Reactive Sequencing for Autonomous Navigation Evolving from Phoenix EDL

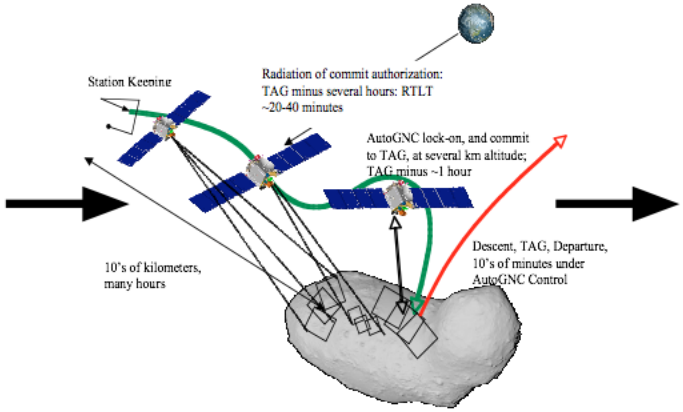
Dr. Christopher A. Grasso
JPL / Blue Sun Enterprises
christopher.a.grasso@earthlink.net

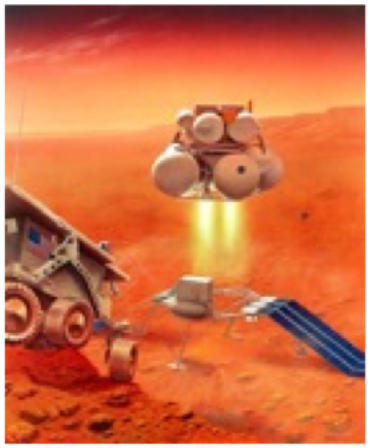
Joseph E. Riedel
Jet Propulsion Laboratory
joseph.e.riedel@jpl.nasa.gov

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

Introduction

- 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 for autonomous comet / asteroid sampling operations using AutoGNC for navigation using 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

Sequencing and commanding

- *Sequencing* is the issuing of commands from an on-board store causing the spacecraft to behave in a desirable way
  - used to coordinate and order all EDL activities
- *Commands* are directives to the spacecraft with some meaning, e.g.

  `MISSION_PHASE "EDL"`
- *VML* stands for Virtual Machine Language
  - standardized multimission sequencing language maintained by JPL
- Command execution timing
  - *absolute sequence*: commands tied to absolute time, exact timing
  - *block*: commands tied to time offset from preceding command
- *Events* are conditions driven by the environment, time can't be predicted
  - VML detects events to suspend and resume block execution using a *WAIT* directive

Sequencing orders on-board commands using time and events

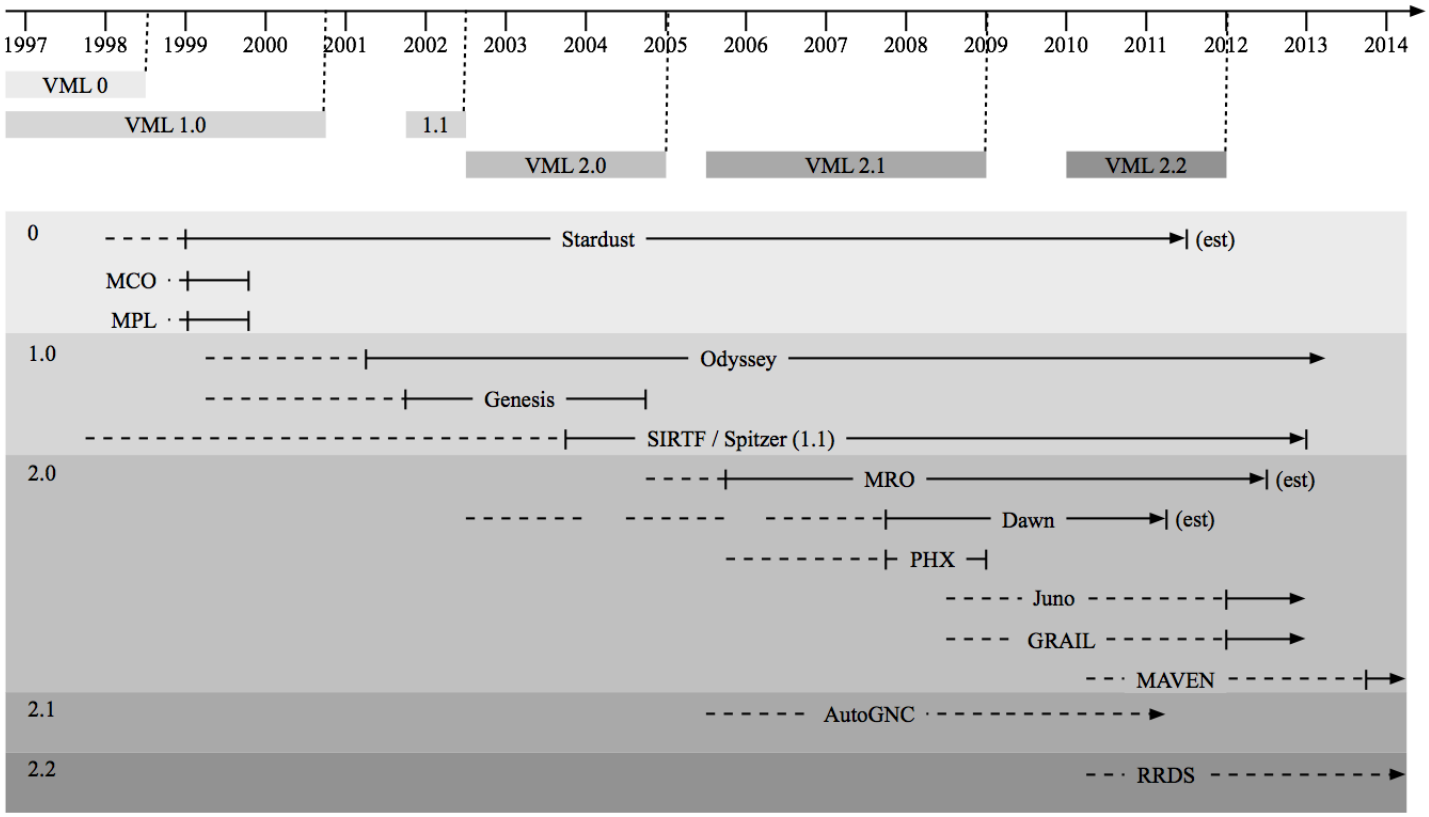
Overview of VML

- Virtual Machine Language
- Flight code is mission-independent: reusable between missions
- Multiple threads of execution running under one RTOS task
- Multiple variable data types and constructs (integers, strings, doubles, etc.)
- Virtual machines execute steps onboard spacecraft: can react to conditions
- Named functions can accept input parameters
- Execution tool for development and debugging runs much faster than realtime on workstations before deployment
- Synchronizable state machines
- State machines constrain operations, lower risk, increase capability

Language tuned to operations

Timeline

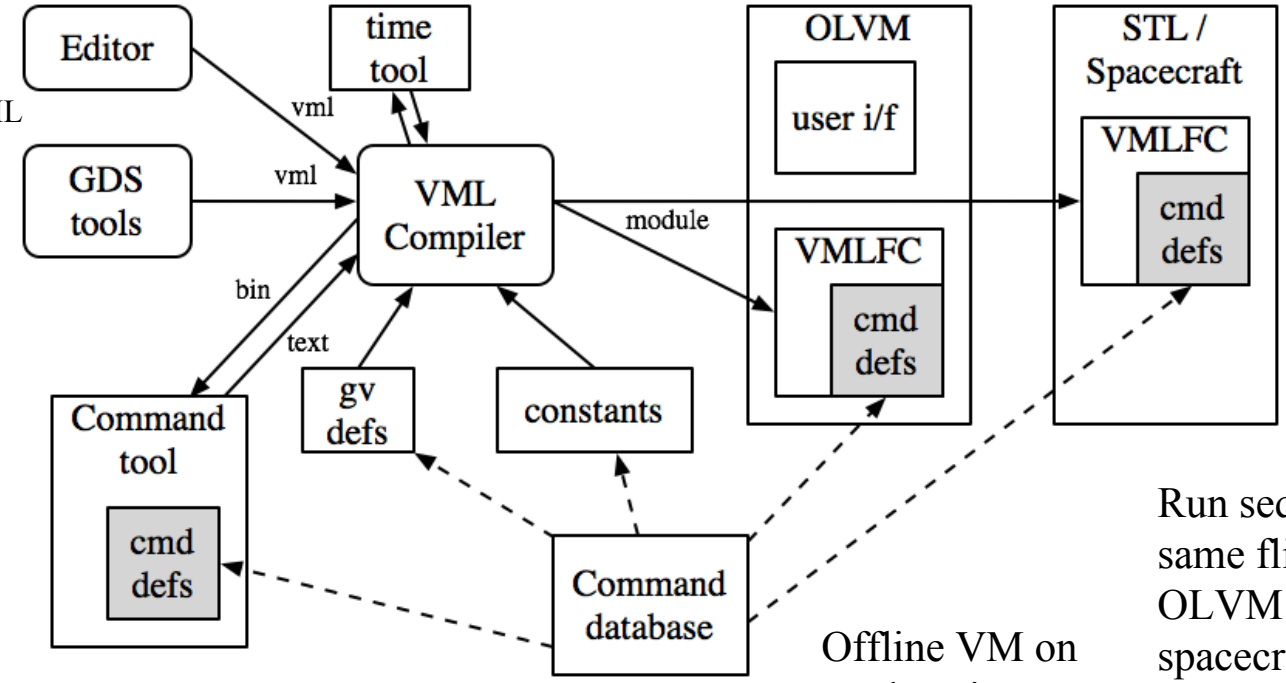
Twelve missions, twelve years, six versions



Long term VML evolution and use

VML tool suite

Human readable VML sequences written using GDS tools or directly with text editor



Compiler accepts VML file, makes binary output: mission independent tool

Offline VM on workstation used to test logic

Run sequence with same flight code in OLVM and spacecraft: 100% fidelity

Allows rapid development and testing of sequences

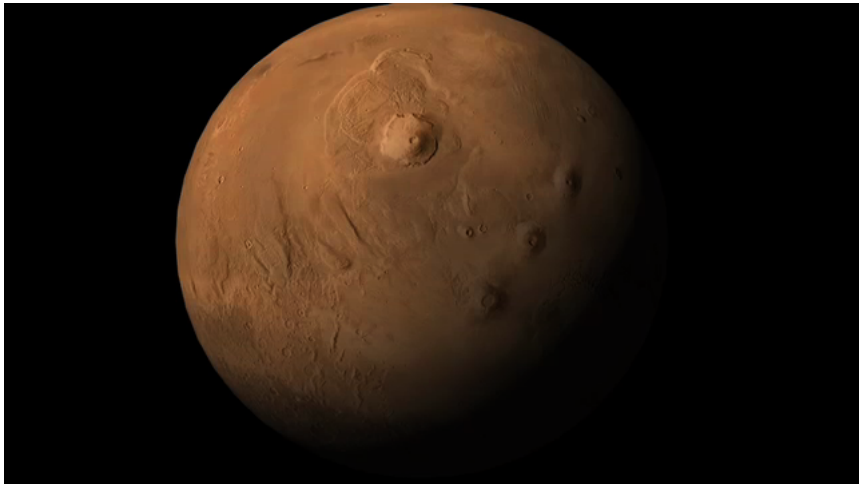
Sample VML block

```
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*

EDL: Entry, Descent and Landing

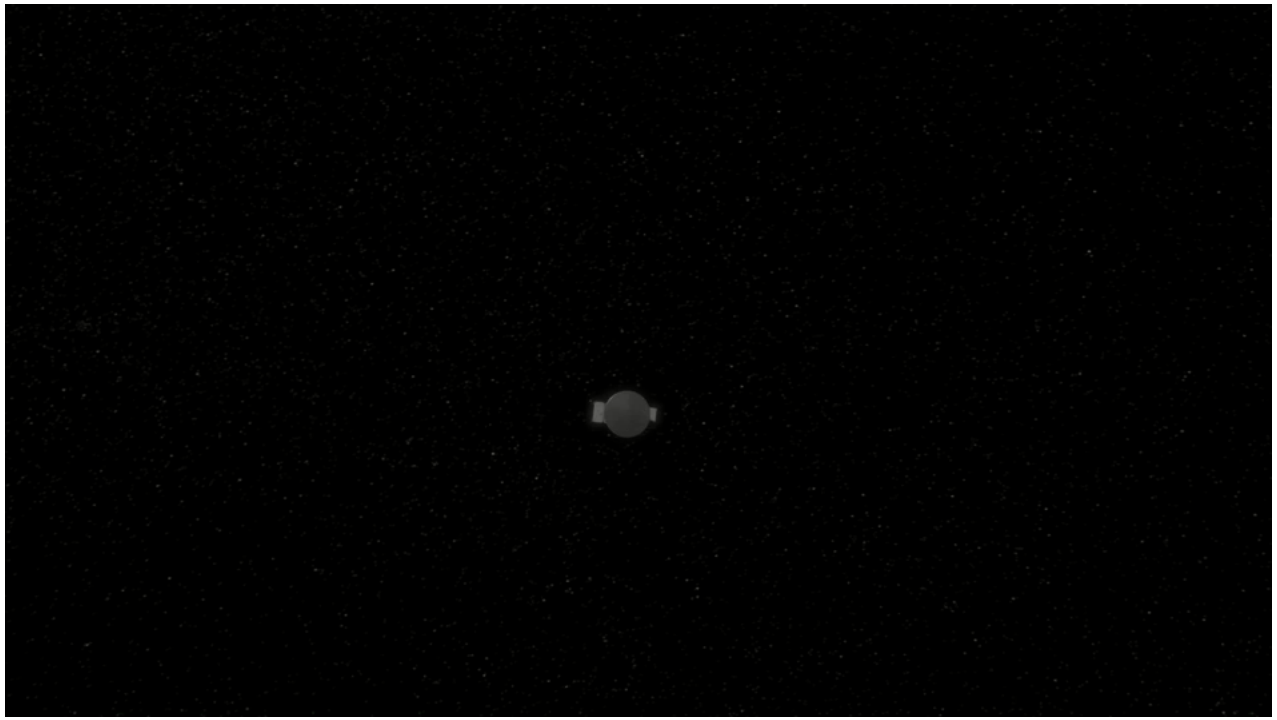
- 601 commands, 4 days, 17000 kph
- Hit ellipse 60 km x 20 km after trip of 470 million km
- Must work perfectly once to land Phoenix on Mars
- Throw away parts of the vehicle: cruise stage with solar arrays, X-band, star trackers
- No direct-to-earth communications after sep: use UHF relay via Odyssey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



"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

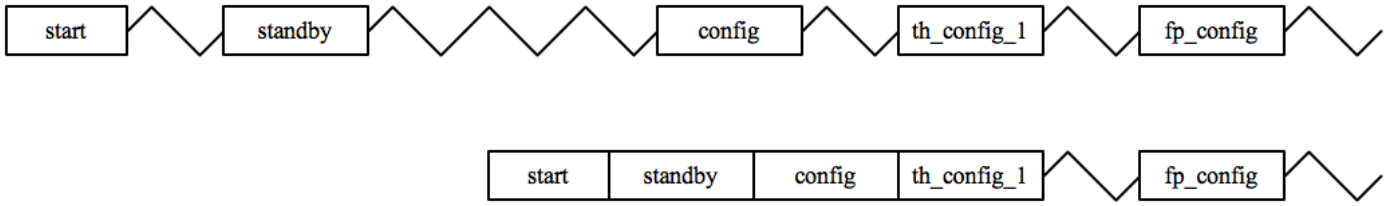


Phoenix EDL visualization

Many actions in short period of time

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

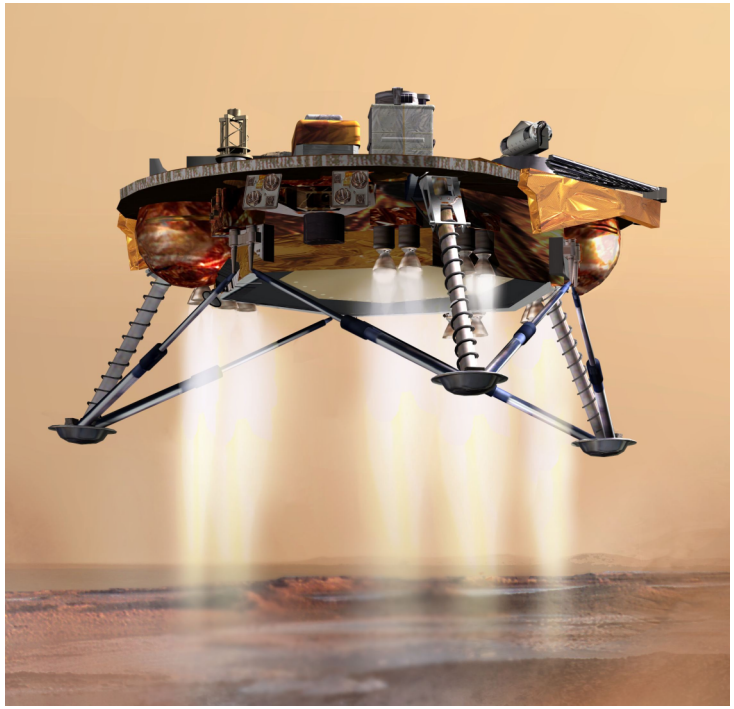
Communications and data requirements

- 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*

State-based approach

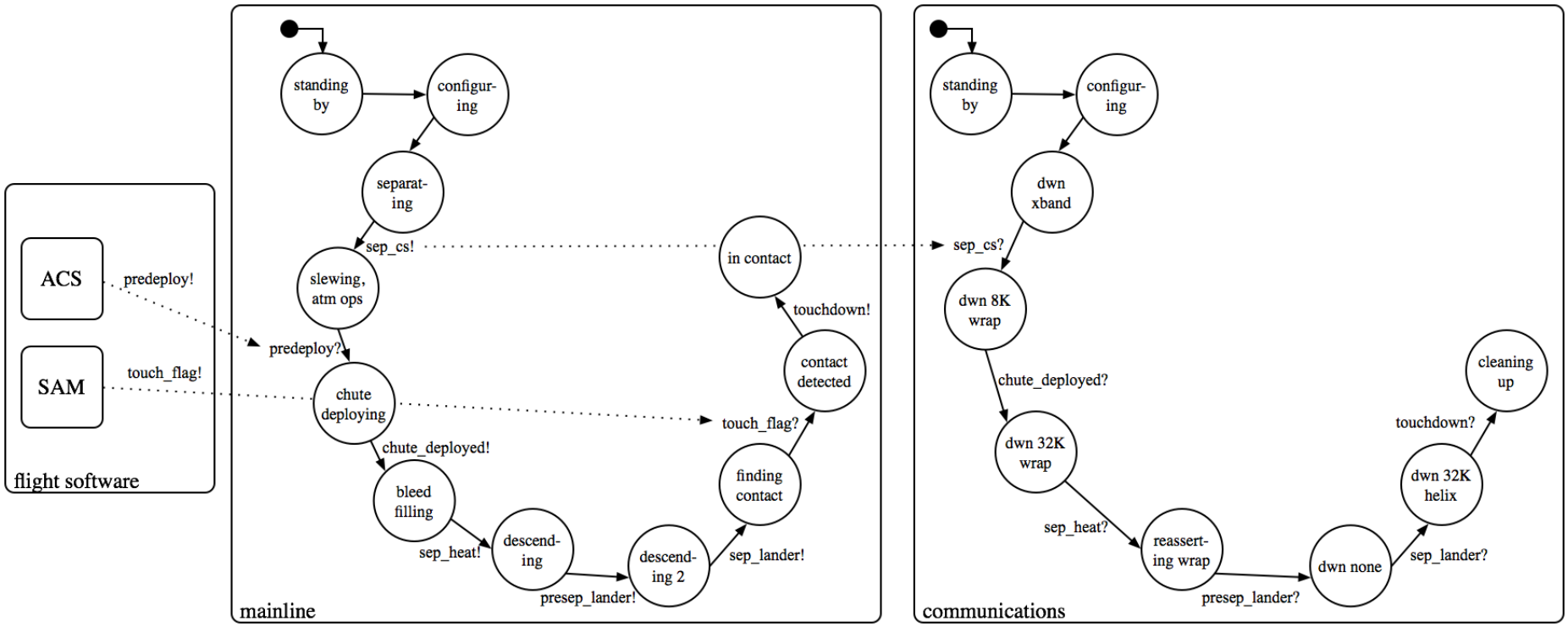
- Six parallel state machines on separate threads
  - mainline
  - sideline
  - communications
  - uplink verification
  - CPU utilization
  - imaging
- Mainline progresses through 27 substates, others follow using signals from mainline
- Centralized catch-up logic, counter represents progression through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

Mainline directs all follower state machines

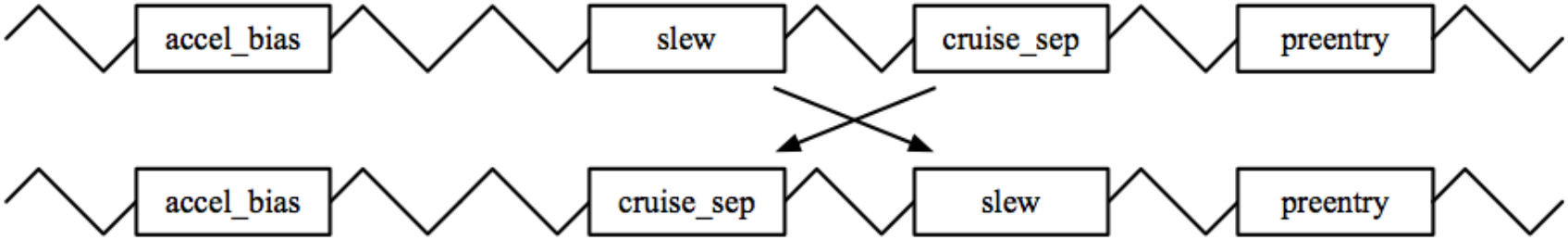
- Mainline receives (?) events sent by flight software by waiting on global variables
- Mainline transmits (!) signals by setting global variables
- Followers receive (?) signals by waiting on global variables



Coordination technique simplifies design, allows use of components

EDL flight experience

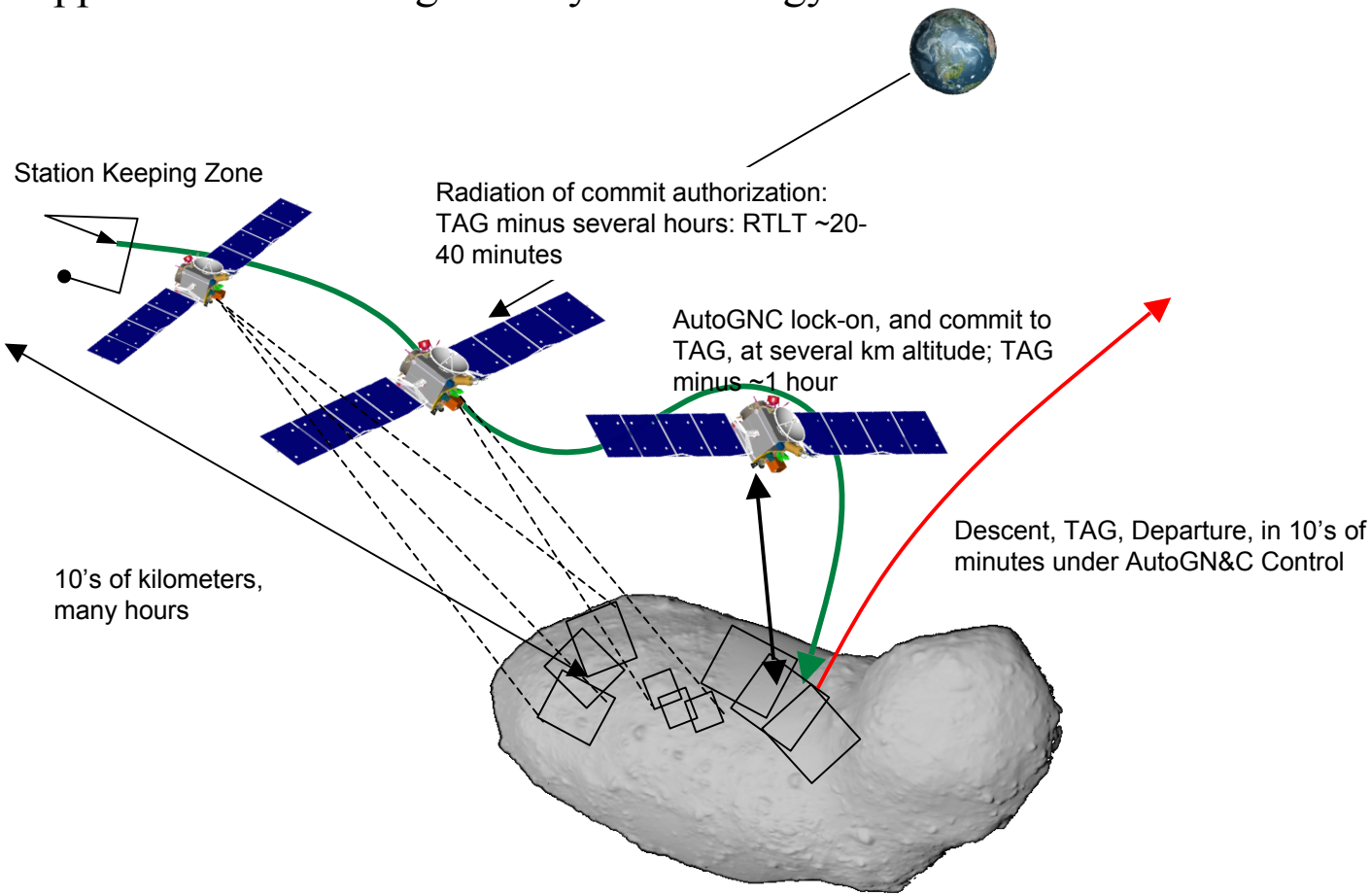
- State-based event-driven sequencing highly successful for EDL
- EDL mainline worked the **first** time, and landed the spacecraft **every** time
  - Last change (aside from comments) made March 2007
  - Last version built May 2007, incorporated into final launch EEPROM file system
  - Other elements 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

Touch and Go (TAG)

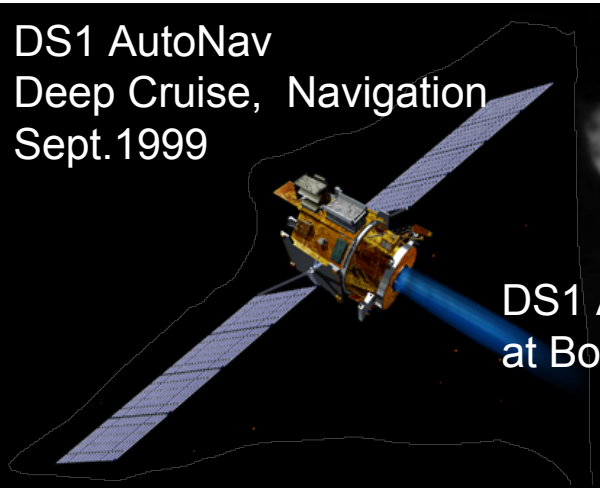
Comet sample approach and TAG: geometry and strategy



Progression is state-oriented

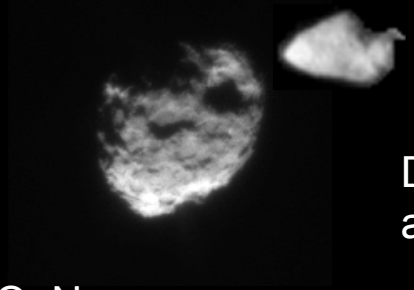
AutoNav heritage

DS1 AutoNav
Deep Cruise, Navigation
Sept.1999



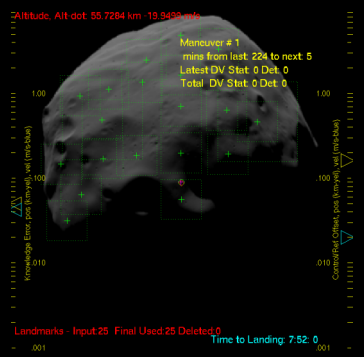
DS1 AutoNav
at Borrelly Sept. 2001

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



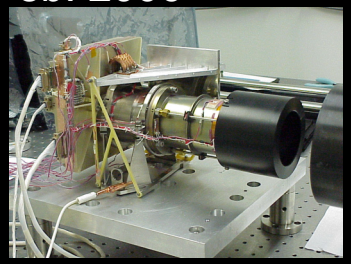
Deep Impact AutoNav
at Tempel 1 July 2005

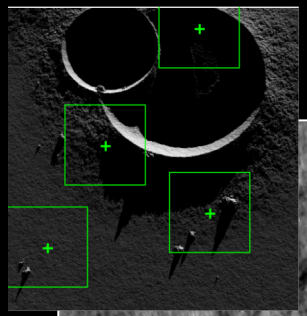
Hayabusa Imaging
Science: Itokawa
Shape Model, Sept.
2005

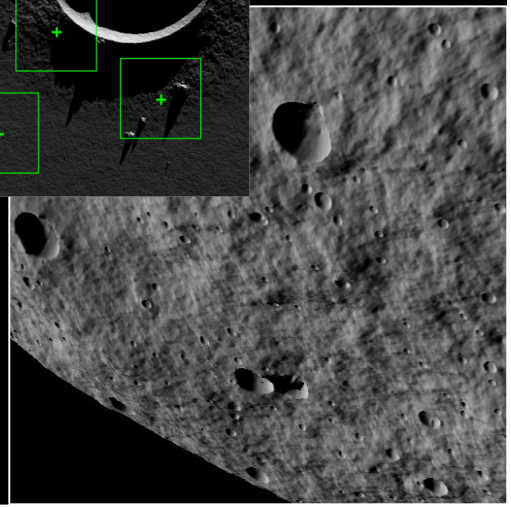


DI AutoNav Phobos
Landing Simulation
Dec. 2005

MRO OpNav
Camera Validation
Feb. 2006

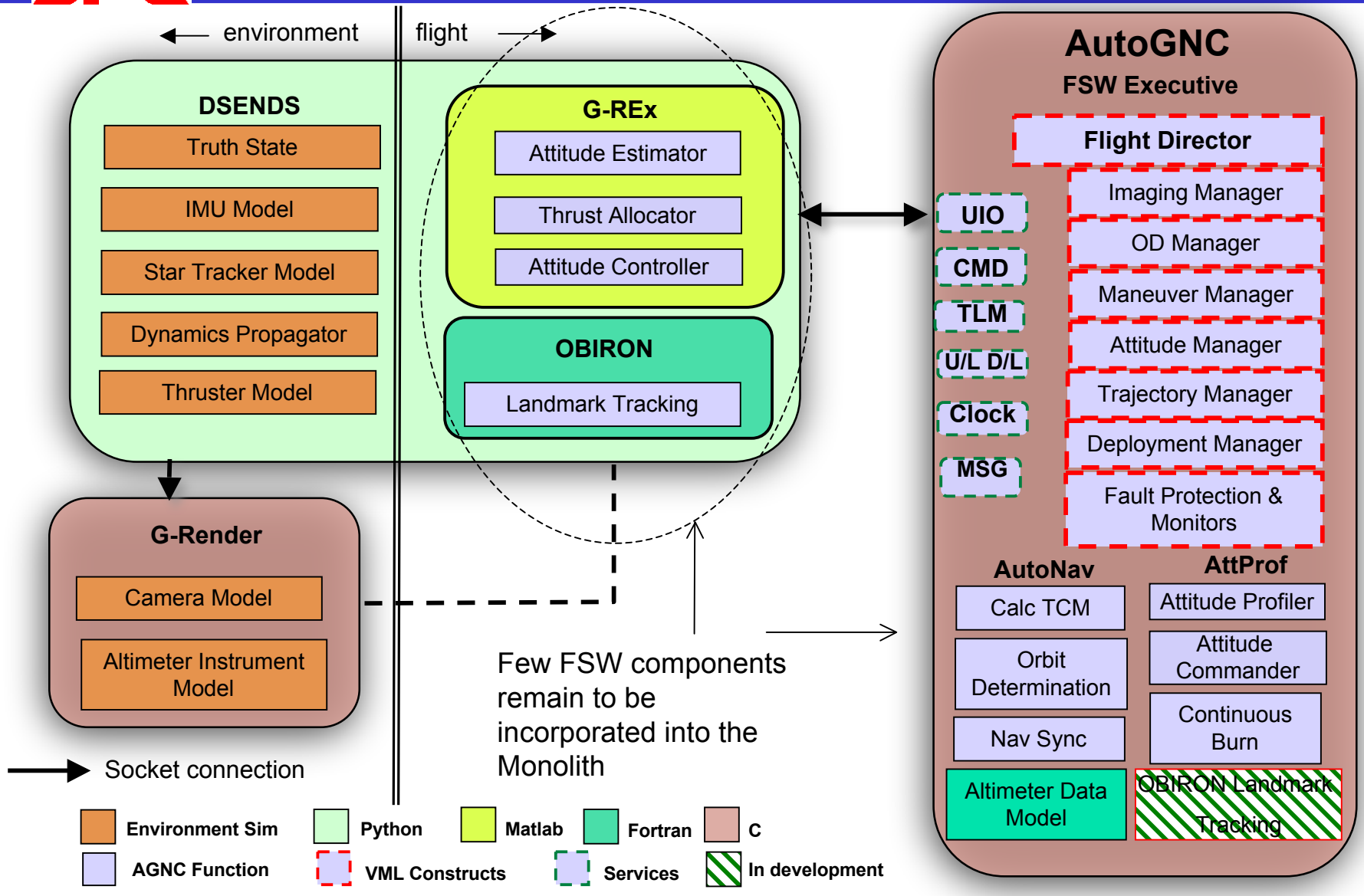






Altair lunar landing and
“Touch and Go” on
Comet AutoGNC
simulations, Winter 2009

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

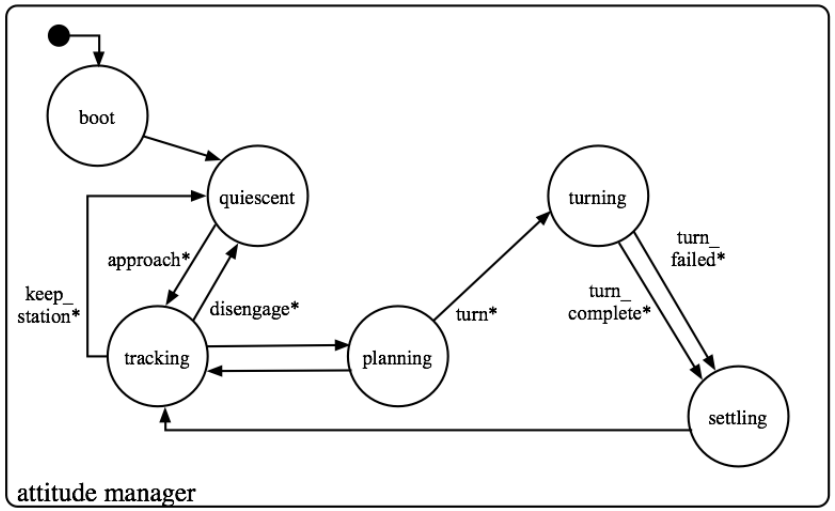
Extend coordination mechanism

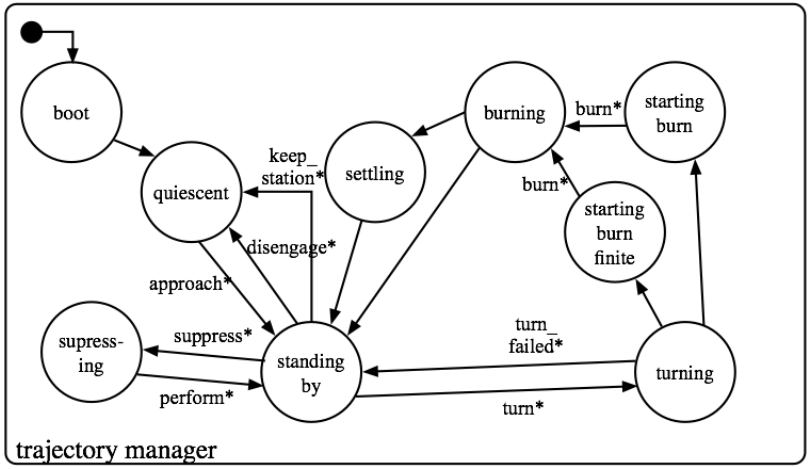
- EDL "land or die" philosophy fundamentally one-way series of activities, so one-way coordination with transmit / receive sufficient
- Multiway coordination needed for TAG: extend VML
- State machines executing in parallel synchronize to transition simultaneously
- Makes visible verifiable permissions 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

Full state machines in VML 2.1

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





- Usually used as system design abstraction
- In VML 2.1, state machines are directly executed: no translation of concepts into a procedural language

Simple representation of complex operations

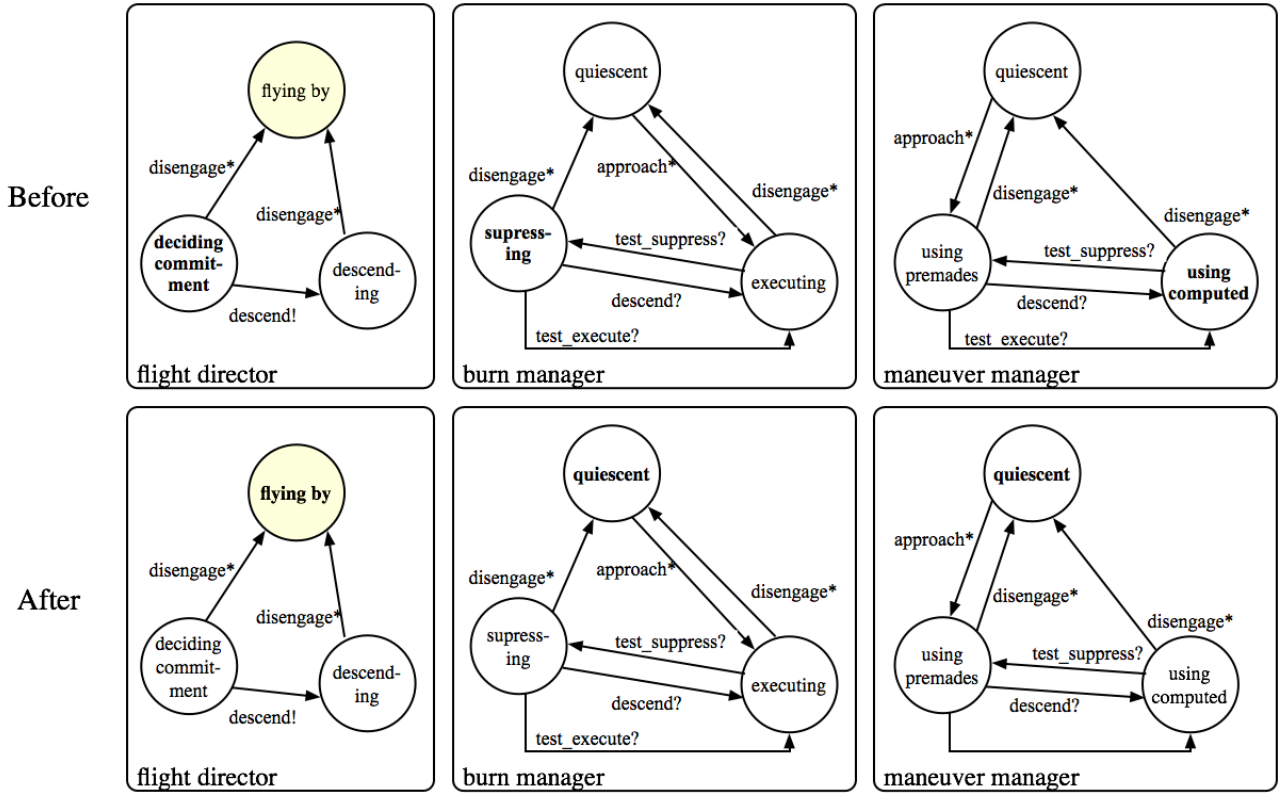
Synchronized coordination

- VML 2.1 and 2.2, not possible in VML 2.0
- Two stage process: check all machines are ready, then take a synchronous signal
- Checking registers constraint satisfaction
  - if a state machine is able to *take* a synchronous signal, register that fact
  - perform check incrementally
- Synchronized signal selection
  - if all state machines featuring the signal are able to take a transition with that signal, have all state machines simultaneously *take* their transitions
  - all state machines with the signal end up in new states simultaneously

Checking transition conditions before taking

Synchronized coordination

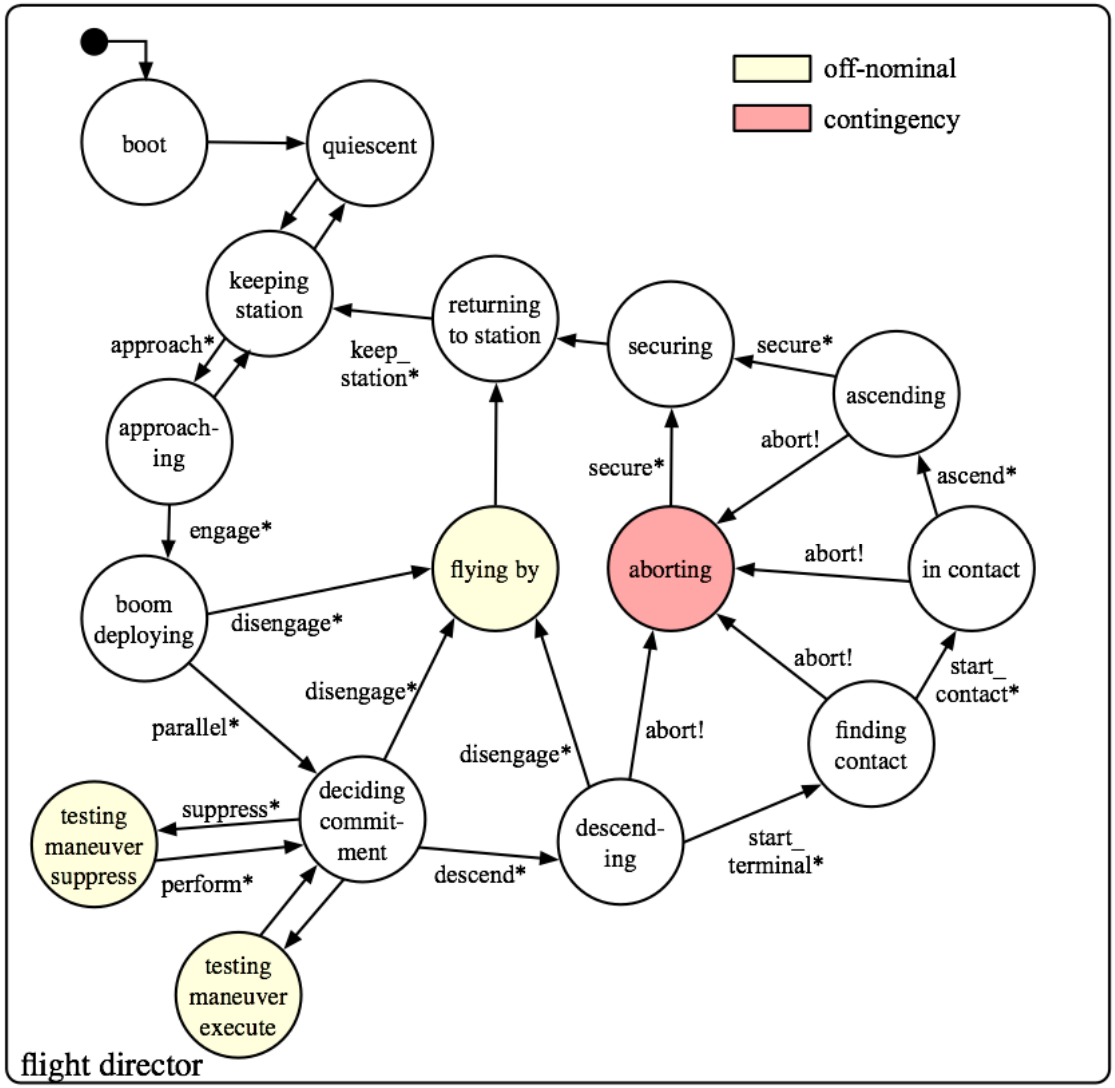
- Two way coordination
- All machines in state with synchronized transition due, then *take* `disengage*`



Checks all present before taking transition

Flight director: high level control of TAG

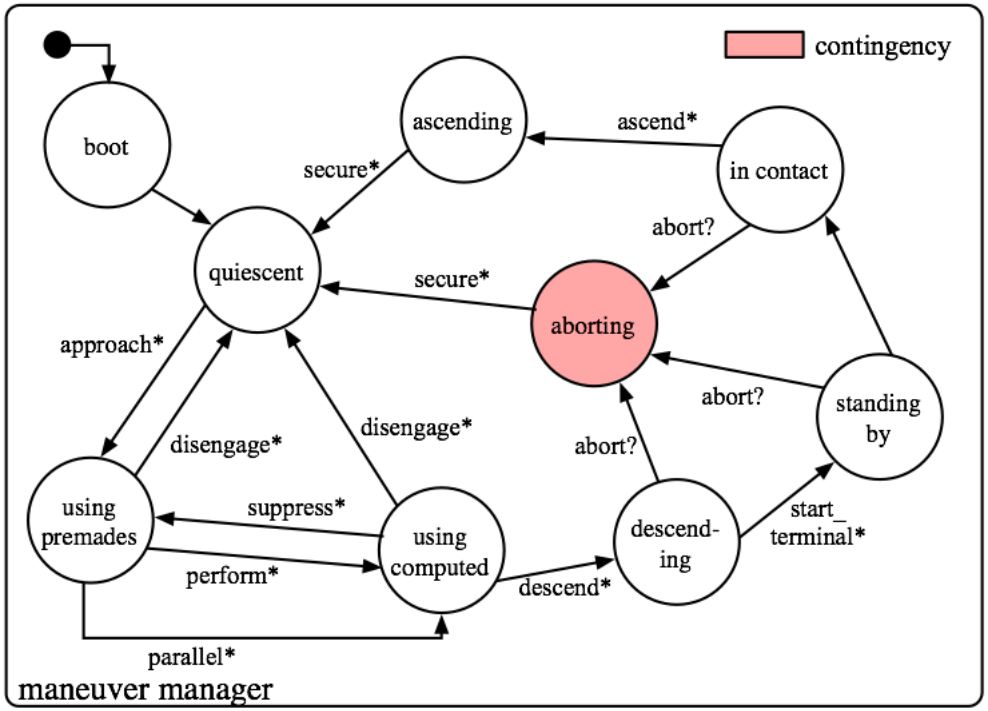
- Extension of EDL architecture
- Protect spacecraft: "live to fight another day" vs. "land or die"
- Ground has many opportunities to intervene
- Spacecraft can decide to shortcut the loop and fly by or abort
- Make complex system behavior transparent
- Manage GNC simply and directly in a high-level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

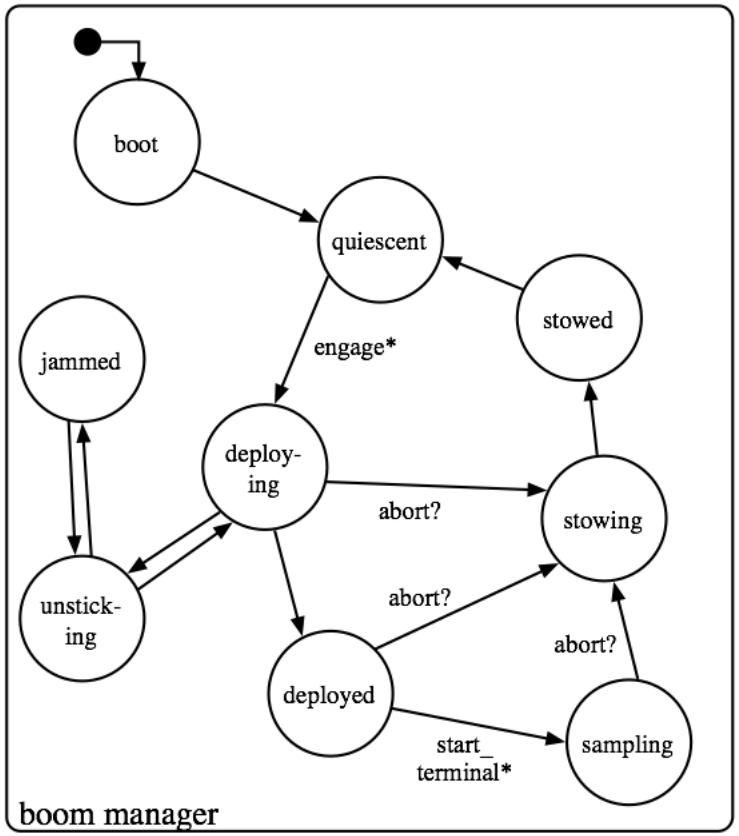


Enhanced EDL-like technique

Sample managers

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

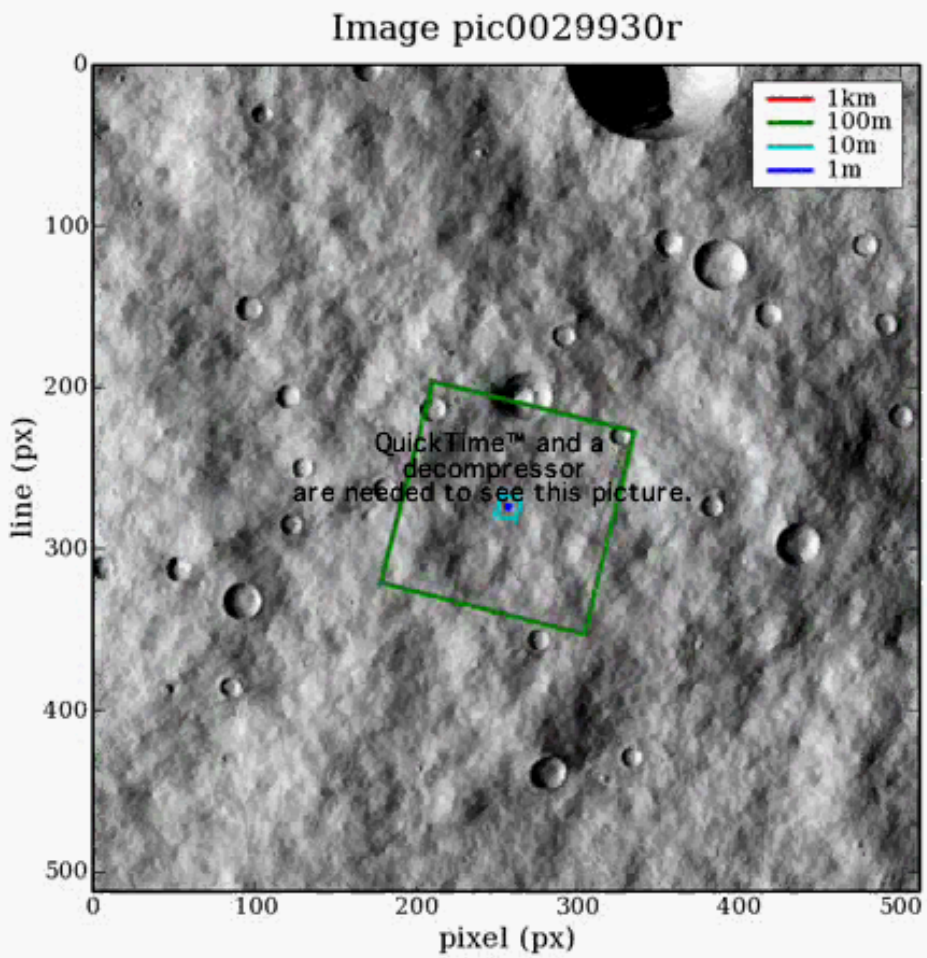




Follow synchronized transitions guided by flight director

Application: approach and contact with body

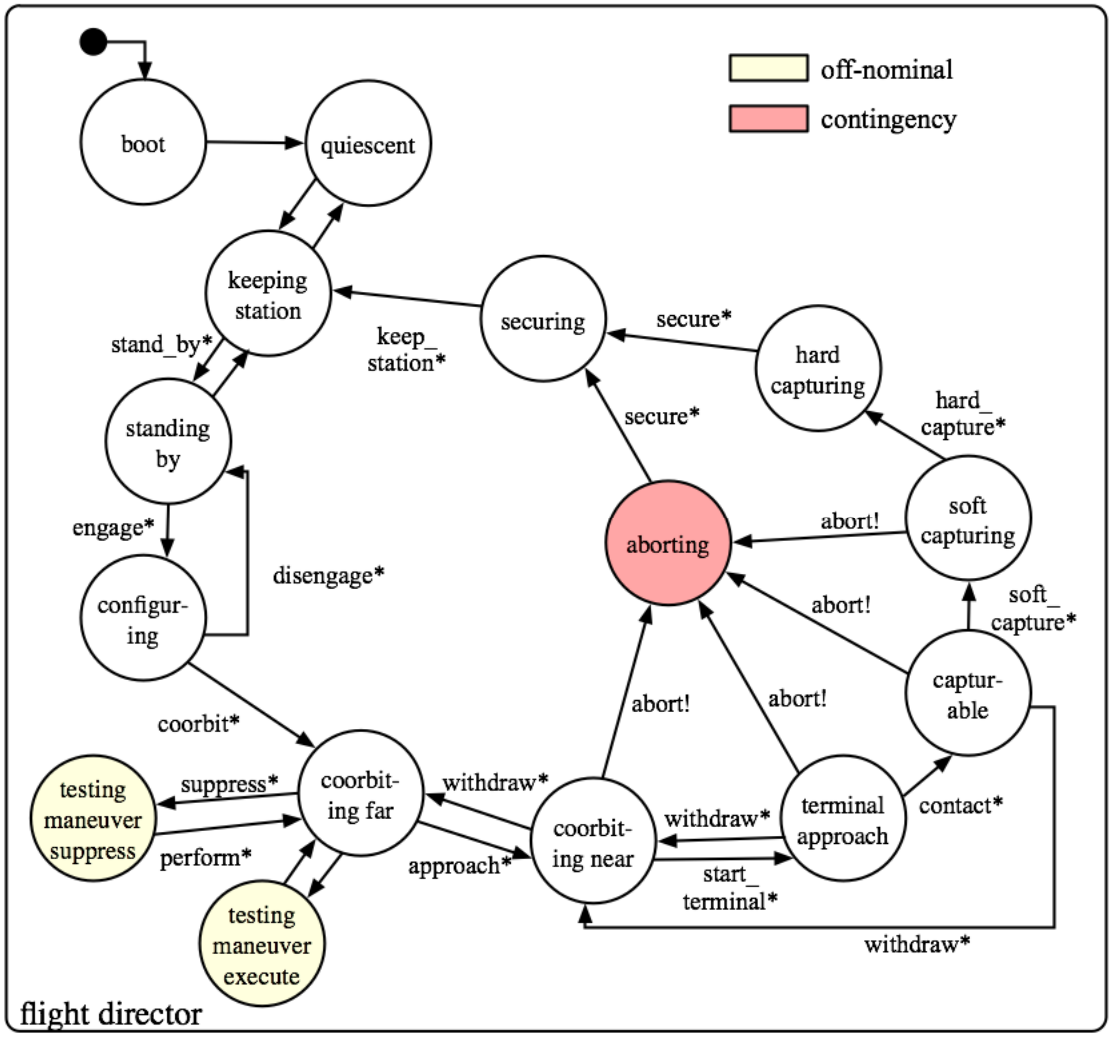
- Use executive to direct lower level spacecraft functions for approaching, sampling, ascending
- Comet, asteroid, lunar
- Tempel-1



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

Lessons learned from Phoenix EDL

- State-driven components easier to manage, more stable than monolithic sequences
- One-way coordination between components is simple and concise in VML
- Components provide excellent operational insight using existing VML telemetry
- Components make changing order of events easier (slew/sep vs. sep/slew)
- Signal interface allows functionality to be added and deleted easily (MARDI)
- Use of faster-than-realtime testing very important: run four days in 30 seconds
- State-driven EDL components excellent precursor to TAG state machines
- State diagrams greatly clarify complex GNC operations
- Synchronous coordination enables executive implementation in VML 2.1 and 2.2 for AutoGNC, building on Phoenix experience
- State-driven approach directly applicable to new missions like Mars Sample Return

State machines important new capability