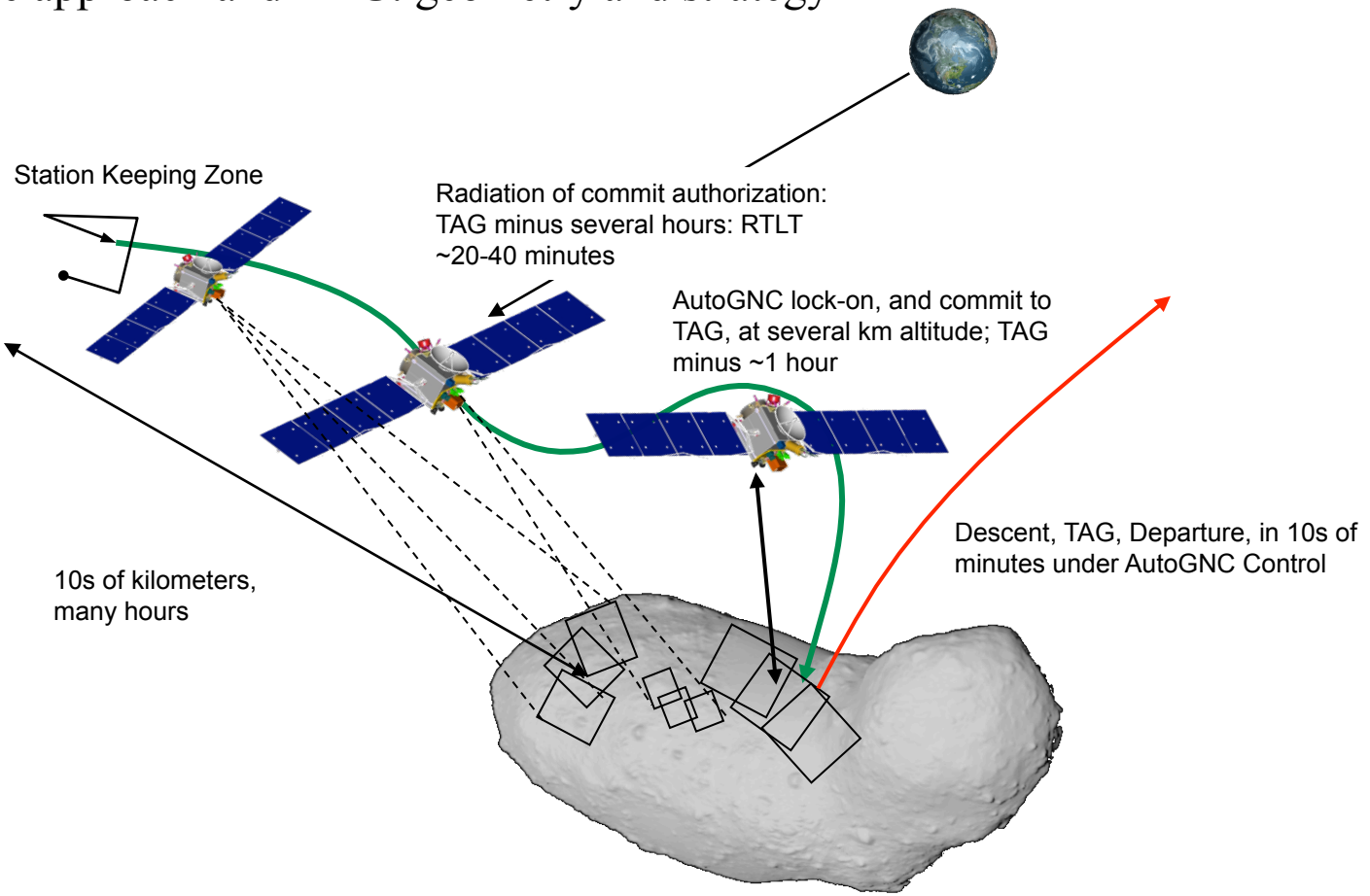


EDL TAG MSR

Build future capabilities on past success

Touch and Go (TAG)

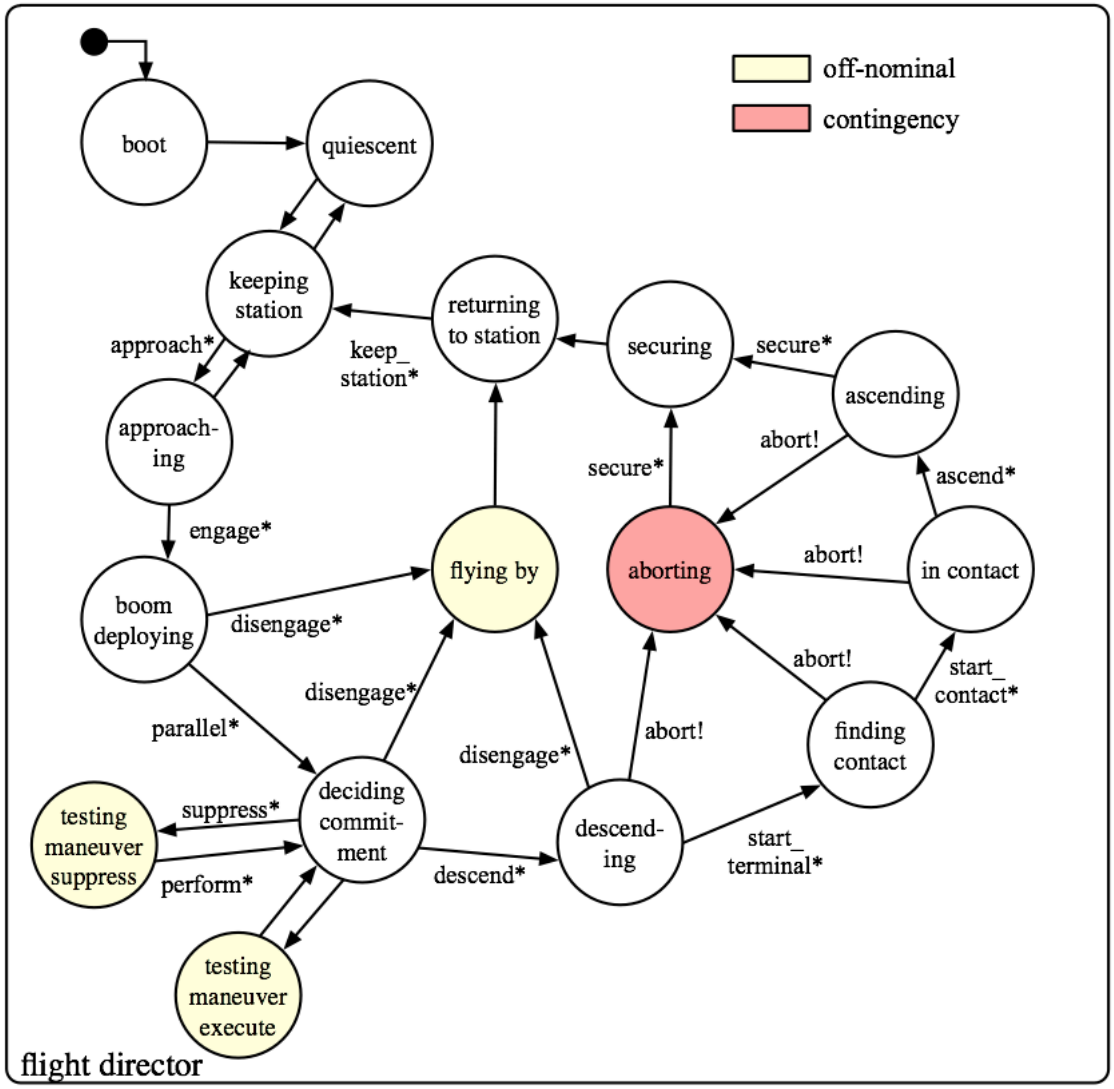
Comet sample approach and TAG: geometry and strategy



Progression is state-oriented

Flight director: high level control of TAG

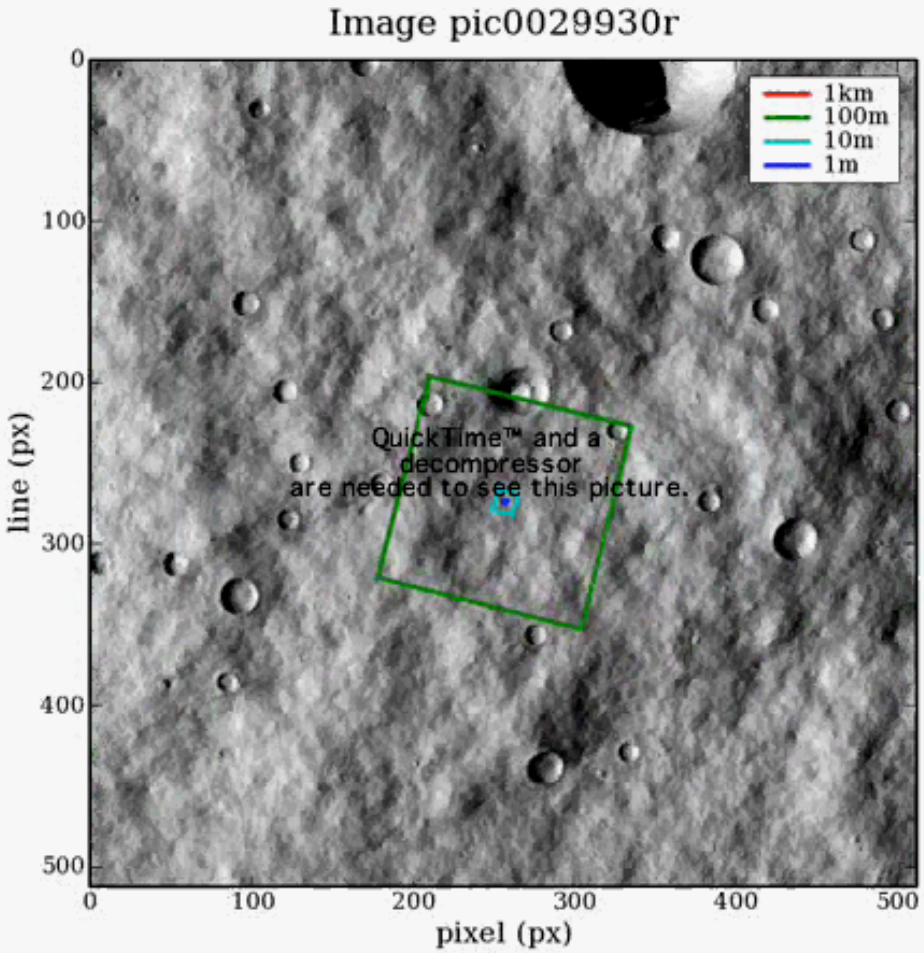
- Extension of EDL architecture
- Make complex system behavior transparent
- Manage GNC simply and directly in a high-level manner
- Protect spacecraft: "live to fight another day" vs. "land or die"
- Ground has opportunities to intervene as allowed by light-speed delay
- Spacecraft can decide to shortcut the loop and fly by or abort
- Flight director is top-level: other state machines control lower level activities



Enhanced EDL-like technique

Application: approach and contact with body

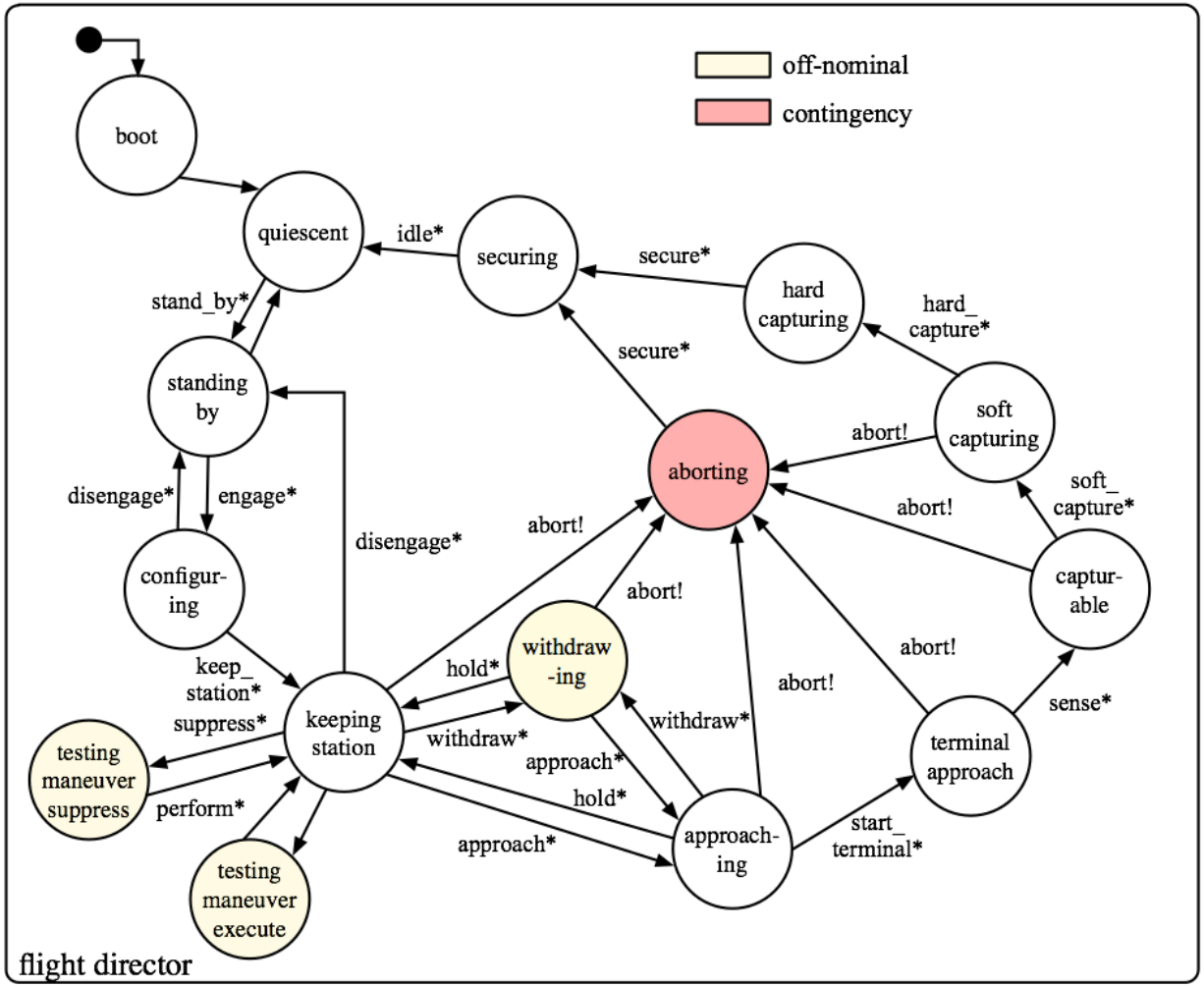
- Use executive to direct lower level spacecraft functions for approaching, sampling, ascending
- Comet, asteroid, lunar
- Tempel-1



One of several possible applications

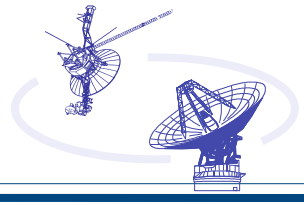
Next step: Mars Sample Return Rendezvous

- Autonomous capture of potentially passive sample canister in Mars orbit
- Flight director, manager alterations, similar architecture to TAG
- Applicable to other rendezvous situations



Modification and evolution of TAG architecture

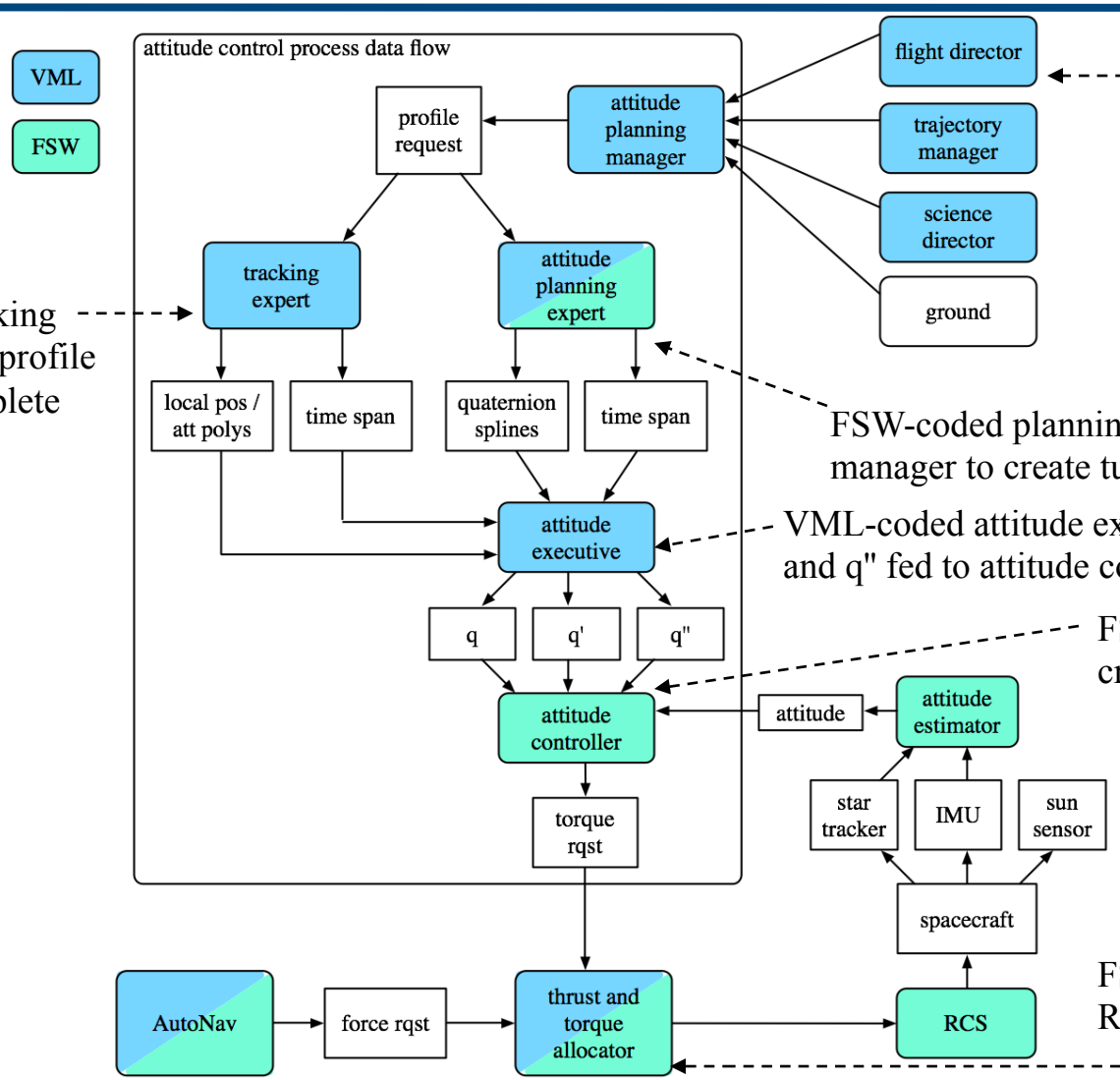
Replacing C-coded GN&C with VML



- Strictly C-code logic structure replaced with VML state-machine design
- Design for complete attitude control system includes the following elements:
  - attitude estimator: current orientation
  - attitude planning expert: plans slew, avoids exclusion zones
  - attitude controller: translates attitude requirements into torque requests
  - thrust / torque allocator: reconciles thrust and torque conflict with hardware
- C-code removed:
  - real-time coordination logic
  - data management elements (e.g. target ephemeris)
  - memory storage and access between elements
  - event registration and response
- C-code computation constructs retained as simple compute elements
  - code atomization
  - simplified compute elements
  - respond to transparent VML hierarchy
- Provides a higher level of reusability between missions

Core GN&C real-time computational elements rewritten in VML

Attitude control process



Profile requests source from expert systems for autonomous activity, ground for manual control (1)

VML-coded tracking expert maintains profile once turn is complete

FSW-coded planning guided by VML manager to create turns (2)

VML-coded attitude executive derives q, q', and q" fed to attitude controller (3)

FSW-coded attitude controller creates torque requests (4)

FSW-coded allocator drives RCS thrusters / wheels (5)

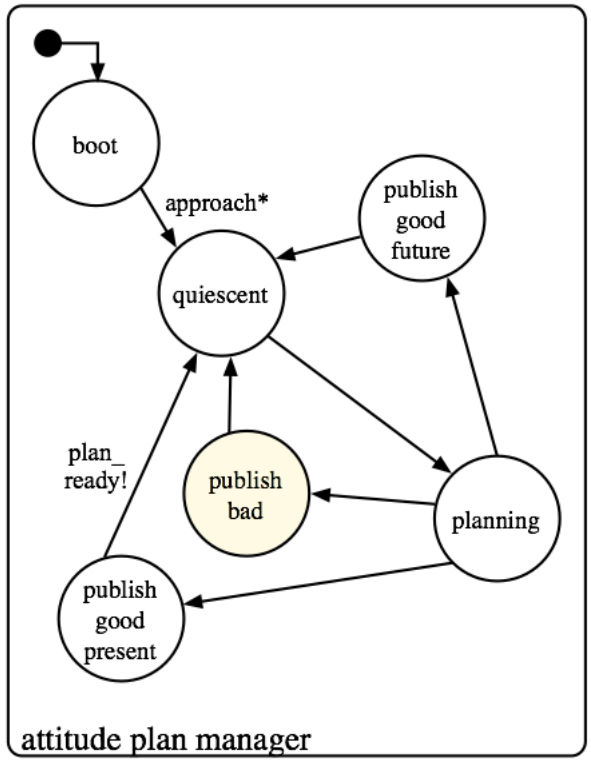
Modification and evolution of TAG architecture

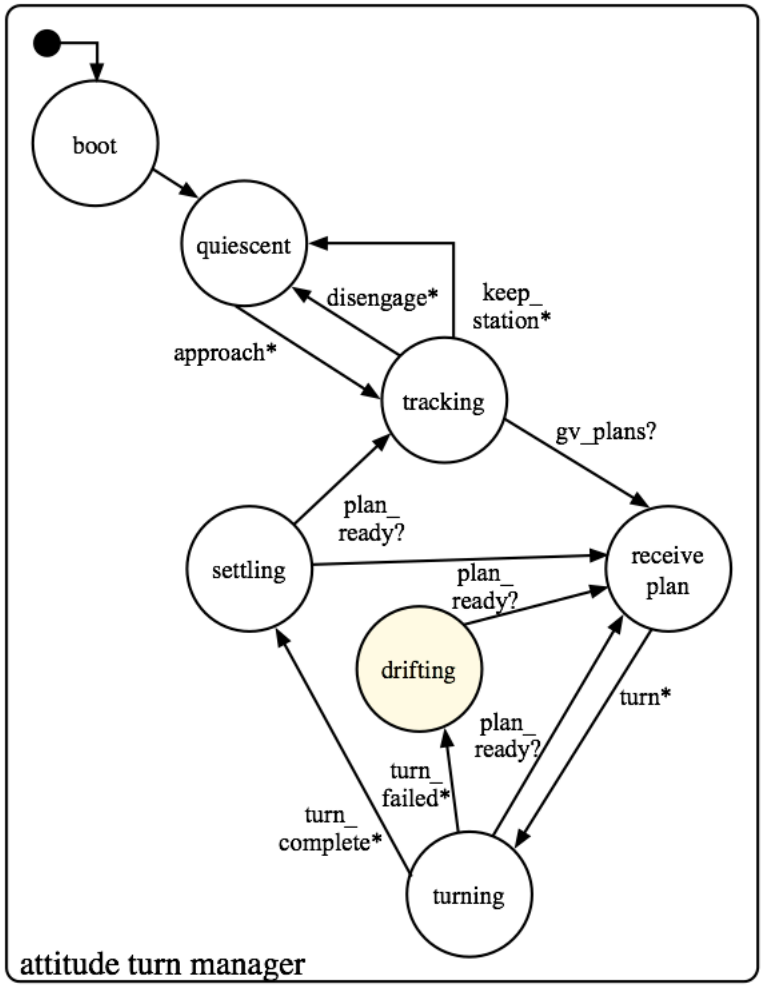
Attitude profiling state machines

Plan manager receives planning request, generates plan

Good plans published for consumption by the turn manager

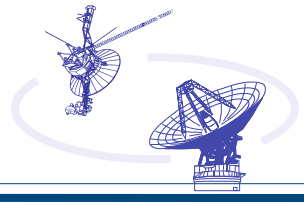
New plans accepted at any time by the turn manager, overriding current turn if underway





State machines coordinate to plan and implement turn

Matrix manipulation in VML



Collections are treated like vectors and matrices, with a set of matrix operations

- compatible sizing and data types for operations checked at runtime
- results stored back into result collection
- in-line operators and functions

Current set of operations

- arithmetic: +, -, *, /
- dot
- cross
- abs(), adjugate(), cofactor(), det(), identity(), invert(), minor(), transpose()
- trigonometry functions
- set value
- spline lookup and evaluation

Future set of operations to be implemented:

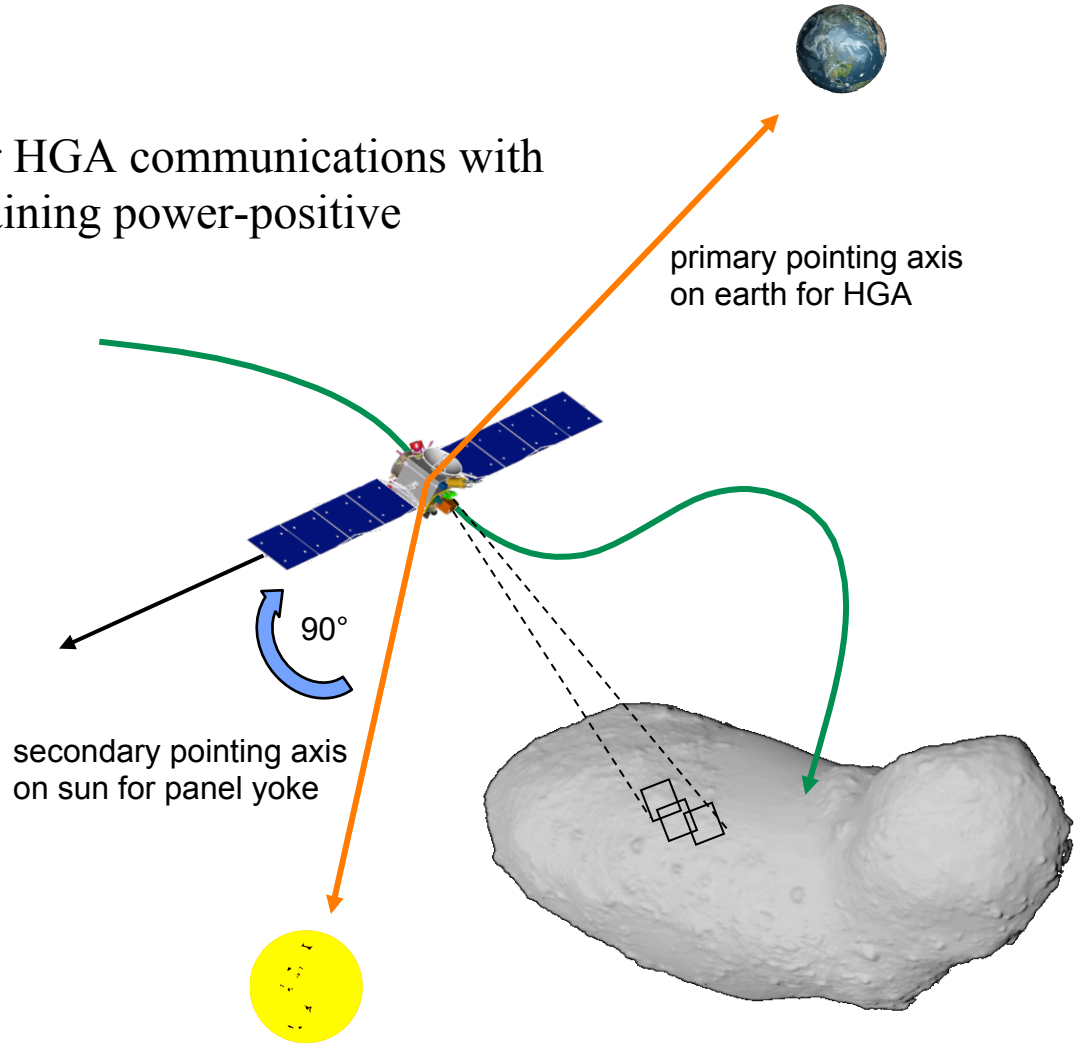
- eigenvector derivation
- trace()

User-provided functions supported by built-in VML external_call

Matrix now available in VML 3.0

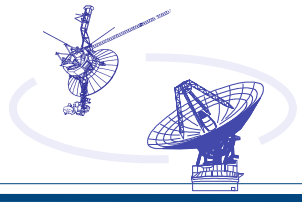
Visualization of pointing vectors

Set up attitude for HGA communications with earth while remaining power-positive



Uses VML-based attitude specification

Set attributes for attitude profiling

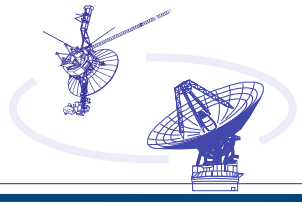


```
method set_primary_target
    input name
    ...
body
    if name = #earth then
       primary_target := gv_target_earth

    else_if name = #sun then
    ...
    end_if
end_body


method set_secondary_target
    input name
    ...
body
    if name = #earth then
       secondary_target := gv_target_earth

    else_if name = #sun then
       secondary_target := gv_target_sun
    ...
    end_if
end_body
```

```
method
data (variable or attribute)
parameter_name:
```

Targets available for ongoing processing

Set attributes for attitude profiling

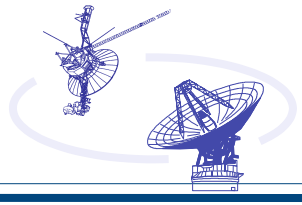


```
method set_primary_axis
    input name
    ...
body
    if name = #hga then
       primary_axis := gv_axis_hga

    else_if name = #imager then
    ...
    end_if
end_body


method set_secondary_axis
    input name
    ...
body
    if name = #panel_yoke then
       secondary_axis := gv_axis_panel_yoke

    else_if name = #x_face then
       secondary_axis := gv_axis_x_face
    ...
    end_if
end_body
```

```
method
data (variable or attribute)
parameter_name:
```

Targets available for ongoing processing

Matrix math in attitude executive

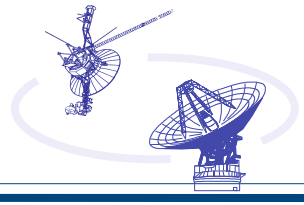


```
private_method follow_tracking_profile
    declare target_pos := vector(3)
    ...
body
    target_pos := primary_target calc_pos t: gv_current_time

    primary_sc_pos := target_pos - gv_sc_pos

    if secondary_target.name = #sun then
        secondary_sc_pos := primary_sc_pos cross gv_sun_pos

    else_if secondary_target.name = #flyby_target then
        rel_vel := gv_sc_vel - gv_target_vel
        secondary_sc_pos :=  rel_vel cross primary_sc_pos
    ...
    end_if

    ; send needed q, q', and q'' to attitude controller
    gv_desired_q := calc_q
                    primary_sc_axis: primary_sc_axis
                    primary_sc_pos: primary_sc_pos
                    secondary_sc_axis: secondary_sc_axis
                    secondary_sc_pos: secondary_sc_pos

    store_and_apply_q_derivatives
end_body
```

```
method
data (variable or attribute)
parameter_name:
```

Matrix math simple in VML

VML advantages for GNC and complex ops



- Real-time coordination between components is simple, concise, and native in VML
- Same automation mechanism across missions reduces development risk
- State diagrams greatly clarify complex operations
- Components allow easy reuse between missions
  - touch-and-go asteroid/comet sampling
  - lunar landing
  - Mars Sample Return
- VML automation and expert systems have advantages over flight software
  - cheaper to develop due to considerably more compact code
  - behavior more visible to developers and operators
  - easier to update in flight
- VML allows "safe sandbox" components to be developed without the expense, risk, and complexity of flight software

VML mission-enabling capabilities