Using Formal Methods to Analyze the Space Shuttle's GPS Change Request

Ben L. Di Vito

VíGYAN, Inc.
30 Research Drive
Hampton, VA 23666

3rd NASA Langley Formal Methods Workshop
11 May 1995

Shuttle Program Background

- Contractor organization
  - Rockwell International – Shuttle prime contractor
  - Loral Space Info. Sys. (was IBM) – Software contractor
  - Draper Lab – Experts in Guidance, Navigation and Control
- Software modifications are packaged as Change Requests (CRs)
  - Usually modest in scope, localized in function
  - Provide capabilities to meet specific mission needs
- Software releases are called Operational Increments (OIs)
  - Include one or more CRs – issued around once per year
- Requirements Analysis
  - Conducted by Loral Requirements Analysts (RAs) prior to turning CR over to development team
  - Once in development, problems are more costly to fix

Global Positioning System (GPS)

GPS is a satellite-based navigation system operated by DoD

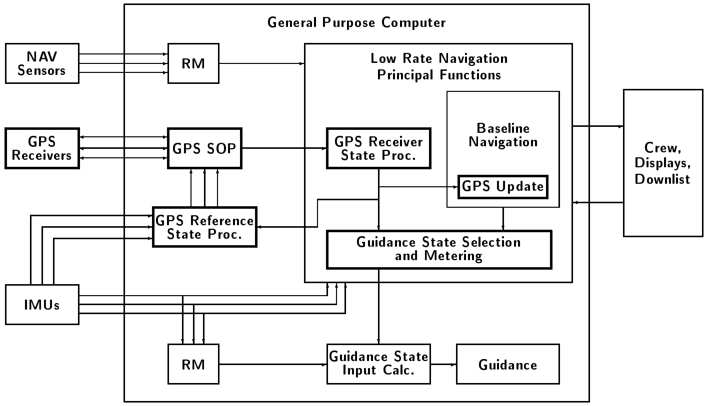
- Constellation of 24 satellites in high orbits
- Receive-only system requires dedicated hardware
  - Need to track 4 or more satellites simultaneously
  - Separate signals need to be recovered after undergoing code division multiplexing
- Receivers solve for position and velocity (and time)
  - Standard Positioning Service gives 100m accuracy
  - Precise Positioning Service gives 10m accuracy
- DoD phasing out TACAN navigation system by 2000

GPS Change Request

Shuttle Program is undertaking a large and complex CR to add GPS navigation capabilities to the Shuttle fleet

- Motivation: loss of TACAN navigation system
  - Need equivalent navigation aid during entry and landing
- Two-phase integration plan
  - Single-string implementation (one receiver)
  - Full-up implementation (three receivers)
- Integrated architecture
  - GPS receivers provide navigation data to General Purpose Computers (GPCs)
  - Several new Principal Functions added to software
  - Many smaller changes made to existing navigation software
- GPS CR more complex than typical CR

GPS Integrated Architecture



Approach to Applying Formal Methods

We are pursuing a selective application of formal methods

- Focus on core subset involving new principal functions
  - GPS Receiver State Processing
  - GPS Reference State Processing
- Use PVS to model principal function requirements
  - Derive stylized specification approach tailored to Shuttle software
  - Maintain traceability to existing requirements
  - Emphasize readability by nonexperts
- Formulate properties and attempt proving later, if warranted
  - Possible candidate is feedback loop from receivers to Receiver State Processing and back

Description of GPS CR Subset

Principal functions are decomposed into subfunctions

- Receiver State Processing
  - GPS IMU Assign
  - GPS Navigation State Propagation
  - GPS State Vector Quality Assessment
  - GPS State Vector Selection
  - GPS Reference State Announced Reset
- Reference State Processing
  - GPS External Data Snap
  - IMU GPS Selection
  - GPS Reference State Initialization and Reset
  - GPS Reference State Propagation

PVS Modeling of Principal Functions

Principal functions are regularly scheduled software entities

- Execution environment provides a large variable space that is read from and written into
- Interface is defined by explicit inputs and outputs enumerated in tables
- Principal functions are composed of several subfunctions invoked sequentially
  - Inputs are “passed down” to subfunctions
  - Outputs are “passed up” from subfunctions
- Local variables may be either transient or persistent
- We use a simple state machine model encoded in PVS
  - Principal function is represented as a single state-transition function

$$M : I \times S \rightarrow [O \times S]$$

PVS Modeling of Principal Functions (Cont'd)

Abstract structure in PVS notation:

```
pf_result: TYPE = [# output: pf_outputs, state: pf_state #]

principal_function (pf_inputs, pf_state,
                    pf_I_loads, pf_K_loads,
                    pf_constants) : pf_result =

      (# output := <output expression>,
         state  := <next-state expression>
      #)
```

- Principal functions use two kinds of variable data (input values, previous-state values) and three kinds of constant data (I-loads, K-loads, constants)
- Executing a principal function produces output values and next-state values
- All side effects are to be captured by this model

Requirements for Receiver State Processing

Requirements for Receiver State Processing (Cont'd)

Modeling of Receiver State Processing

Selected data types rendered in PVS

```
major_mode_code:    TYPE = nat
mission_time:       TYPE = real
GPS_id:             TYPE = {n: nat | 1 <= n & n <= 3}

receiver_mode:      TYPE = {init, test, nav}
AIF_flag:           TYPE = {auto, inhibit, force}

M50_axis:           TYPE = {Xm, Ym, Zm}
position_vector:    TYPE = [M50_axis -> real]
velocity_vector:    TYPE = [M50_axis -> real]
GPS_positions:      TYPE = [GPS_id -> position_vector]
GPS_velocities:     TYPE = [GPS_id -> velocity_vector]

GPS_predicate:      TYPE = [GPS_id -> bool]
GPS_times:          TYPE = [GPS_id -> mission_time]
GPS_FOM_vector:     TYPE = [GPS_id -> GPS_figure_of_merit]
tolerance_vector:   TYPE = [GPS_id -> real]
WGS84_to_EF_matrix: TYPE = [earth_fixed_axis -> [WGS84_axis -> real]]
```

Sample Subfunction of Receiver State Processing

```
ref_state_anncd_reset_out: TYPE = [#
      GPS_anncd_reset_avail:    GPS_predicate,
      GPS_anncd_reset:          GPS_predicate,
      R_ref_anncd_reset:        GPS_positions,
      T_anncd_reset:            mission_time,
      T_ref_anncd_reset:        mission_time,
      V_IMU_ref_anncd_reset:    velocity_vector,
      V_ref_anncd_reset:        GPS_velocities
      #]

ref_state_announced_reset(DT_anncd_reset,
                          GPS_DG,
                          GPS_SW_cap,
                          R_GPS,
                          T_anncd_reset,
                          T_current_filt,
                          T_GPS,
                          V_current_GPS,
                          V_GPS) : ref_state_anncd_reset_out
```


Principal Function Interface Types

```
rec_sp_inputs: TYPE = [#
      crew_deselect_rcvr:     GPS_predicate,
       . . .                   . . .
      V_GPS_ECEF:             GPS_velocities_WGS84,
      V_IMU2_save:            GPS_velocities
      #]

rec_sp_state: TYPE = [#
      GPS_DG_prev:            GPS_predicate,
       . . .                   . . .
      V_last_GPS_sel:         velocity_vector
      #]

rec_sp_I_loads: TYPE = [#
      acc_prop_min:           real,
       . . .                   . . .
      M_WGS84_to_EF:          WGS84_to_EF_matrix,
      sig_diag_GPS_nom:       cov_diagonal_vector
      #]
```


Principal Function Interface Types (Cont'd)

```
rec_sp_K_loads: TYPE = [#
      GPS_SW_cap:             num_GPS
      #]

rec_sp_constants: TYPE = [#
      deg_to_rad:             real,
      earth_rate:             real,
      G0:                     real,
      nautmi_per_ft:          real
      #]

rec_sp_outputs: TYPE = [#
      corr_coeff_GPS:         real,
      DELR_ratio_QA2_display: GPS_ratios,
       . . .                   . . .
      V_ref_anncd_reset:      GPS_velocities
      #]

rec_sp_result: TYPE = [# output: rec_sp_outputs, state: rec_sp_state #]
```


Principal Function Specification

```
GPS_receiver_state_processing((rec_sp_inputs:    rec_sp_inputs),
                              (rec_sp_state:     rec_sp_state),
                              (rec_sp_I_loads:   rec_sp_I_loads),
                              (rec_sp_K_loads:   rec_sp_K_loads),
                              (rec_sp_constants: rec_sp_constants) )
                              : rec_sp_result =
  LET
      IMU_assign_out =
          IMU_assign(
              V_current_filt       (rec_sp_inputs),
              V_IMU2_save          (rec_sp_inputs)),

      nav_state_prop_out =
          nav_state_propagation(
              acc_prop_min         (rec_sp_I_loads),
               . . .                . . .
              V_GPS_ECEF           (rec_sp_inputs),
              V_last_GPS           (IMU_assign_out),
              V_last_GPS_sel       (rec_sp_state) ),
```

Principal Function Specification (Cont'd)

```
SV_qual_assess_out =
    state_vector_quality_assessment(
        GPS_DG              (rec_sp_inputs),
         . . .               . . .
        V_GPS               (nav_state_prop_out) ),

state_vect_sel_out =
    state_vector_selection(
        corr_coeff_GPS_nom  (rec_sp_I_loads),
         . . .               . . .
        V_GPS_sel           (nav_state_prop_out) ),

ref_st_ann_reset_out =
    ref_state_announced_reset(
        DT_anncd_reset      (rec_sp_I_loads),
         . . .               . . .
        V_GPS               (nav_state_prop_out) )
```


Principal Function Specification (Cont'd)

```
IN
   (# output := (#
        corr_coeff_GPS :=      corr_coeff_GPS     (state_vect_sel_out),
         . . .                  . . .              . . .
        GPS_fail_QA4 :=        GPS_fail_QA4       (SV_qual_assess_out),
        GPS_lat :=             GPS_lat            (state_vect_sel_out),
        GPS_lon :=             GPS_lon            (state_vect_sel_out),
         . . .                  . . .              . . .
        V_ref_anncd_reset :=   V_ref_anncd_reset  (ref_st_ann_reset_out)
        #),

     state := (#
        GPS_DG_prev :=         GPS_DG_prev        (SV_qual_assess_out),
        GPS_SV_sel_avail :=    GPS_SV_sel_avail   (state_vect_sel_out),
         . . .                  . . .              . . .
        V_GPS_sel :=           V_GPS_sel          (state_vect_sel_out),
        V_last_GPS_sel :=      V_last_GPS_sel     (nav_state_prop_out)
        #)
    #)
```


Results (In Progress)

Experience with effort so far:

- **Outlook is promising, but it's still early**
- **CR requirements are still converging**
  - **Another revision cycle is likely**
  - **After next cycle, FM results should be more meaningful**
- **FM-based review is helping requirements analysis**
  - **Many interface errors being detected**
- **PVS can be used effectively to formalize this application**
  - **Custom specification approach should be easy to duplicate**
  - **Good prospects for continuation by nonexperts**
  - **Specification activity assisted by tools, but doing manual specification is also feasible here**

Feedback from Requirements Analysts

Some RAs are optimistic about potential impact of FM

- **Approach used is helpful in detecting two classes of errors:**
  - **"Requirements meet the CR author's intent; CR will work"**
  - **"Interfaces Documented and Consistent"**
- **Preliminary comparison with conventional process**
  - **Errors detected in Reference State Processing versus those also found by current process**

Error Severity	With FM	Existing
High Major	2	0
Low Major	5	1
High Minor	17	3
Low Minor	6	0
Totals	30	4

Continuation Plans

Plan for near term CR analysis

- Continue elaborating formal specifications using current style
  - Complete two principal functions under way
  - Consider adding other functions as resources allow
  - Collect feedback and data from RAs
- Go down to moderate level of detail
  - Incorporate enough detail to capture branching conditions
- Evaluate tradeoffs of formalizing subsystem properties
  - Properties based on sequences of state vectors
  - Limited proving may be worthwhile

Summary

- Formal methods aiding requirements analysis on a significant Shuttle CR
- Specification techniques customized for Shuttle software
- So far, keeping up with requirements analysis process
  - Enables meaningful comparisons
- Early results are encouraging
- Remains to be seen whether techniques can be transferred to and adopted by RAs