Z and ProCoSA based specification of a distributed FDIR in a satellite formation

Ch. Castel**, J.-Ch. Chaudemar*, J.-F. Gabard**, C. Tessier**
***ONERA-DCSD, *SUPAERO*

Abstract

On-board FDIR (Fault Detection, Isolation and Recovery) is contemplated for autonomous satellite formations. Several FDIR strategies have been specified using the Petri net - based software ProCoSA (for the dynamic aspects) on the one hand, and the set theory - based Z specification language (for the static aspects) on the other hand. ProCoSA enables to specify the different state changes triggered by the different events within the formation; Z enables to describe the relations and constraints (invariants) between the state variables.
The paper focuses on a global specification including both the dynamic and static aspects, through a formal link between ProCoSA and Z. The link is implemented and allows some properties of the strategies to be checked.

1 Introduction

The autonomous formation flying of multiple spacecraft to replace a single large satellite will be an enabling technology for a number of future missions. Potential applications include synthetic apertures for surveillance and high-resolution interferometry missions, or for taking widespread field measurements for atmospheric survey missions [Cra].Very precise autonomous coordination and control differentiate formations from constellations. The challenge is to develop both the software and the hardware to allow separate, unconnected spacecraft to function as if they were a single, solid structure [Nas]. Spacecraft within a formation may be different from one another and the different parts of one instrument may be distributed among several spacecraft.
FDIR (Fault Detection, Isolation and Recovery) is the means to detect off-nominal conditions, isolate the problem to a specific subsystem/component, and recover of vehicle systems and capabilities [NAS05]. Formation flying brings a new concept in FDIR, i.e. the *formation* has to be considered as an entity in itself. Indeed the scientific mission is performed by the formation (and not by the individual spacecraft). Therefore specific FDIR strategies [CGL$^+$06] have to be considered in order to deal with formation specific failures e.g. instrument failure, problems with the formation geometry, inter-spacecraft communication failures.

In this paper, FDIR is considered as an operational function that contributes to the autonomy of the system and whose main purpose is to maintain the availability of each satellite of the formation for the mission. A global specification of FDIR concepts including both the dynamic and static aspects, through a formal link between ProCoSA and Z, is presented. The link is implemented and allows some properties of the strategies to be checked. As an example, a typical anomaly that may affect the formation integrity, namely a violation of the "Keep Out Zone"[1] is considered with a centralised centralised strategy.
The paper is organised as follows : after a short presentation of the Z notation, the next section describes FDIR concepts for a satellite formation and their implementation for a unique anomaly case by using Z specification language. Then the centralised strategy we have designed is executed with a ProCoSA simulation refined by Z data and constraint definitions. We explain the linking between Z schemas and ProCoSA Petri nets for our case study. An analysis section presents the interest to associate Z-modelling and ProCoSA simulation for a safety-critical system like a satellite formation called Simbol-X.

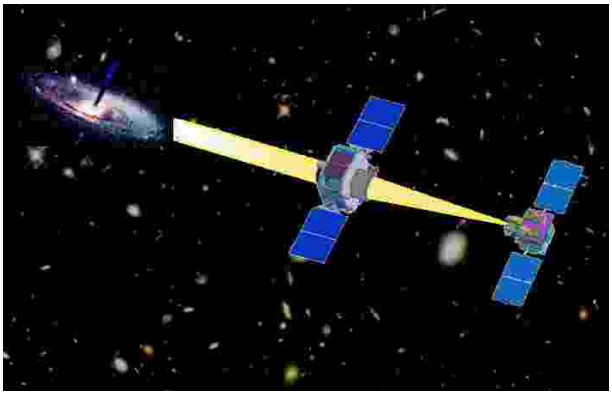

Figure 1: Simbol-X mission (http://apc-p7.org/APC_CS/Experiences/SIMBOLX/index.phtml)

2 Z-modelling of FDIR within a satellite formation

Z [Spi00, But01] is a formal specification language based on the set theory and the predicate logic. The Z specification of a system consists of state variables, an initialisation and a set of operations on state variables. *Invariants*, which represent constraints which must always be satisfied, are associated with state variables. The basic element of Z specification is the *schema*.

[1] Keep Out Zone (KOZ) is defined as a safety sphere around each satellite that another satellite must not enter.

2.1 Z-modelling of FDIR concepts

The first step is to model the main concepts that take part into the formation FDIR. This is a static view of FDIR agents with their main constraints.

FDIR Regardless of the components implementing it, FDIR is considered as a kind of operational function *FONCTION_OP*, including non-empty finite sets of functions for detection (*detections*) and recovery (*reconfigurations*), and dealing with a fixed non-empty finite set of defaults (*traite*).

FDIR

$$
\begin{array}{l}
FONCTION\_OP \\
detections : \mathbb{F}_1\, DETECTION \\
traite : \mathbb{F}_1\, DEFAUT \\
reconfigurations : \mathbb{F}_1\, RECONFIGURATION
\end{array}
$$

SIMBOLX We focus on a particular system called Simbol-X [Sim07], composed of two satellites, *sat1, sat2* within a *formation*, and a non-empty set of ground stations, *stationsol*. This system is characterised by a set of operational functions, *fct_ops* in relation to each other (*participe_a*).
The predicate part states that the formation is composed of one detector satellite and one mirror satellite whose KOZ radius are specified, and the different operational functions are clearly distinct if they relate to either the formation or the detector satellite or the mirror satellite.

SIMBOLX

$$
\begin{array}{l}
sat1, sat2 : SATELLITE \\
formation : FORMATION \\
stationsol : \mathbb{F}_1\, SOL \\
fct\_ops : \mathbb{F}_1\, FONCTION\_OP \\
participe\_a : FONCTION\_OP \leftrightarrow FONCTION\_OP \\
\hline
formation.satellites = \{sat1, sat2\} \\
sat1.type = detecteur \land sat1.koz = 18 \\
sat2.type = miroir \land sat2.koz = 15 \\
\langle formation.dispose\_de\_fop, sat1.dispose\_de\_fop, \\
sat2.dispose\_de\_fop\rangle \ \mathsf{partition}\ fct\_ops \\
\mathrm{dom}\, participe\_a = fct\_ops \\
\mathrm{ran}\, participe\_a = fct\_ops \\
\forall fct : FONCTION\_OP \mid fct \in fct\_ops \bullet \{(fct \mapsto fct)\} \cap participe\_a = \emptyset
\end{array}
$$

SATELLITE A satellite is a kind of FDIR actor (*ACTEUR*) characterised by a *type* (detector or mirror), a *koz* radius and some operational functions (*dispose_de_fop*)

such as a FDIR function (*dispose_de_fdir*), an intersatellite communication function (*dispose_de_fcom_sat*), and a communication function with a ground station (*dispose_de_fcom_sol*).
The single constraint relates to a minimum value for the KOZ radius.

SATELLITE

$$
\begin{array}{l}
ACTEUR \\
type : ROLE \\
koz : \mathbb{N}_1 \\
chainefonctionnelle : \mathbb{F}_1\ CHAINE\_FCT \\
dispose\_de\_fop : \mathbb{F}_1\ FONCTION\_OP \\
dispose\_de\_fdir : FDIR \\
dispose\_de\_fcom\_sat : FCT\_COM\_SAT \\
dispose\_de\_fcom\_sol : FCT\_COM\_SOL \\
\hline
koz \geq 5
\end{array}
$$

SOL A ground station is a kind of actor (*ACTEUR*) characterised by some operational functions (*dispose_de_fop*) and a communication function with a satellite (*dispose_de_fcom_sat*).

SOL

$$
\begin{array}{l}
ACTEUR \\
dispose\_de\_fop : \mathbb{F}_1\ FONCTION\_OP \\
dispose\_de\_fcom\_sat : FCT\_COM\_SAT
\end{array}
$$

FORMATION The formation is a kind of FDIR actor (*ACTEUR*), composed of a non-empty finite set of satellites (*satellites*) characterised by a distance (*distance*), a status related to the KOZ, *status_vkoz* which can take two values that are *normal* for a normal status and *en_panne* for a failure status, and some operational functions (*dispose_de_fop*) such as a FDIR function (*dispose_de_fdir*).
The predicate part states properties concerning distance between both satellites for a normal status.

FORMATION

$$
\begin{array}{l}
\mathit{ACTEUR} \\
\mathit{satellites} : \mathbb{F}_1\, \mathit{SATELLITE} \\
\mathit{distance} : \mathit{SATELLITE} \times \mathit{SATELLITE} \pfun \mathbb{N}_1 \\
\mathit{dispose\_de\_fop} : \mathbb{F}_1\, \mathit{FONCTION\_OP} \\
\mathit{dispose\_de\_fdir} : \mathit{FDIR} \\
\mathit{status\_vkoz} : \mathit{ETAT\_V\_KOZ} \\
\hline
\mathrm{dom}\, \mathit{distance} \subseteq \mathit{satellites} \times \mathit{satellites} \\
\mathit{status\_vkoz} = \mathit{normal} \Leftrightarrow \\
(\forall\, \mathit{sat}1, \mathit{sat}2 : \mathit{SATELLITE} \mid (\mathit{sat}1, \mathit{sat}2) \in \mathrm{dom}\, \mathit{distance} \\
\bullet\ \mathit{distance}(\mathit{sat}1, \mathit{sat}2) > \mathit{sat}1.\mathit{koz} + \mathit{sat}2.\mathit{koz})
\end{array}
$$

As far as FDIR itself is concerned, several strategies have been defined [CGL$^+$06] and have been implemented with ProCoSA. Only a centralised strategy is dealt with in this paper.

OPERATION A recovery operation is characterised by its type (*nature*), i.e. reset, redundancy switch or movement operations.

OPERATION

$$
\begin{array}{l}
\mathit{nature} : \mathit{TYPE\_OPERATION}
\end{array}
$$

STRATEGIE_FDIR An FDIR strategy has a type, *type_strat*, i.e. centralised, mixed or distributed. It concerns the relationship between a recovery operation and an operation function, connects FDIR function to FDIR cases, implies FDIR functions, intersatellite communications and communications with the ground, and a non-empty set of defaults.
The invariant relationships stipulate that these functions are well associated to the concerned satellites.

STRATEGIE_FDIR

$type\_strat : TYPE\_STRATEGIE\_FDIR$
$reconfig : OPERATION \leftrightarrow RECONFIGURATION$
$satisfait : FDIR \leftrightarrow FDIR\_CAS$
$fdir\_sat : SATELLITE \leftrightarrow FDIR$
$com\_sat : SATELLITE \leftrightarrow FCT\_COM\_SAT$
$com\_satsol : SATELLITE \leftrightarrow FCT\_COM\_SOL$
$defauts : \mathbb{F}_1\, DEFAUT$

$\forall sat : SATELLITE;\ fd : FDIR \bullet (sat, fd) \in fdir\_sat$
$\Rightarrow fd = sat.dispose\_de\_fdir$
$\forall sat : SATELLITE;\ fc : FCT\_COM\_SAT \bullet (sat, fc) \in com\_sat$
$\Rightarrow fc = sat.dispose\_de\_fcom\_sat$
$\forall sat : SATELLITE;\ fc\_sol : FCT\_COM\_SOL \bullet (sat, fc\_sol) \in com\_satsol$
$\Rightarrow fc\_sol = sat.dispose\_de\_fcom\_sol$

S_CENTR_1 This is one of the centralised FDIR strategies that we have designed for satellite formations. This strategy is such as each satellite carries its own detection function and only the detector satellite *sat1* carries the recovery capabilities. Moreover, only *sat1* communicates with the ground for FDIR.

S_CENTR_1

STRATEGIE_FDIR
SIMBOLX

$type\_strat = centralisee$
$defauts \subseteq sat1.dispose\_de\_fdir.traite \cup sat2.dispose\_de\_fdir.traite$
$fdir\_sat = \{(sat1, sat1.dispose\_de\_fdir), (sat2, sat2.dispose\_de\_fdir)\}$
$sat1.dispose\_de\_fdir.detections \neq \emptyset$
$sat2.dispose\_de\_fdir.detections \neq \emptyset$
$sat1.dispose\_de\_fdir.reconfigurations \neq \emptyset$
$sat1.dispose\_de\_fdir.reconfigurations \subseteq \operatorname{ran} reconfig$
$sat2.dispose\_de\_fdir.reconfigurations \cap \operatorname{ran} reconfig = \emptyset$
$com\_sat = \{(sat1, sat1.dispose\_de\_fcom\_sat), (sat2, sat2.dispose\_de\_fcom\_sat)\}$
$com\_satsol = \{(sat1, sat1.dispose\_de\_fcom\_sol)\}$

2.2 State evolution for a KOZ violation case

The initial values of the Simbol-X formation are given by the following Z schema:

Init_FORMATION

$$
\begin{array}{l}
FORMATION \\
\Xi SIMBOLX \\
\hline
status\_vkoz = normal \\
sat1.type = detecteur \land sat1.koz = 18 \\
sat2.type = miroir \land sat2.koz = 15
\end{array}
$$

Let us consider the operations due to a formation fault such as a KOZ violation. This fault (*ev_fdir?*) is detected when both satellites get closer.

PANNE_V_KOZ

$$
\begin{array}{l}
\Xi SIMBOLX \\
\Delta FORMATION \\
ev\_fdir? : DEFAUT \\
\hline
ev\_fdir? \in formation.dispose\_de\_fdir.traite \\
(sat1, sat2) \in \operatorname{dom} distance \\
distance(sat1, sat2) > sat1.koz + sat2.koz \\
distance'(sat1, sat2) \leq sat1.koz + sat2.koz
\end{array}
$$

For the detector satellite and for the mirror satellite, the fault is respectively expressed by *anom_S1?* and *anom_S2?* and detected by their detection functions:

NORMAL2PANNE

$$
\begin{array}{l}
\Xi SIMBOLX \\
\Delta FDIR \\
anom\_S1? : DEFAUT \\
anom\_S2? : DEFAUT \\
\hline
anom\_S1? \in sat1.dispose\_de\_fdir.traite \\
anom\_S2? \in sat2.dispose\_de\_fdir.traite \\
sat1.dispose\_de\_fdir.detections \cap detections' \\
\neq detections \cap sat1.dispose\_de\_fdir.detections \\
sat2.dispose\_de\_fdir.detections \cap detections' \\
\neq detections \cap sat2.dispose\_de\_fdir.detections
\end{array}
$$

To express intersatellite communication, we define a non-exhaustive enumerated type of messages:

$$
\begin{array}{l}
MESSAGE ::= messok \mid messperteISL \mid messperteRF \mid messalarme \\
\mid messreconf \mid messmanoeuvre \mid messsol \mid messcritique
\end{array}
$$

The mirror satellite sends an alarm message to the detector satellite in order for it to deal with this default and develop an operational strategy.

$$INFO2S1 \mathrel{\widehat=} [mess! : MESSAGE \mid mess! = messalarme]$$

The KOZ violation requires a quick reaction of the formation. A high priority recovery strategy is planned by the FDIR satellite, i.e. the detector satellite, for the whole formation.

RECONF_F

$\Xi SIMBOLX$
$\Delta STRATEGIE\_FDIR$

$reconfig = \emptyset$
$reconfig' \neq \emptyset$

A message stating the appropriate manoeuvre is sent by the detector to the mirror satellite.

$$RECONF\_S2 \mathrel{\widehat=} [mess! : MESSAGE \mid mess! = messreconf \land mess! = messmanoeuvre]$$

The manoeuvre is finished when a normal state is recovered.

FIN_MANOEUVRE

$\Xi SIMBOLX$
$\Delta FORMATION$

$status\_vkoz = en\_panne$
$status\_vkoz' = normal$

Therefore the KOZ violation processing is: first detection, then a manoeuvre executed by both the satellites.

$$ViolationKOZ \mathrel{\widehat=} NORMAL2PANNE \fatsemi INFO2S1 \fatsemi RECONF\_F \fatsemi RECONF\_S2 \fatsemi FIN\_MANOEUVRE$$

3 ProCoSA simulation of KOZ violation

3.1 Petri nets

The simulation of KOZ violation with ProCoSA (figure 2) distinguishes three behaviours corresponding to the detector satellite state (*etat_S1*), the mirror satellite state (*etat_S2*), and the FDIR satellite (*FDIR_S1*), namely the detector satellite.
A KOZ violation fault affects both satellites and makes the *state-S_i* (*etat_Si*) Petri nets pass from the nominal (*normal*) to the fault state (*en_panne*) whereas *FDIR_S1* net passes from the nominal (*nominal_formation*) to the detection state (*D*). For *etat_S2*, an additional fault state is introduced to take into account the sending of an alarm message to the FDIR satellite if the intersatellite communication link is available (*COM_OK_S2_vers_S1* [2]). Then *FDIR_S1* passes to state *reactif_S1*

[2] COM_OK_S2_vers_S1 and COM_OK_S1_vers_S2 are two global places used in other Petri nets modelling the intersatellite communication state. In this paper, we don't focus on this aspect.

meaning that a security reaction is performed for this type of fault whereas *etat_S1* passes to state *reconf_en_cours*. Indeed, as *FDIR_S1* carries all the FDIR knowledge and algorithms, only *FDIR_S1* can perform a reaction, even if the fault is detected on another spacecraft. *FDIR_S1* then passes to state *att_reconf_S2* if the intersatellite link is available (*COM_OK_S1_vers_S2*) meaning that actions necessary to recovery are expected from the mirror satellite. Thus each *state-S_i* net passes to state *on-going reconfiguration* (*reconf_en_cours*), before going back to the nominal state. At last *FDIR_S1* goes back to the nominal state.

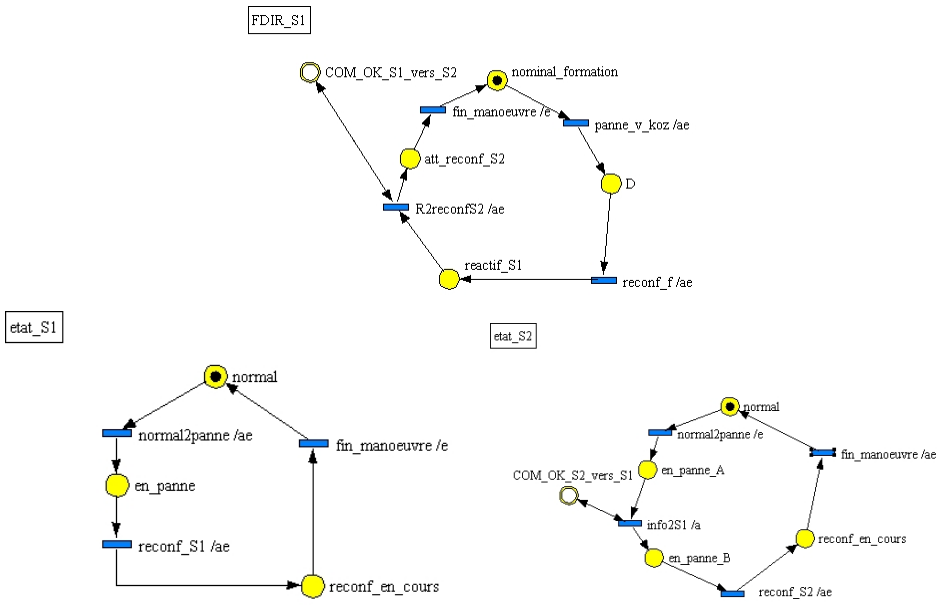

Figure 2: KOZ violation recovery simulated with ProCoSA

3.2 Linking Z schemas and ProCoSA Petri nets

As pointed out in [HH99, Xud01, PJ03], Z schemas are well suited to define data structures, system constraints and functional processing whereas Petri nets are a graph-based formal model for representing the control structures and dynamic behaviours of concurrent and distributed systems that cannot be explicitly described in Z. Accordingly a relationship between Z and Petri nets offers a coherent formalism of specification for designing reliable systems.

In this section, we present how to combine Z schemas and ProCoSA Petri nets for the specification of FDIR in a satellite formation. The first step consists in building a relevant Z model of a satellite formation. By analysing the requirement description of the system, we identify its main components (*FORMATION, SATELLITE, SOL*) and functionalities (*FDIR, OPERATION, STRATEGIE_FDIR*)

which define the overall system structure. Then, the definition of Z-operations (*PANNE_V_KOZ, NORMAL2PANNE, INFO2S1, RECONF_F, RECONF_S2, FIN_MANOEUVRE*) refines the model for the specific case of the KOZ violation. Furthermore, ProCoSA simulation enables to link Z-operations to Petri net transitions: one Z operation defines one Petri net transition. In the predicate part of a Z operation schema, the pre-condition part which is expressed through non-dashed variables is the guard specifying the enabling condition of the corresponding transition, whereas the post-condition part expressed through dashed variables defines its firing result. Moreover, ProCoSA events and messages can be associated with transitions and may respectively correspond to input and output variables in Z operations. According to the rule stated above, the Petri net transitions *panne_v_koz, normal2panne, info2S1, reconf_f, reconf_S2, fin_manoeuvre* correspond to operation schemas described in the Z model. The transition *reconf_S1* is added to mean that formation recovery strategy *reconf_f* is composed of an internal S1 recovery function. This recovery function is partially hinted in the *reconfig* variable of the Z-operation called *RECONF_F*. In fact, this transition is similar to *reconf_f*, so it corresponds to the same Z-operation *RECONF_F*.

In our model, there is only an implicit relationship between local variables in Z operation schemas and input or output places of a transition in ProCoSA Petri nets. Some expressions relate to the system state before and after transition firing, like the expression concerning detection functions in the predicate part of *NORMAL2PANNE* Z operation schema.

Furthermore, the initial marking of the ProCoSA Petri nets is consistent with the initial state schema in Z (*Init_FORMATION*).

For a complete simulation, a new ProCoSA procedure will be developed to simulate the intersatellite distance variation and the Z property concerning the distance and KOZ.

4 Analysis of Z-ProCoSA relationship

For consistency reasons, the two processes, i.e. the Z specification and the ProCoSA simulation, were jointly carried out in order to fully benefit from the advantages of each method. Z provides the formal aspect for the specification of the system, whereas ProCoSA allows to focus on dynamic aspects and the sequences of the state variations. This has enabled a better understanding of the formation behaviour faced with a KOZ violation anomaly by revealing not very precise requirements, e.g. which satellite first operates the collision avoidance manoeuvre.

Thanks to data and constraint definitions, the Z specification has allowed to modify an existing ProCoSA simulation of an FDIR centralised strategy that only took into

account the state evolutions.
Conversely, the ProCoSA simulation also contributes to develop the Z model by describing state changes and control flows between the various Petri nets, i.e. the behavioural aspect. The causal relation or state sequence is represented graphically: normal state, then fault detection, at last recovery with manoeuvre.

Compared to other models [HH99, Xud01, PJ03], the complexity of the approach seems lower and the Z-ProCoSA relationship analysis weakly relates to net places and their corresponding Z schemas and confidently relies on ProCoSA property analysis tool (place safety, detection of dead markings).

5 Conclusion

ProCoSA is suitable to take into account the behaviour of concurrent and distributed systems like FDIR for a satellite formation, whereas Z is well known for data abstraction and functional specification. The idea is that the combination of both formalisms leads to very reliable models.
The next steps in our work are the following:

- implement a hybrid simulation with discrete and continuous state variables, e.g. to simulate the intersatellite distance variation and the Z properties concerning the distance and KOZ ;
- refine the simulation by taking time into account, i.e. state duration, delay between satellite operations and concurrence between both satellites ; thus, we will test other FDIR strategies that need more time for converging or involve ground stations.
- define a formal methodology applying proof checking that is based on combined Z and Petri net specifications.

Appendix: ProCoSA

A Petri net $< P, T, F, B >$ is a bipartite graph with two types of nodes: P is a finite set of places; T is a finite set of transitions [DA05]. Arcs are directed and represent the forward incidence function $F : P \times T \rightarrow \mathbb{N}$ and the backward incidence function $B : P \times T \rightarrow \mathbb{N}$ respectively. The marking of a Petri net is defined as function $\mathcal{M} : \mathcal{P} \rightarrow \mathbb{N}$: tokens are associated with places. The evolution of tokens within the net follows transition firing rules. Petri nets allow sequencing, parallelism and synchronization to be easily represented. An *interpreted Petri net* is such that conditions and events are associated with transitions.

ProCoSA [BGVBT06] is a software environment meant for controlling and monitoring highly autonomous systems. System autonomy is usually obtained by putting

together various functions, among which: data analysis (sensor data, monitoring data, operator's inputs), nominal mission monitoring and control (vehicle and payload control actions), decision (management of disruptive events, replanning). These functions, which are often developed as separate subsystems, have to co-operate in order to fulfil the autonomous system behaviour requirements for the specified missions. More precisely, the needs are the following:

- off-line tasks: specification of the co-operation procedures between subsystem software; subsystem coding for embedded operation;
- on-line tasks: procedure monitoring, event monitoring, and management of the dialog with the operator.

ProCoSA includes the following components:

- EdiPet, a graphical interface for Petri nets which is used both by the developer for procedure design and by the operator for execution monitoring;
- JdP, the Petri net player, that executes the procedures, fires the event-triggered transitions of the Petri nets and synchronises the activation of the associated sub-system functions; a socket-based communication protocol allows data to be exchanged with external subsystem software;
- Tiny, a Lisp interpreter dedicated to distributed embedded applications.

The Petri nets used by ProCoSA are interpreted Petri nets: triggering events such as activation or event generation requests are attached to the transitions. Timers can be programmed: a special activation request enables a timer variable to be instantiated, which allows actions with a limited duration to be modelled.
The ProCoSA procedures are used to model the desired behaviours of the autonomous system; the hierarchical modelling features offered by ProCoSA enable to structure the whole application in a generic way: at the highest description level, generic behaviours can be described, regardless of the characteristics of a given vehicle; at the lowest level, they specify the sequences of elementary actions to be performed by the vehicle or the payloads; this modular approach enables a quick adaptation to system changes (e.g. taking into account a new payload).
An important feature of ProCoSA lies in the fact that there is no code translation step between the Petri net procedures and their execution: they are directly interpreted by the Petri net player, thus avoiding any supplementary error causes.
ProCoSA finally includes a verification tool, which makes use of the Petri net analysis techniques to check that some "good" properties are satisfied by the procedures, both at the single procedure level and at the whole project level (that is to say taking into account inter-net connections); the following properties are checked: place safety (not more than one token per Petri net place), detection of dead markings (deadlocks), detection of cyclic firing sequences (loops).

References

[BGVBT06] M. Barbier, J.-F. Gabard, D. Vizcaino, and O. Bonnet-Torrès. ProCoSA: a software package for autonomous system supervision. In *CAR'06 - First Workshop on Control Architectures of Robots*. Montpellier, FR, 2006.

[But01] M. Butler. *Introductory notes on specification with Z.* University of Southampton, GB, 2001. `http://www.ecs.soton.ac.uk/m̃jb/EL208/znotes.pdf`.

[CGL+06] Ch. Castel, J.-F. Gabard, B. Laborde, R. Soumagne, and C. Tessier. FDIR strategies for autonomous satellite formations - A preliminary report. In *AAAI 2006 Fall Symposium "Spacecraft Autonomy: Using AI to Expand Human Space Exploration"*. Washington DC, USA, October 2006.

[Cra] Univ. Cranfield. Satellite formation flying for an interferometry mission . `http://www.cranfield.ac.uk/soe/space/flying.htm`.

[DA05] R. David and H. Alla. *Discrete, continuous, and hybrid Petri nets*, 2005.

[HH99] M. Heiner and M. Heisel. Modeling safety-critical systems with z and petri nets. In *Proceedings of the 18 th International Conference on Computer Computer Safety, Reliability and Security*, volume 1698 of *Lecture Notes In Computer Science*, pages 361 – 374. Springer, 1999.

[Nas] Nasa. Technology - formation flying . `http://planetquest.jpl.nasa.gov/technology/formation_flying.cfm`.

[NAS05] NASA. Glossary - NASA Crew Exploration Vehicle, SOL NNT05AA01J, Attachment J-6. `http://www.spaceref.com/news/viewsr.html?pid=15201`, 2005.

[PJ03] Fr. Peschanski and D. Julien. When concurrent control meets functional requirements, or z +. In *Third International Conference of B and Z Users*, volume 2651 of *Lecture Notes In Computer Science*, pages 79–97. Springer, 2003.

[Sim07] Simbol-X - Le vol en formation pour un tlescope X de nouvelle gnration. `http://www.cnes.fr/web/5848-simbol-x.php`, 2007.

[Spi00] M. Spivey. *The Fuzz manual*. Oxford, GB, 2000. `http://spivey.oriel.ox.ac.uk/mike/fuzz/fuzzman.pdf`.

[Xud01] H. Xudong. PZ nets – a formal method integrating Petri nets with Z. In *Information and Software Technology*, volume 43 Issue 1, pages 1 – 18. Elsevier Science, 2001.