





GNC software production process

M.Delpech – PY Guidotti - JC Berges



Context

- SSC has developed a significant experience in code generation process using Mathworks tools → GNC software was auto coded for the SMART-1 mission (lunar probe)
- Process reused for PRISMA to develop GNC software and numerous components of the flight software (including I/O)
- Matlab / Simulink is the only environment to perform GNC algorithm integration and validation → 2 possibilities for delivering software contributions:
 - traditional source code (C, C++) encapsulated within a special Simulink block (S_function)
 - Simulink based design
- Design rules and recommendations are given by SSC to partners using Simulink for code development (Model Based On Board Software)

Besides the technical interest, CNES choice of the second option was driven by the short development timeframe





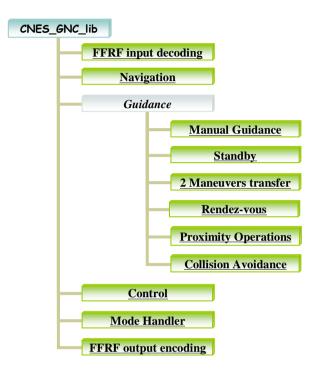
Development approach

- Internal development performed by the GNC team
 - CNES DCT/SB/PS et DCT/SB/MO
 - Partnership with Astrium and TAS for 2 guidance functions
- 11 functions integrated in a Simulink library:
 - Simulink = model oriented approach that simplifies the development of complex algorithms and simulators
 - From initial prototyping, to simulator for studies, until flight software
 - Direct interface with Matlab
- → CNES_GNC_lib = Simulink library delivered to SSC
- Quick development :
 - Start nominal mission = 2006
 - First integration in Prisma flight software = january 2008
 - Flight version = march 2009
 - → Even faster for the extended mission activities

Matlab/Simulink/RTW used by SSC to develop:

- flight software
- numeric satellite simulator (Satsim)









Simulink development

Utilisation intensive de Simulink

- 58 entrées (524 éléments), 27 sorties (566 éléments), 35 paramètres modèles
- 11 librairies internes pour 11 fonctions dédiées
- 679 sous-systèmes, 11553 blocks Simulink
- 3 s-functions codées en C (Navigation + FFRF_IF)

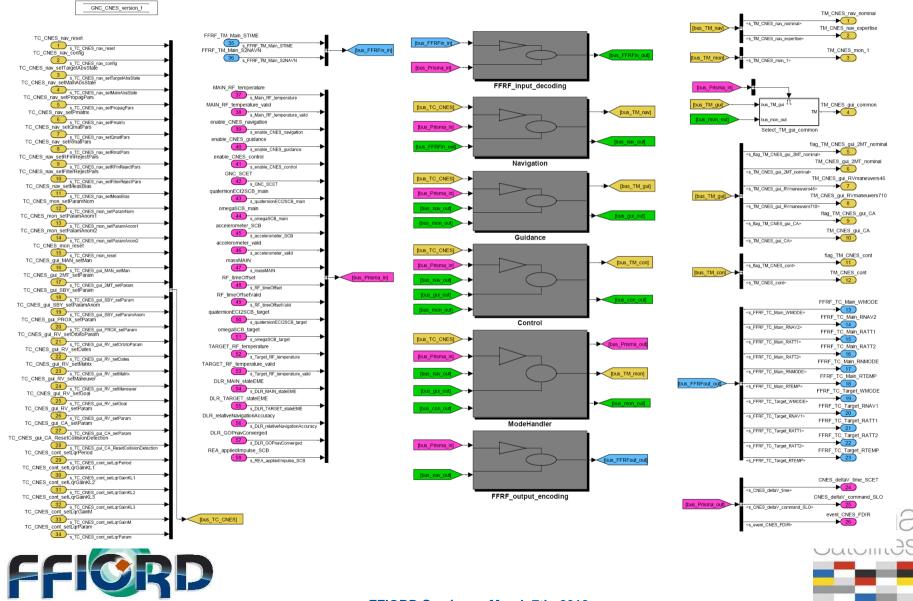
>TC_CNES_nav_reset	
>TC_CNES_nav_config	TM_CNES_nav_nominal
>TC_CNES_nav_setTargetAbsState	
>TC_CNES_nav_setMainAbsState	TM_CNES_nav_expertise
>TC_CNES_nav_setPropagPars	
>TC_CNES_nav_setPmatrix	THE CAUCE man 4
>TC_CNES_nav_setQmatPars	TM_CNES_mon_1
>TC_CNES_nav_setRmatPars	
>TC_CNES_nav_setRFmRejectPars	TM_CNES_gui_common
>TC_CNES_nav_setFilterRejectPars	
>TC_CNES_nav_setMeasBias	flag_TM_CNES_gui_2MT_nominal
>TC_CNES_mon_setParamNom	
>TC_CNES_mon_setParamAnom1	TM_CNES_gui_2MT_nominal
TC_CNES_mon_setParamAnom2	
>TC_CNES_mon_reset	TM_CNES_gui_RVmaneuvers46
TC_CNES_gui_MAN_setMan	
TC_CNES_gui_2MT_setParam	TM CNES gui RVmaneuvers710
>TC_CNES_gui_SBY_setParam	im_crecs_gai_revilaneavers/10
TC_CNES_gui_SBY_setParamAnom	
>TC_CNES_gui_PROX_setParam	flag_TM_CNES_gui_CA
>TC_CNES_gui_RV_setOrbitoParam	
>TC_CNES_gui_RV_setDates	TM_CNES_gui_CA
TC_CNES_gui_RV_setMatrix	
>TC_CNES_gui_RV_setManeuver	flag_TM_CNES_cont
TC_CNES_gui_RV_setGoal	
>TC_CNES_gui_RV_setParam	TM_CNES_cont
TC_CNES_gui_CA_setParam	
TC_CNES_gui_CA_ResetCollisionDetection	FFRF TC Main WMODE
TC_CNES_cont_setLqrPeriod	TTREETC_MUNI_WWODE

FC_CNES_cont_setLqrGainKL1	FFRF TC Main RNAV2
CC_CNES_cont_setLqrGainKL2	
FC_CNES_cont_setLqrGainKL3	
FC_CNES_cont_setLqrGainM	FFRF_TC_Main_RATT1
FC_CNES_cont_setLqrParam	
FFRF_TM_Main_STIME	FFRF_TC_Main_RATT2
FRF_TM_Main_\$2NAVN	
MAIN_RF_temperature	FFRF_TC_Main_RNMODE
MAIN_RF_temperature_valid	
enable_CNES_navigation	FFRF TC Main RTEMP
enable_CNES_guidance	
enable_CNES_control	FEDE TO Torrest WMODE
GNC_SCET	FFRF_TC_Target_WMODE
quaternionECl2SCB_main	
omegaSCB_main	FFRF_TC_Target_RNAV1
accelerometer_SCB	
accelerometer_valid	FFRF_TC_Target_RATT1
massMAIN	
RF_timeOffset	FFRF_TC_Target_RATT2
RF_timeOffsetValid	
quaternionECI2SCB_target	FFRF TC Target RTEMP
omegaSCB_target	
TARGET_RF_temperature	
TARGET_RF_temperature_valid	CNES_deltaV_time_SCET
DLR_MAIN_stateEME	
DLR_TARGET_stateEME	CNES_deltaV_command_SLO
DLR_relativeNavigationAccuracy	
DLR_GOPnavConverged	event_CNES_FDIR
REA appliedImpulse SCB	



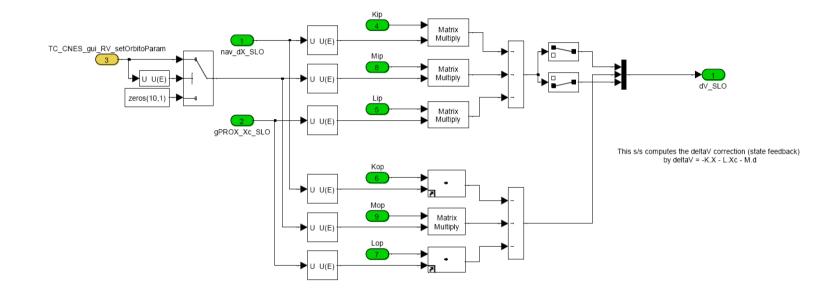


Simulink development Software high level architecture





Simulink development

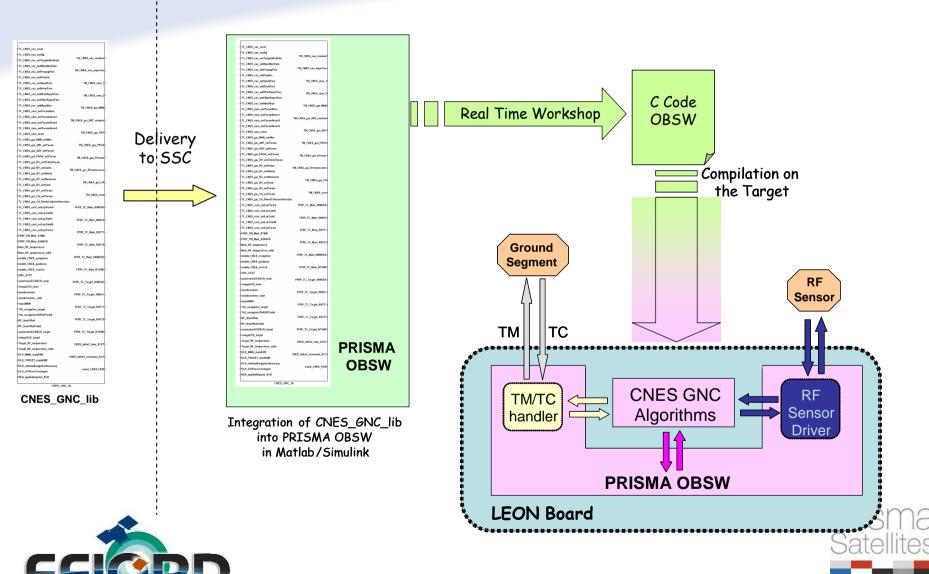








Integration and code generation (option 1)



RF Sensor

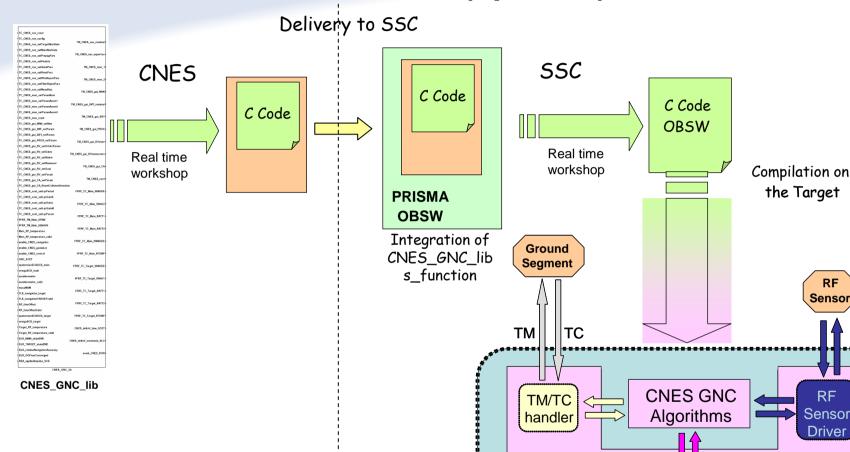
Sensor

Driver

PRISMA OBSW



Integration and code generation (option 2)





LEON Board



Validation approach

CNES has chosen an hybrid approach:

- Deliver the software to SSC according to option 1
 - → To limit the validation risk and complexity given the tight development schedule (with a new software production process)
- Perform code generation in CNES to validate independently the code generation process at CNES_GNC_lib level
 - → To reduce the number of iterations and cross validate in case of anomaly

Validation of generated code in 2 steps

- 1. validation in Simulink environment
- 2. validation on the real target (LEON processor)

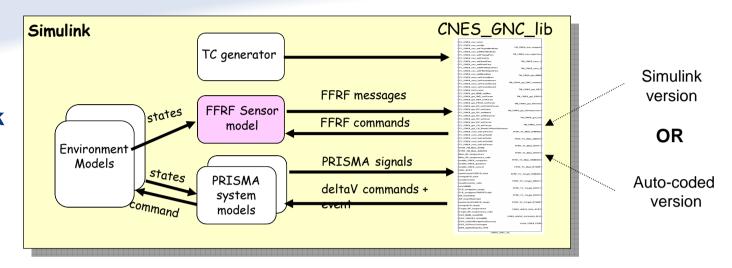




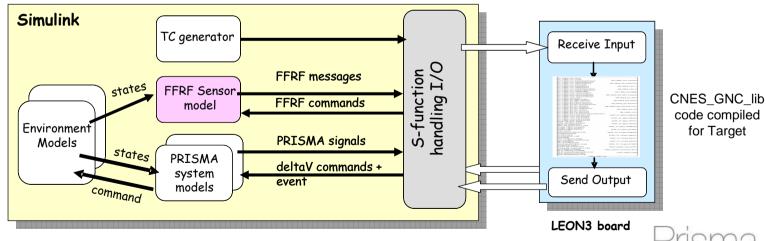


GNC validation at CNES

Step 1
All Simulink



Step 1 Hybrid simulation







Software requirements

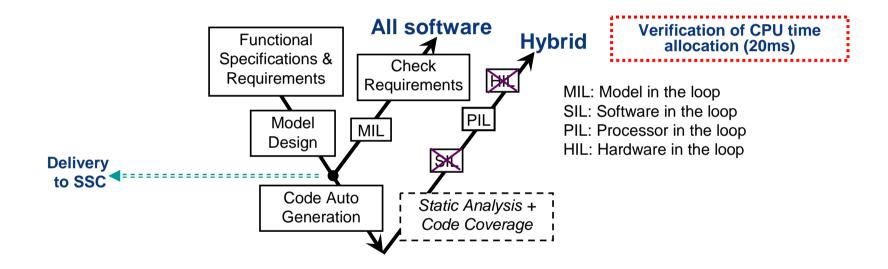
- To be validated (SSC point of view)
 - Library interface (list of I/O ports with dimensions and types)
 - CPU load (20 ms for a CNES_GNC_lib cycle @1Hz)
 - Coding rules (mostly guidelines)
 - Safety: generates no exceptions
- To be validated (CNES point of view)
 - SSC requirements +
 - GNC functional behavior
 - GNC performance behaviour







GNC validation at CNES









GNC validation at SSC

Incremental CNES GNC lib validation with SSC testbenches

SFPT-2 Satellite Test Bench	April 2009	System and Functional Test Bench: → Software validation
System tests OPSIM	October 2009	System tests campaign: → verification of procedures and end to end validation (all scenario processing to telemetry delivery).
System tests GNCViewer	October 2009	System tests campaign: → Validation of scenarii
Patch Validation GNCViewer + OPSIM	March 2010	Flight operations / extended mission Validation of software patches

- Satellite Test Bench: AIT with FFRF emulator in the loop
- <u>OPSIM</u>: Operational Simulator real time operational testbench with FFRF emulator
- <u>GNCViewer</u>: all Simulink tool including the Prisma GNC software and capable to run operational procedures and mission scenarii (20x real time). Includes a FFRF sensor model.







GNC validation at SSC

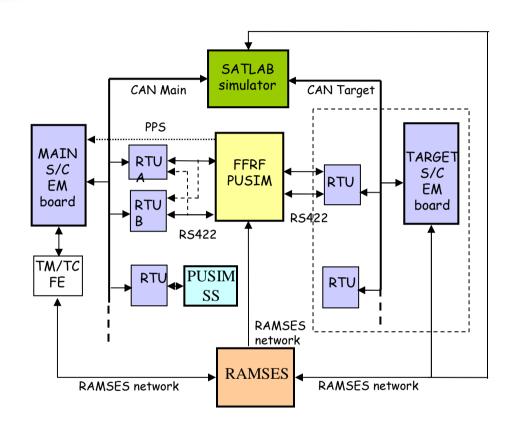
■ OPSIM testbench

Main Characteristics

- Most of the avionics FM components are integrated in the satellites (emulators replace instruments that cannot be stimulated)
- Complete TM/TC chain
- FFRF PUSIM connected to both S/C

Test objectives

- GNC software validation (closed loop tests)
- Procedure / program validation



All validations were performed without any hardware dynamic simulator !!







GNC validation at SSC

Simplified process for extended mission:

→ The new software was regarded as a patch

GNCViewer	September 2011	Functional and performance validation → cross-check of results on specific mission programs
OPSIM	September 2011	Code validation -> including real time allocation, TM processing

Relevant process → No anomaly was detected in flight!

Code generation + pragmatic validation approach

→ from new GNC design to software upload: 4 months



