Command and Sequencing Technology for Small Body Missions:

Utilizing advanced flight software to simplify operations and enhance science return

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

Joseph E. Riedel
Jet Propulsion Laboratory
joseph.e.riedel@jpl.nasa.gov

Andrew.T.Vaughan
Jet Propulsion Laboratory
andrew.t.vaughan@jpl.nasa.gov

BLUE SUN

Introduction

- Spacecraft actions and science collection depend on sequencing
- Sequencing applies a constrained set of actions to operate the spacecraft
  - *not* custom flight software controlling subsystems
  - interpreted scripts
  - execute in a "safe sandbox" to prevent common coding mistakes and operator errors
  - easily changed and updated
  - require lower level of review and scrutiny
- 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 technologies enable robust spacecraft operations

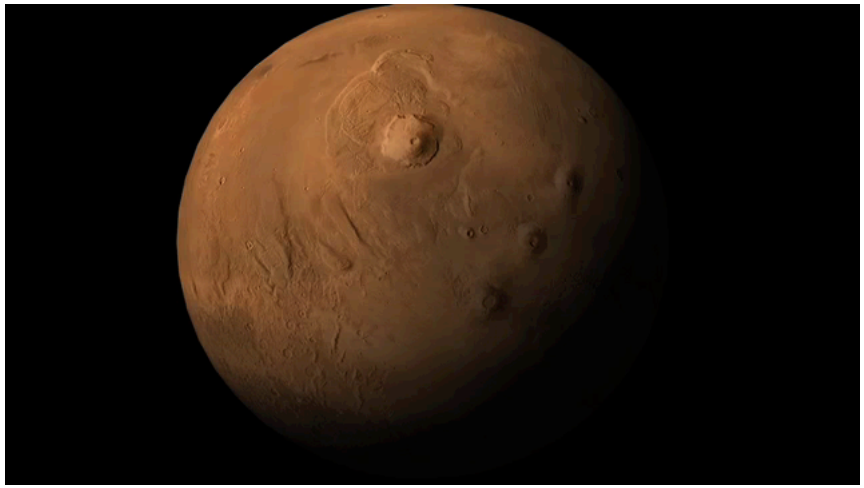
Spectrum of sequencing

- 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

Complex sequencing: Entry, Descent and Landing

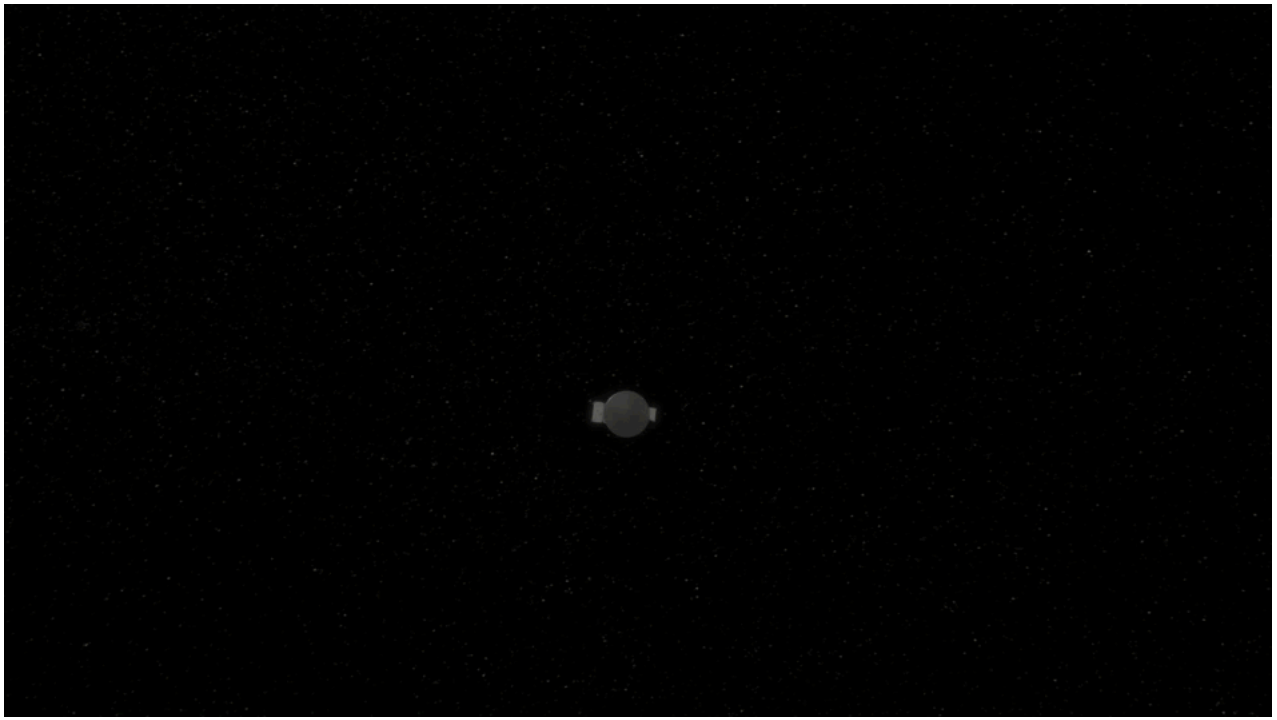
- 601 commands, 4 days, 17000 kph
- Hit ellipse 60 km x 20 km after trip of 470 million km
- Must work perfectly once to land Phoenix on Mars
- Throw away parts of the vehicle: cruise stage with solar arrays, X-band, star trackers
- No direct-to-earth communications after sep: use UHF relay via Odyssey 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 / advanced]:

```
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]:

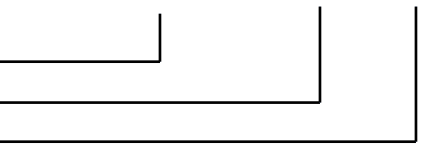
```
A2015-072T03:32:11.1 issue ccd_expose "narrow", 5.0
```

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]:

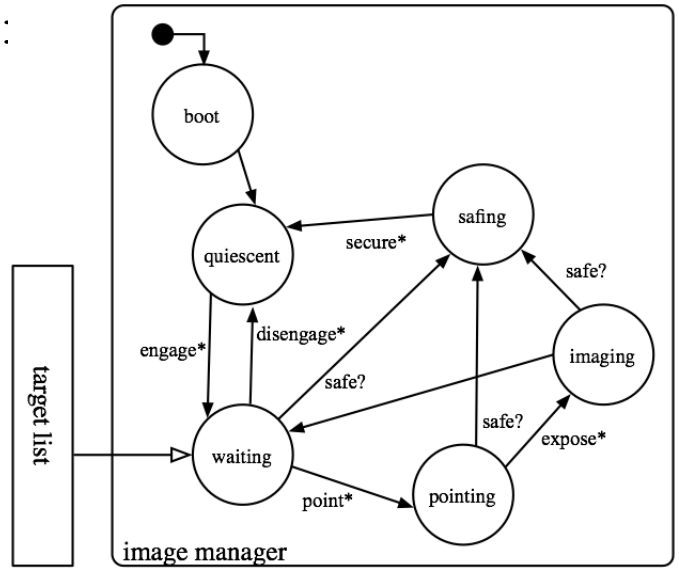
```
A2015-072T03:32:11.1 call take_pictures "narrow", 5.0, 4
      block take_pictures
          input field
          input duration
          input num
          declare i := 0
      body
          if gv_ccd_power > 2.0 then
              for i := 0 to num do
                  issue_dyamic "ccd_expose",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

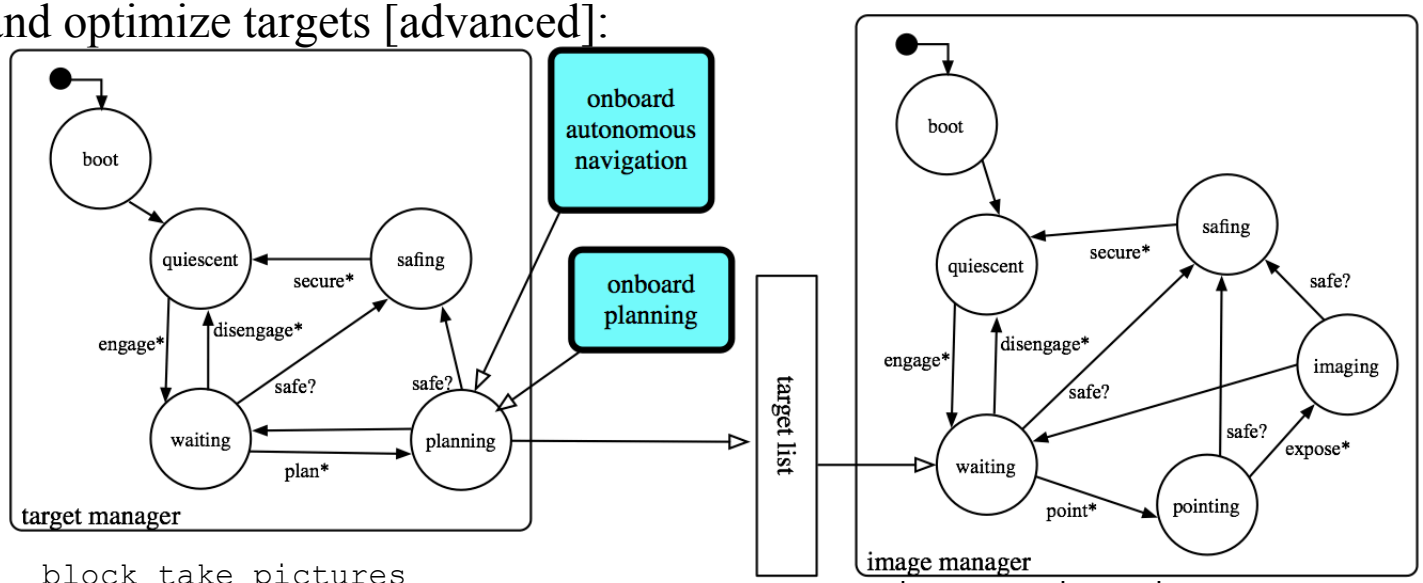



```
block take_pictures
    input field
    input duration
    input num
    declare i := 0
body
    if gv_ccd_power > 2.0 then
        for i := 0 to num do
            issue_dyamic "ccd_expose", field, duration
            delay_by duration
        end_for
    end_if
end_body
```

Onboard trigger of reusable block with targets

Reactive sequencing

Replan and optimize targets [advanced]:

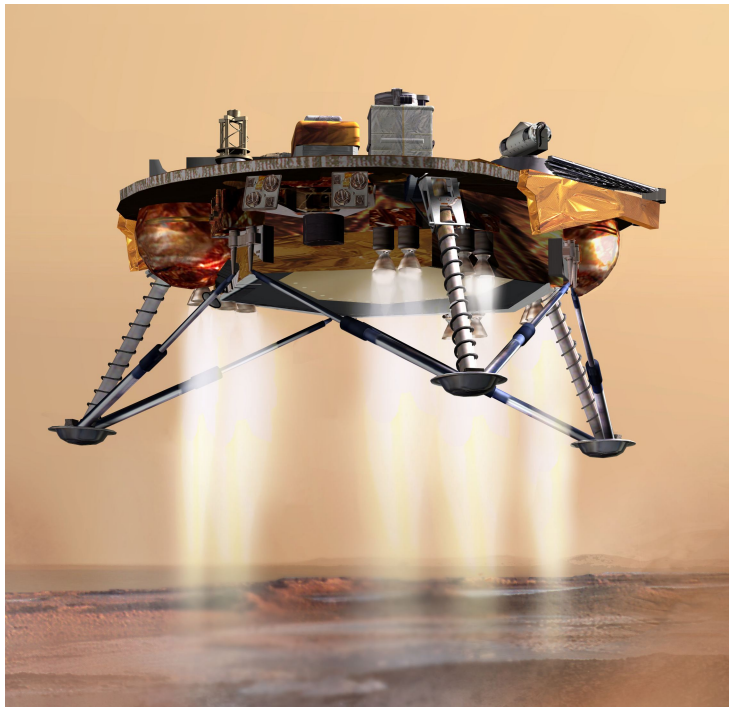



```
block take_pictures
    input field
    input duration
    input num
    declare i := 0
body
    if gv_ccd_power > 2.0 then
        for i := 0 to num do
            issue_dyamic "ccd_expose", field, duration
            delay_by duration
        end_for
    end_if
end_body
```

Onboard replan for targets of opportunity

State-based approach to complex activities

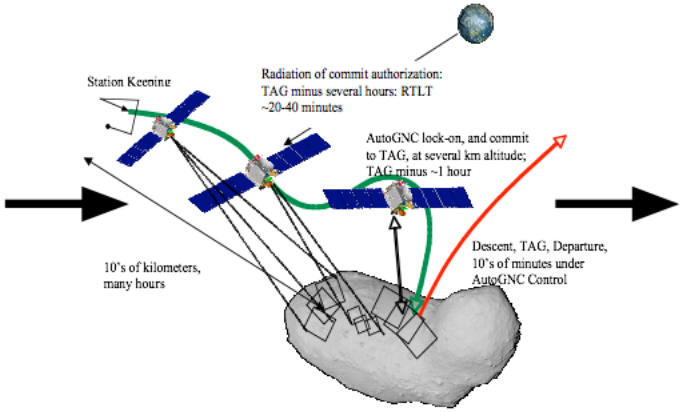
- EDL: Six parallel state machines
  - mainline
  - sideline
  - communications
  - uplink verification
  - CPU utilization
  - imaging
- Mainline progresses through 27 substates, others follow using signals from mainline
- Centralized catch-up logic, counter represents progression through substates
- 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

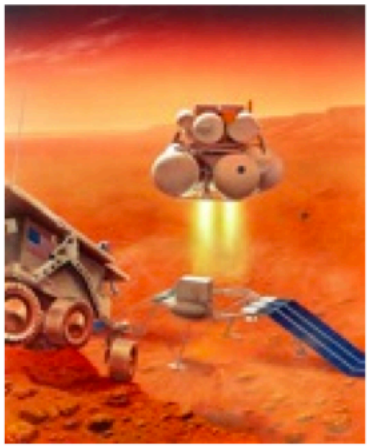


Concise division of labor, concise coordination points

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 enhanced to enable autonomous rendezvous and docking for a potential Mars sample return mission using VML 2.2



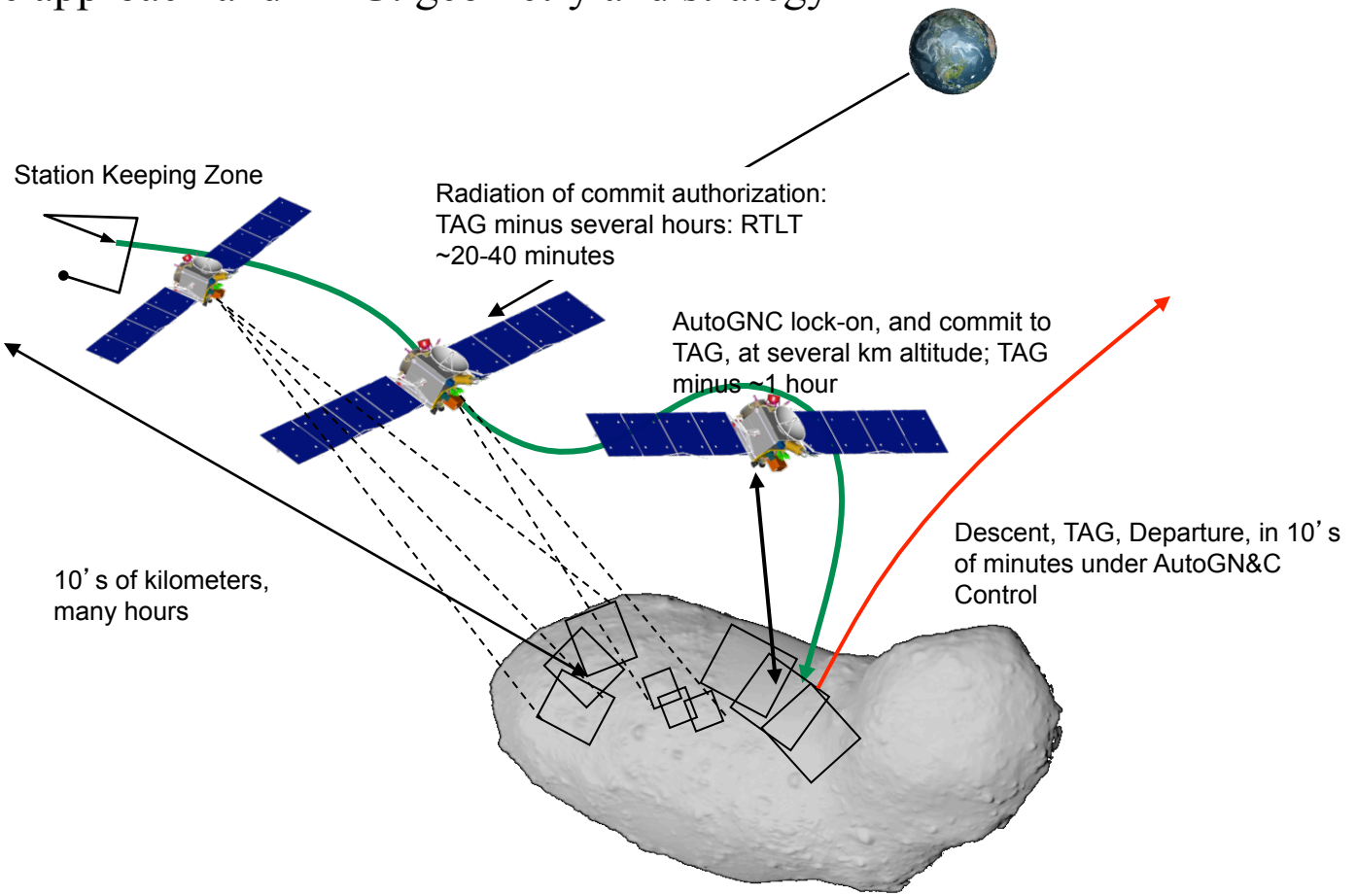


EDL TAG MSR

Build future capabilities on past success

Touch and Go (TAG)

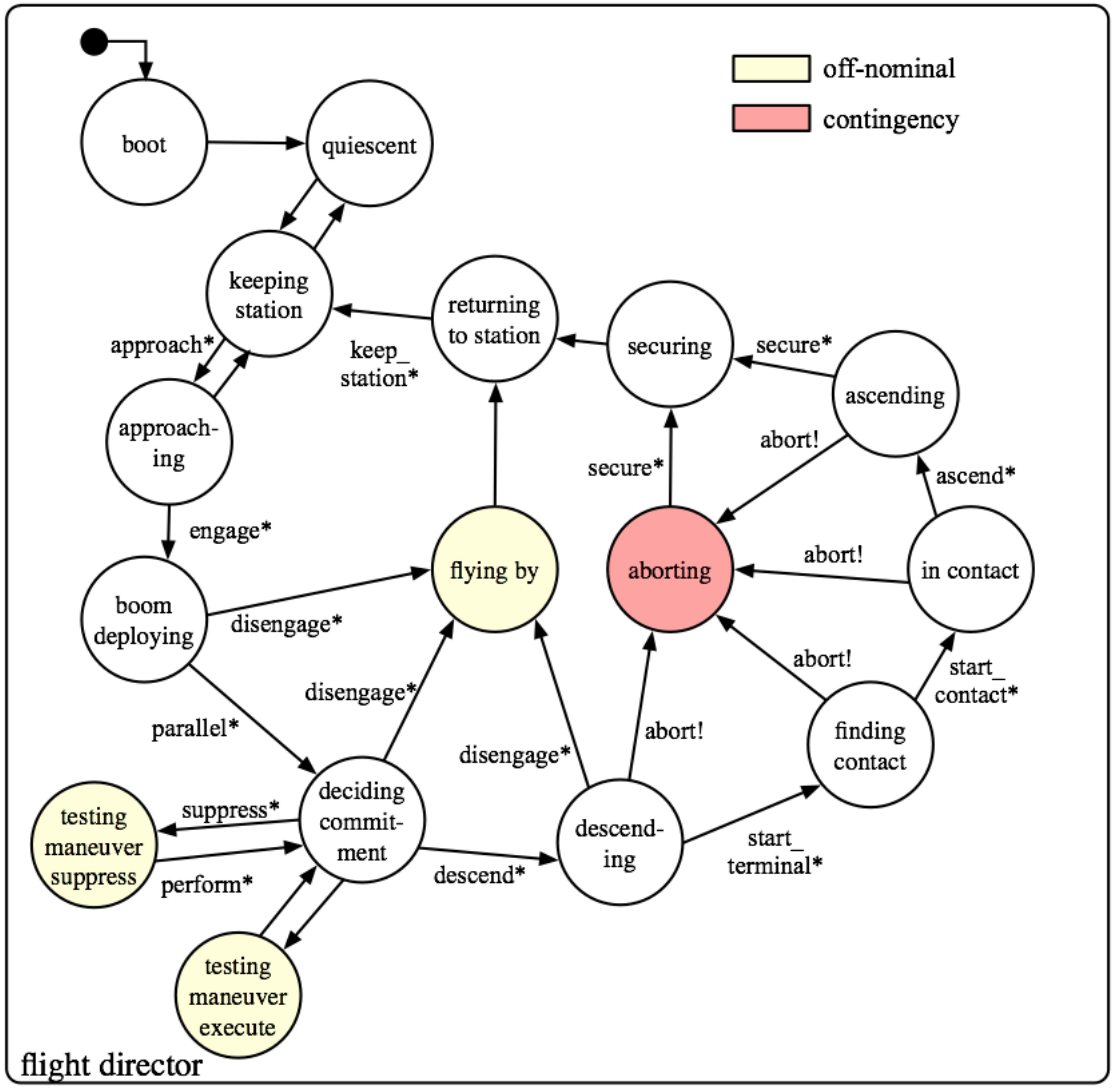
Comet sample approach and TAG: geometry and strategy



Progression is state-oriented

Flight director: high level control of TAG

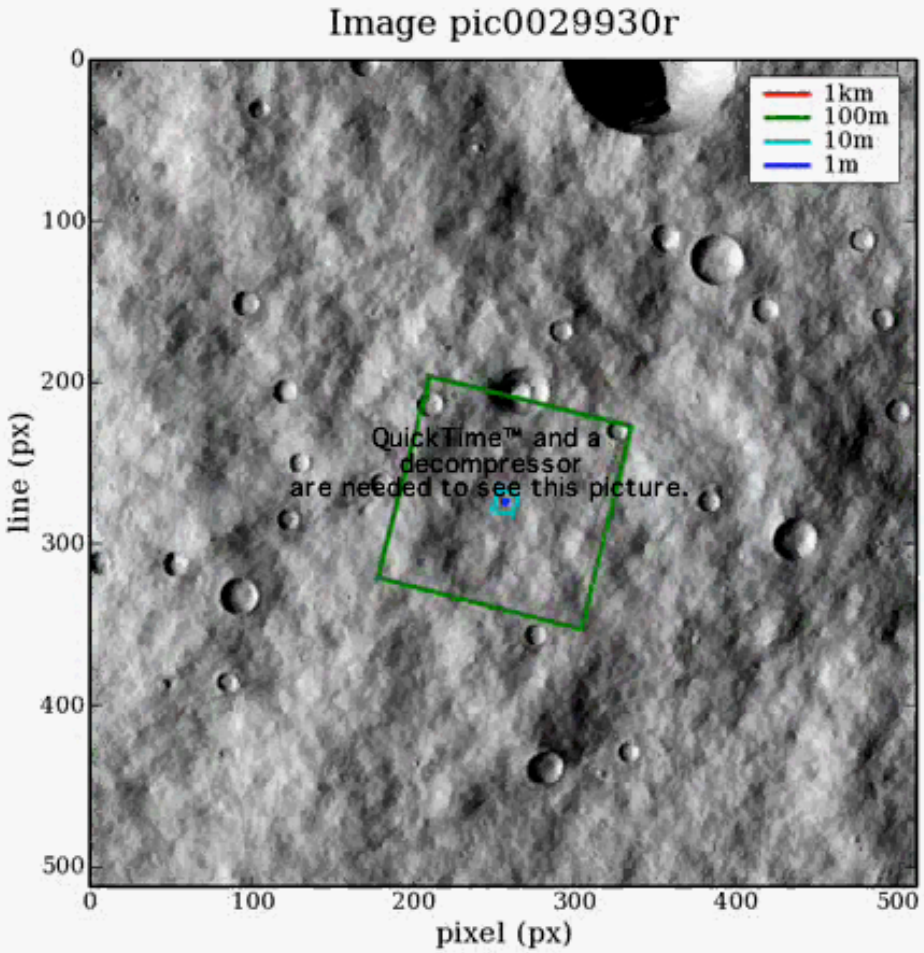
- Extension of EDL architecture
- Make complex system behavior transparent
- Manage GNC simply and directly in a high-level manner
- Protect spacecraft: "live to fight another day" vs. "land or die"
- Ground has many opportunities to intervene
- Spacecraft can decide to shortcut the loop and fly by or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



Enhanced EDL-like technique

Application: approach and contact with body

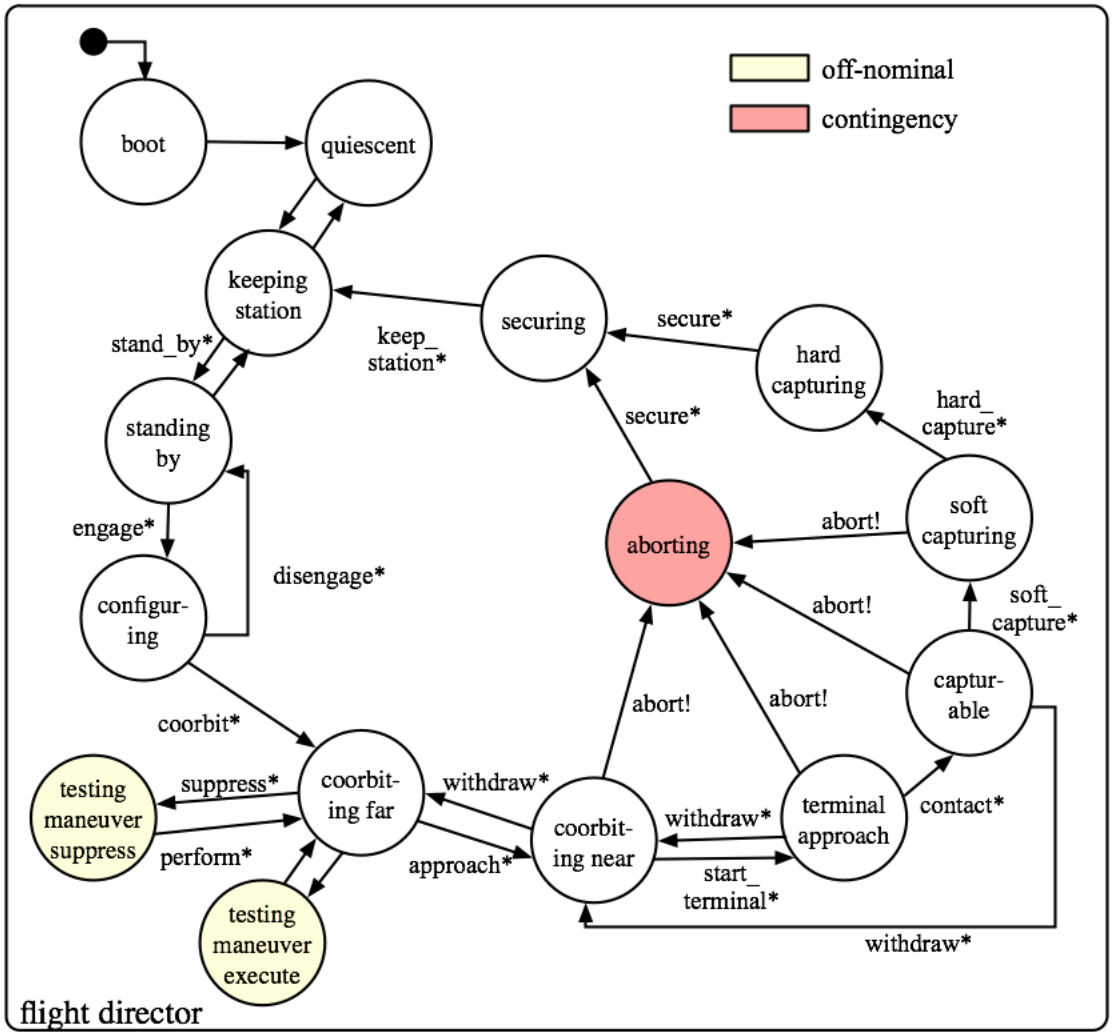
- Use executive to direct lower level spacecraft functions for approaching, sampling, ascending
- Comet, asteroid, lunar
- Tempel-1



One of several possible applications

Next step: Mars Sample Return Rendezvous

- Autonomous capture of 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

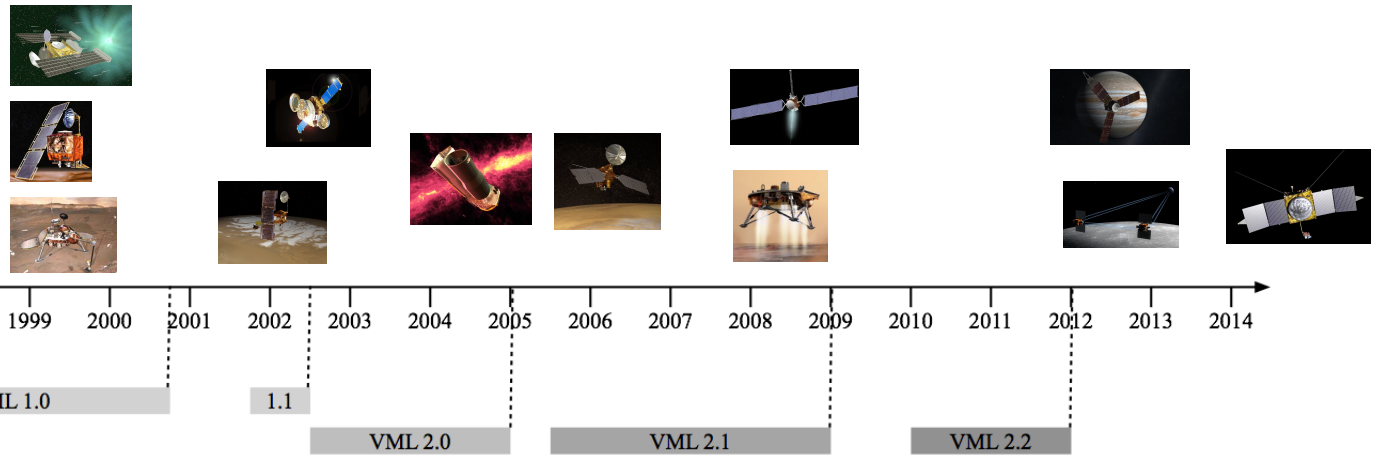
VML solutions for small body missions

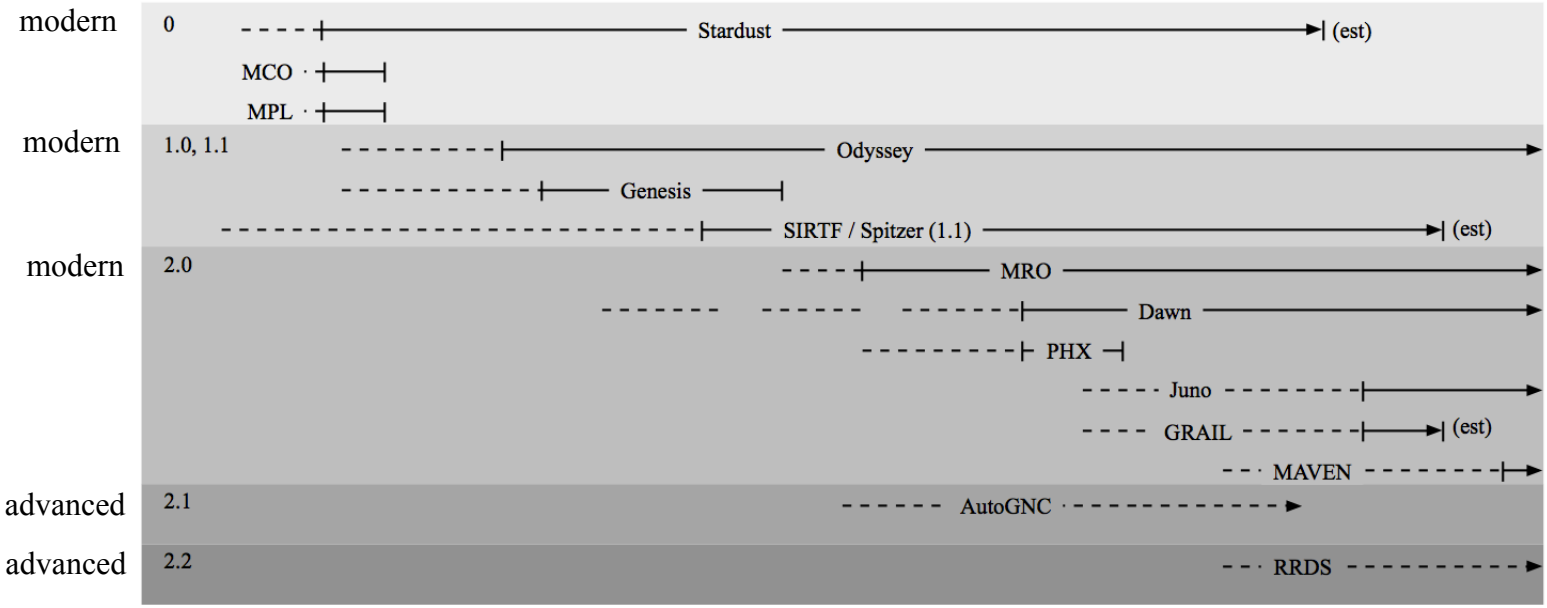
- Virtual Machine Language
- Parallel and serial execution of activities
- Absolute time-tagged master sequences (engineering and science)
- Reusable blocks for automated engineering and science activities
  - communications, ranging / doppler, maneuvering
  - wheel desaturation, power management, fault recovery configuration
  - science instrument operation
- Libraries for holding engineering and science blocks onboard, uplink savings
- Objects for instrument operations
  - coupled data and instructions
  - allows teams to work with greater independence
- State machines with coordination for complex activities when needed
  - rendezvous and docking
  - sampling
- Faster-than-realtime workstation-hosted test harness (OLVM)

Wide range of capabilities allow advanced operations

VML deep-space flight heritage

12 flight missions
44 flight years
6 versions





Evolution, not revolution

BLUE SUN

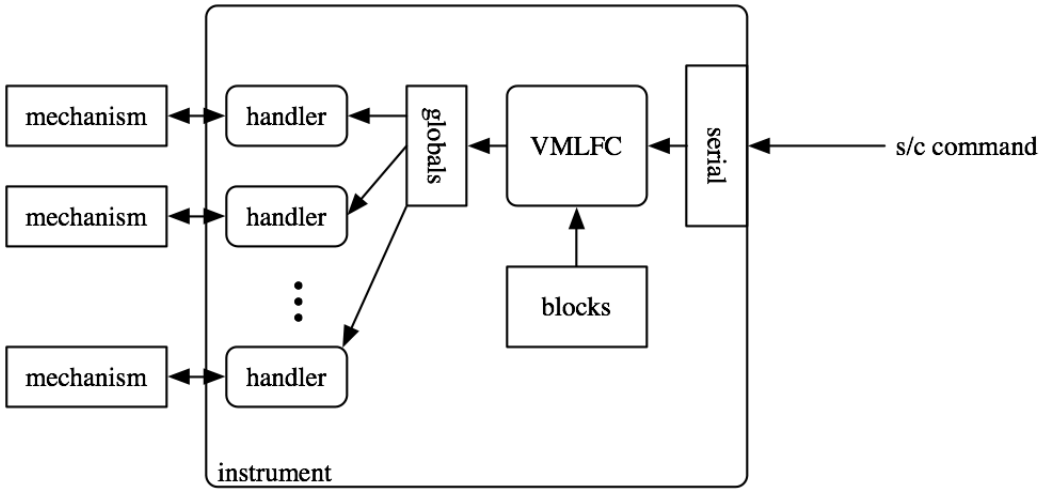
VML advantages

- Same automation mechanism across missions reduces development risk
- VML automation has advantages over flight software
  - cheaper to develop
  - more visible
  - easy to update
- Coordination between components is simple and concise in VML
- Components provide excellent operational insight using existing VML telemetry
- Components allow easy addition or deletion of functionality
- Logic testing fast: run four days of Mars Odyssey sequencing in 30 seconds
- State diagrams greatly clarify complex operations
- Executive implementation for AutoGNC easy to develop and modify
- State-driven approach directly applicable to new missions
  - touch-and-go asteroid/comet sampling
  - lunar landing
  - Mars Sample Return

VML provides mission-enabling automation capabilities

Micro-sequencing within instruments

- Treat instruments as very tiny spacecraft: serial i/o connection
- Sequencing provides reusable control elements to implement spacecraft command responses
  - simple patch: replace block within instrument, no software change
- Advantages:
  - software commonality across divergent instruments
  - reduced learning curve for developers and operators
  - pre-built solutions available early in instrument development
  - inboard VML available if spacecraft does not feature adequate sequencing capability



Reduce development time, provide common capability

BLUE SUN

Conclusions

- Sequencing capabilities have a large impact on operability of missions
- Small body missions for sampling and rendezvous require limited periods of autonomous activities which advanced sequencing can provide
- Sequence elements allow common functionality to be modularized and reused
  - reduced uplink rates and DSN contact times
  - reduced review time
  - increased insight into operations
  - automation and limited autonomy
- Advanced sequencing allows "safe sandbox" components to be developed without the expense, risk, and complexity of flight software
  - components easily changed, tested, and updated
  - much lower risk than flight software: eliminate most potential errors
  - much lower expense than flight software: operations personnel
  - requires lower level of review and scrutiny than flight software
- VML-style capabilities are an excellent fit to the needs of small body missions

Advanced sequencing technologies enable small body missions

Backup slides

Backup slides follow

More detail for further discussion

Overview of VML

- Virtual Machine Language
- Provides mission automation and autonomy via commands and logic
- VML flight code is mission-independent: reusable between missions
- VML accommodates any format for spacecraft commands
- Multiple threads of execution running under one RTOS task: repeatable testing
- Multiple data types and constructs (integers, strings, doubles, collections, etc.)
- Virtual machines execute steps onboard spacecraft
  - events react to conditions
  - timed execution
- Named functions accept input parameters: reusable logic
- Workstation execution tool for development: measured 100,000x real-time
- Synchronizable state machines constrain operations, lower risk, increase capability
- Free to government / university users

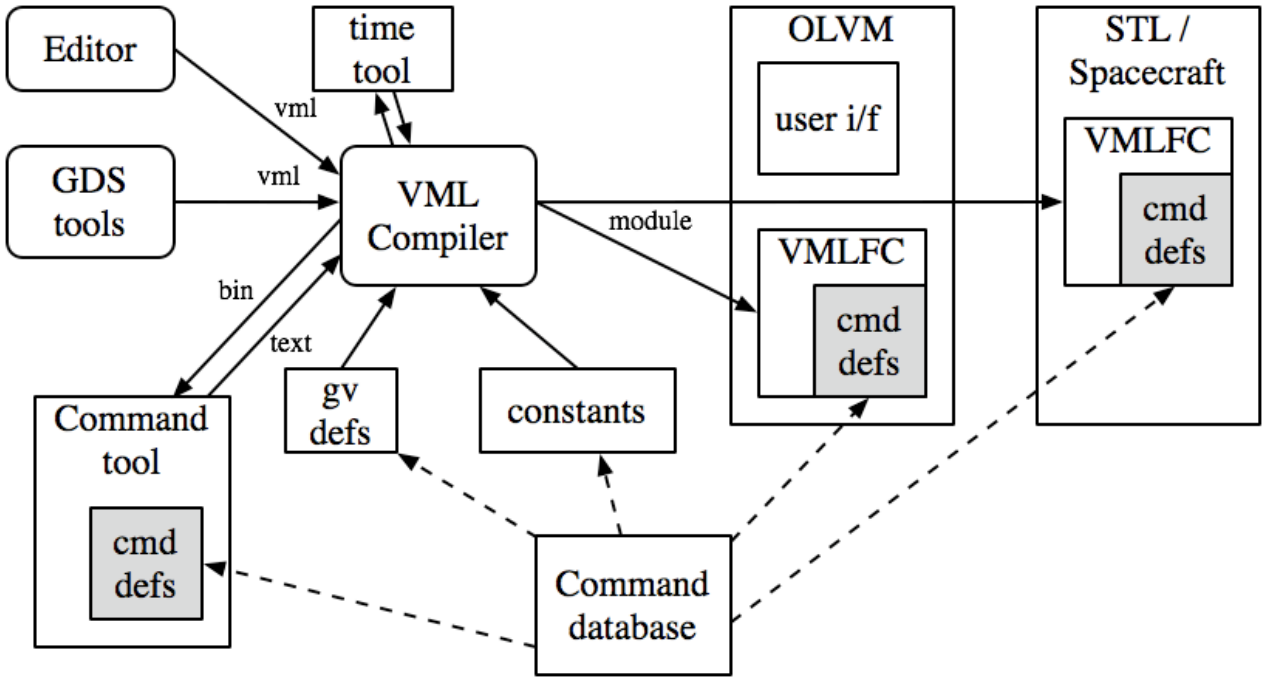
Language tuned to operations

Evolution of VML functional use

	ver	group	#	linear	aero	instr	prop	FP cfg	FP det	FP res	con-straint	nav	state	data
Stardust	0.0	LM/JPL	8	X				X						
MCO	0.0	LM/JPL	8	X				X						
MPL	0.0	LM/JPL	8	X				X						
Odyssey	1.0	LM/JPL	8		X	X	X	X						
Genesis	1.0	LM/JPL	8			X	X	X						
Spitzer	1.1	LM/JPL	12			X	X	X						
MRO	2.0	LM/JPL	20		X	X	X	X						
Dawn	2.0	OSC/JPL	12			X	X							
Phoenix	2.0	LM/JPL	16		X	X	X	X						
Juno	2.0	LM/JPL	32			X	X	X						
GRAIL	2.0	LM/JPL	16			X	X	X						
MAVEN	2.0	LM/LASP	32		X	X	X	X	X	X	X			
TAG (tech)	2.1	BSE/JPL	20			X	X	X	X	X	X	X	X	
MSR (tech)	2.2	BSE/NASA	32		X	X	X	X	X	X	X	X	X	X

VML usage increases with mission experience

VML tool suite



Human readable VML written with text editor or autogenerated → Compiler accepts VML file, makes binary output: mission independent tool → Workstation hosts Offline VM, tests logic faster than real-time → Software Test Lab or spacecraft, 100% flight code fidelity with OLVM

Rapid development and testing of sequence products

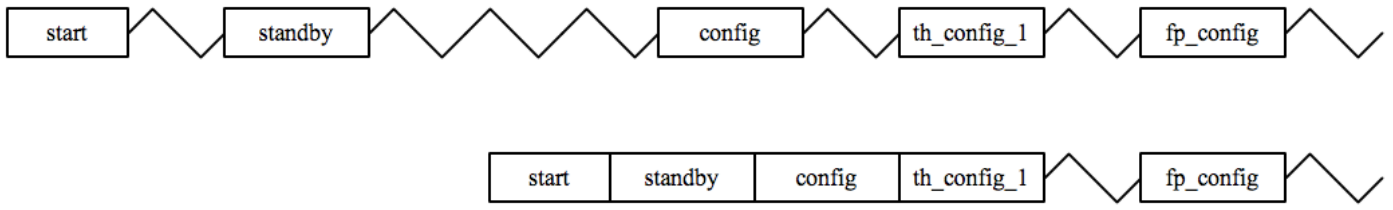
Sample VML block

```
block point_and_shoot
    input ra
    input dec
    input exposures
    input exposure_time
    declare double pointing_delta := -1.0
    declare int i := 0
body
    issue_dynamic "slew_to", ra, dec
    pointing_delta := wait gv_pointing_error < 0.0005 timeout R00:00:25.0

    if pointing_delta < 0  &&  gv_pointing_fault_respond then
        call recover_from_pointing_error ra, dec
        return false
    end_if

    for i := 1 to exposures do
        issue expose_ccd
        delay_by exposure_time
    end_for
    return true
end_body
```

Execute logic in a *safe sandbox*

Sequencing requirements and constraints

- 37 critical requirements to satisfy from the Phoenix EDL baseline reference mission
- Load and start four days out
- Shift activities with updates to estimated entry time
  - influence of gravity as spacecraft approaches
  - altitude of atmosphere at atmospheric interface
  - last update a few hours out
- Catch up if spacecraft resets before landing: *slinky effect*



Many design implications

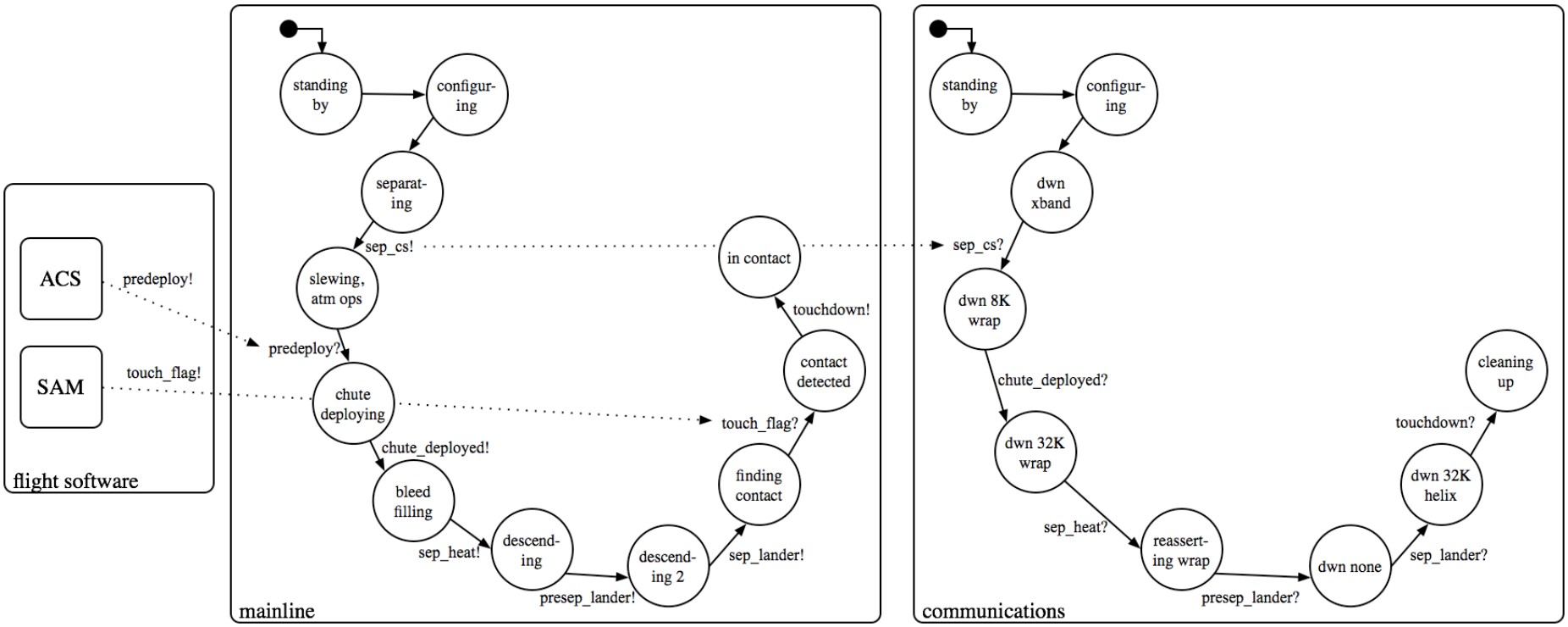
Communications and data requirements

- 13 requirements of the 37 requirements dealt with communications and data
- Shift activities with updates to estimated entry time
- Operate MARDI descent imager (later removed)
- Perform uplink verification
- Collect CPU utilization information
- Record times of critical events
- Remove telemetry backlog after switching antennas
- Send swan song if reboot after cruise stage sep
- Manage communications hardware in response to changing spacecraft configuration
  - X-band 700 bps until cruise stage separation
  - UHF antennas change: wraparound after sep, then helix after dropping out of backshell
  - UHF modes: none / carrier only / 8 Kbps / 32 Kbps
  - retransmit critical data collected during plasma blackout
  - wide variety of prechannelized streams with different information at varying update rates

Manage with separate components: *modularize*

Mainline directs all follower state machines

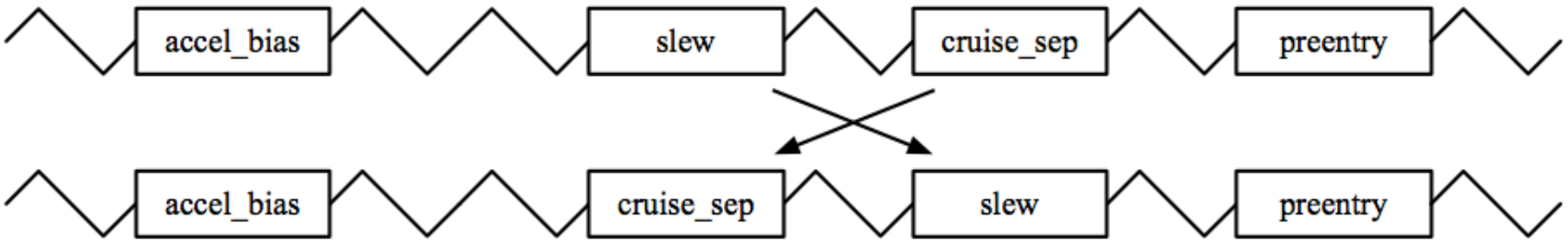
- Mainline receives (?) events sent by flight software by waiting on global variables
- Mainline transmits (!) signals by setting global variables
- Followers receive (?) signals by waiting on global variables



Coordination technique simplifies design, allows use of components

EDL flight experience with VML

- State-based event-driven sequencing highly successful for EDL
- EDL mainline worked the **first** time, and landed the spacecraft **every** time
  - Last change (aside from comments) made March 2007
  - Last version built May 2007, incorporated into final launch EEPROM file system
  - Other elements failed during testing, but EDL mainline landed anyway
  - EDL mainline unchanged before launch, ran EEPROM version for landing May 2008
- Easily able to remove MARDI imaging functionality without changing mainline
- Easily able to switch "slew before separate" to "separate before slew"



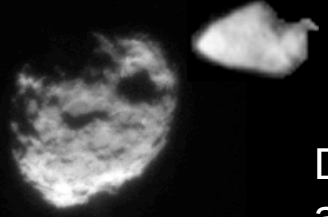
Components allowed considerable product stability and flexibility

AutoNav heritage

DS1 AutoNav
Deep Cruise, Navigation
Sept.1999

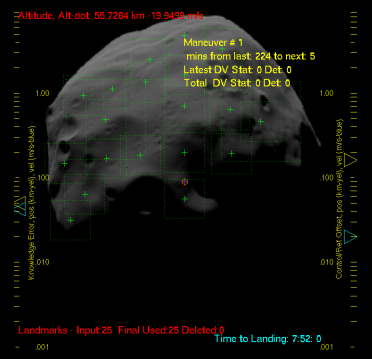
DS1 AutoNav
at Borrelly Sept. 2001

Stardust AutoNav
at Annefrank and Wild 2,
Nov. 2002, Jan. 2004



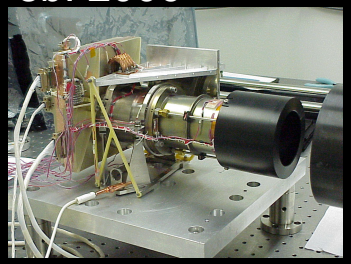
Deep Impact AutoNav
at Tempel 1 July 2005

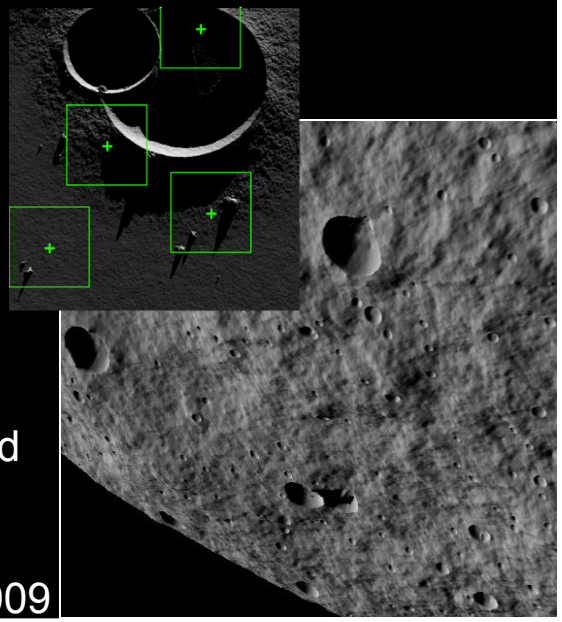
Hayabusa Imaging
Science: Itokawa
Shape Model, Sept.
2005



DI AutoNav Phobos
Landing Simulation
Dec. 2005

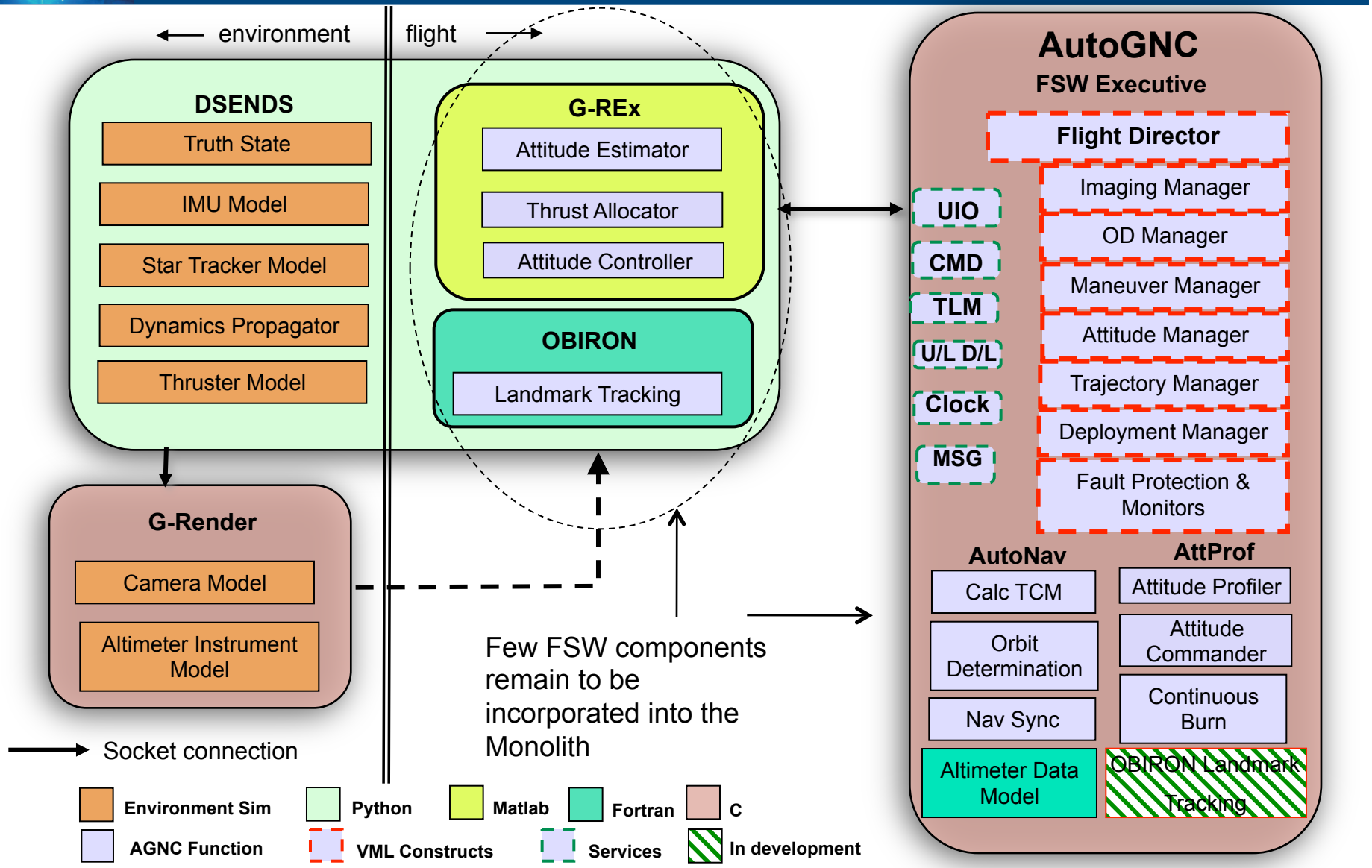
MRO OpNav
Camera Validation
Feb. 2006





Altair lunar landing and
“Touch and Go” on
Comet AutoGNC
simulations, Winter 2009

AutoNav + VML = AutoGNC



State machines guide AutoGNC services

- Attitude profiling: turn to desired attitudes
- Attitude commanding and thrust allocation: implement tick-by-tick attitude/torque and translation/force requests to thrust allocator and RCS implementation
- Landmark tracking/image processing: model surface terrain, identify and locate surface landmarks
- Altimetry modeling: model surface terrain, predict LIDAR-based ranging
- Orbit determination: data fusion of imagery, altimetry, accelerometry, data filtering, spacecraft position / velocity estimation and propagation.
- Maneuver design: computes trajectory corrections using estimated spacecraft state
- Maneuver implementation: plan and implement burn execution, monitor burn progress

VML state machines orchestrate AutoGNC services

TAG different than EDL

- Repeatable rather than one-way
- Reversible: may abort and go around
- More paths to support than EDL's "land or die" linear progression
  - wave-off could result in flyby
  - interim checks to confirm correct processing
  - deactivation of high level fault protection
  - commitment decision based on LIDAR / AutoNav lockup, ground permission
  - descent / ascent burns
  - solar panel articulation: stow then redeploy
  - contact detection
  - sample management
  - nominal and emergency withdrawals
  - reactivation of high level fault protection

State machines even more useful for TAG

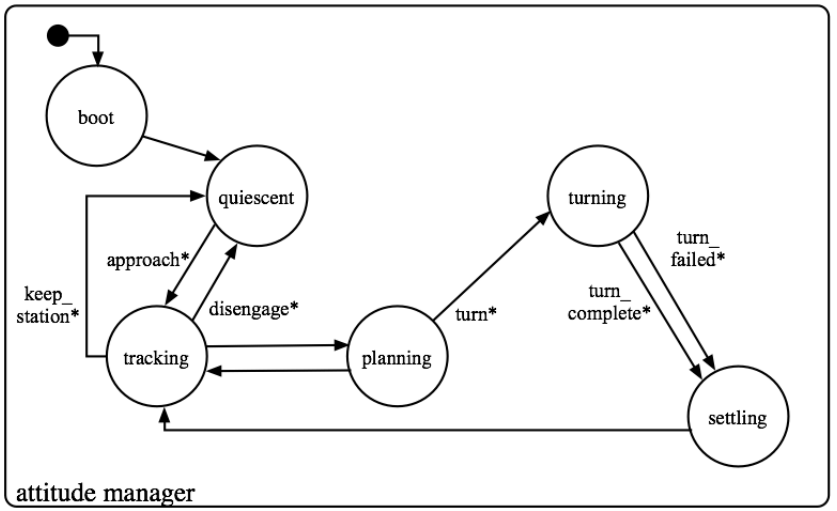
Extend coordination mechanism

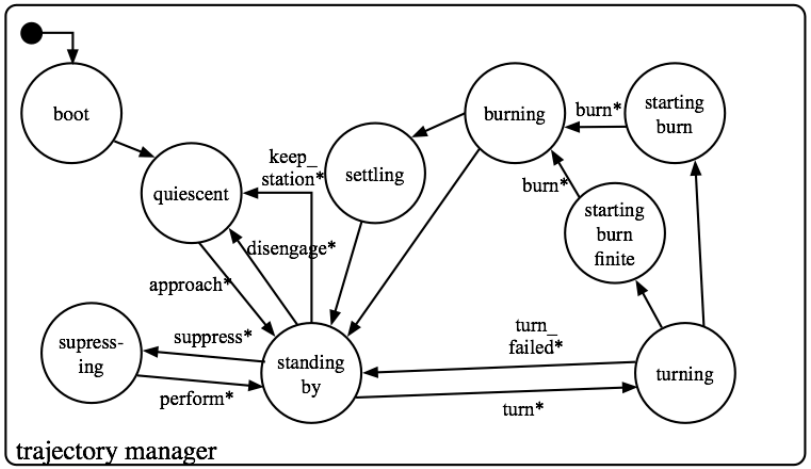
- EDL "land or die" philosophy fundamentally one-way series of activities, so one-way coordination with transmit / receive sufficient
- Multiway coordination needed for TAG: extend VML
- State machines executing in parallel synchronize to transition simultaneously
- Makes visible verifiable permissions to take state transitions
- Specialized transitions with same signal names cause simultaneous transition
  - transmit: one way notification of action to any receivers (!) [EDL]
  - receive: reception of transmitted signal with no acknowledgment (?) [EDL]
  - synchronize: two way check (*) [new for AutoGNC TAG mission]

Coordination causes individual state machines to run as a system

Synchronized state machines in VML 2.2

- Named states go to other states
  - direct jump
  - take named transition to coordinate with other state machines
- Concise representation of operations
- Allows problems to be modularized



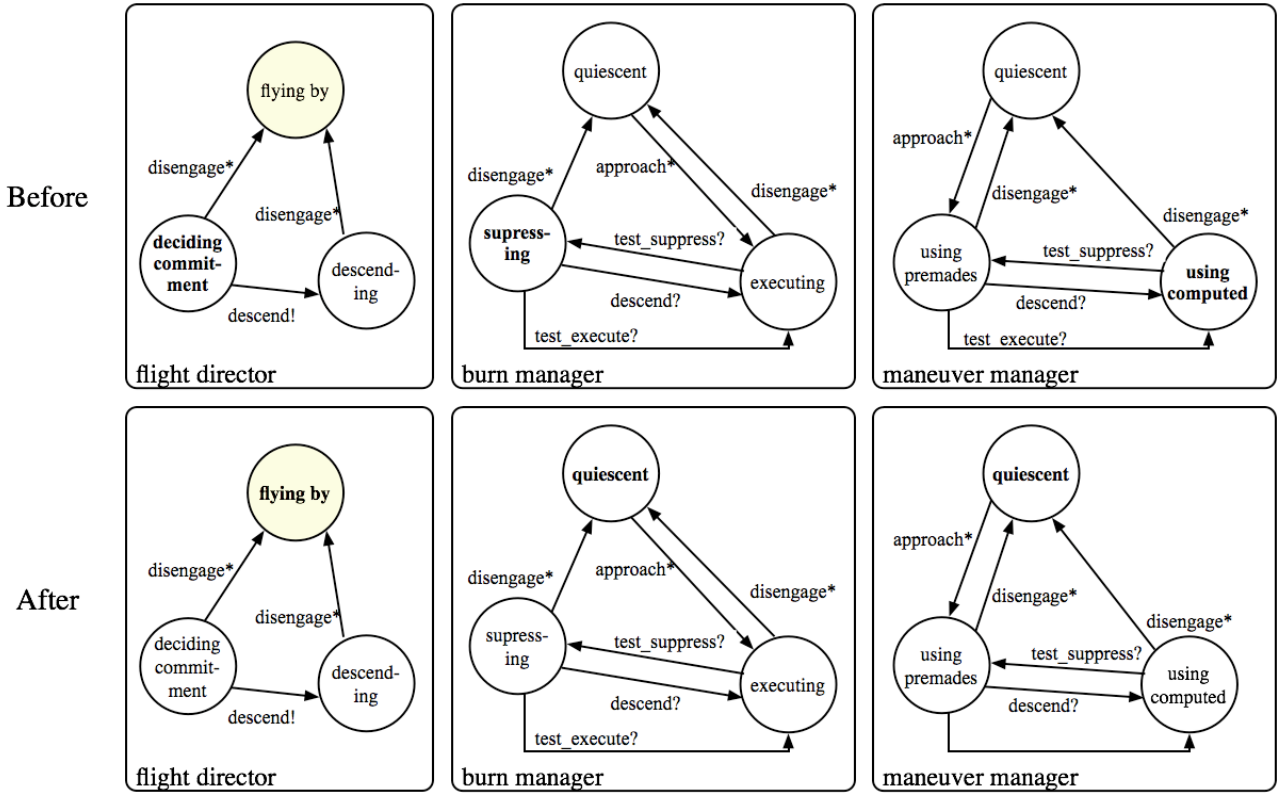


- Usually used as system design abstraction
- In VML 2.2, state machines are *directly executed*: no translation into a procedural language

Simple representation of complex operations

Synchronized coordination

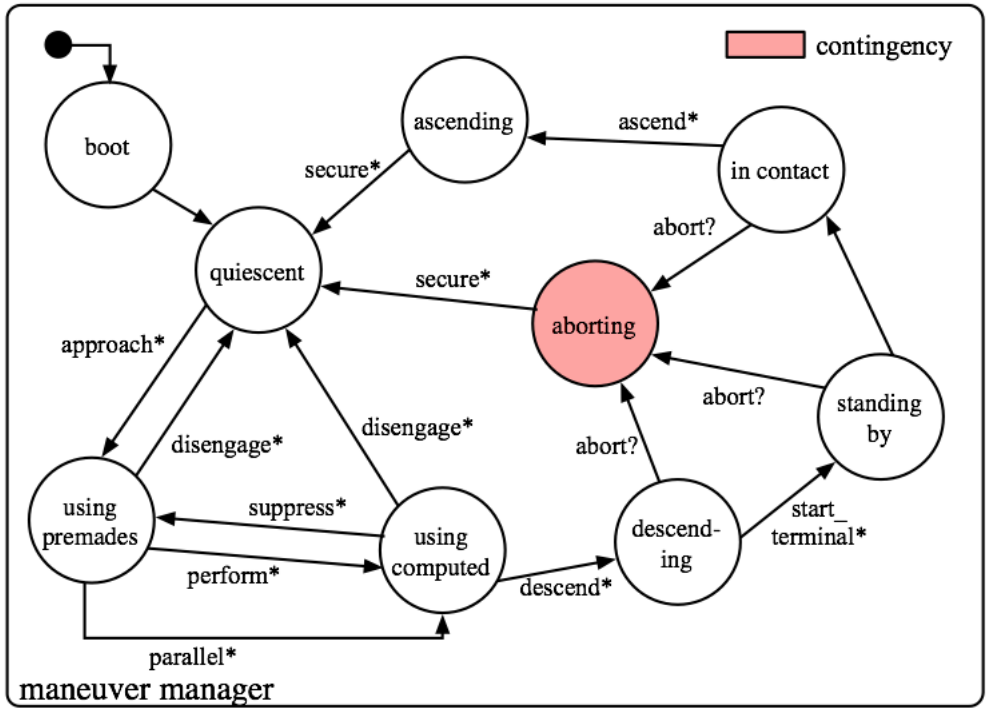
- Two way coordination
- All machines in state with synchronized transition due, then *take* `disengage*`

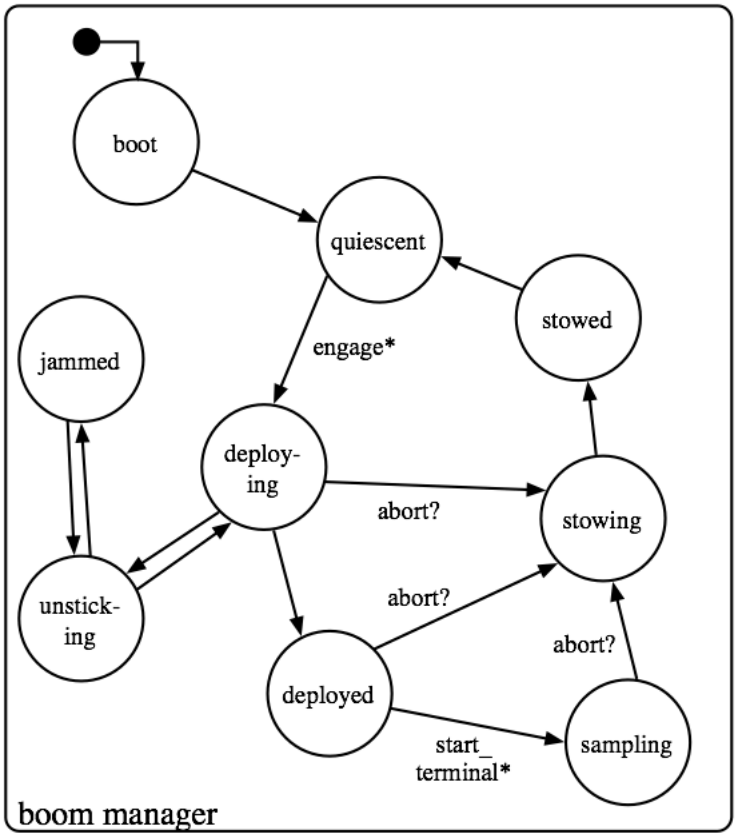


Checks all present before taking transition

Sample managers

- Manage lower level activities involving software (AutoGNC and others)
- Manage lower level activities involving physical change (panels, boom)





Follow synchronized transitions guided by flight director