Journées FAC'2012
Formalisation des Activités Concurrentes
4 et 5 avril 2012, IRIT UPS

Nets-within-nets to model innovative space system architectures

Frédéric Cristini and Catherine Tessier
{frederic.cristini,catherine.tessier}@onera.fr
ONERA–DCSD, UR-CD, Toulouse

Abstract *This paper focuses on a nets-within-nets approach to model, simulate and evaluate innovative space system architectures subjected to a wide spectrum of emerging space threats. We will consider networked constellations of autonomous microsatellites - or Autonomous Networked Constellations (ANCs) - in which several heterogeneous interacting entities (satellites and ground stations), rely on resources (functions or sub-systems) to achieve an Earth observation mission. Petri nets are well adapted for modeling ANCs as they feature event-triggered state changes and concurrent communication accesses. Moreover nets-within-nets allow hierarchical state propagation to be represented. The ANC model is based on* Renew 2.2 *which we use to deal with top-down, bottom-up and horizontal synchronizations so as to represent state propagations from an entity to its nested resources and vice-versa, and from an entity to another one. A model and simulation of a simple ANC subjected to threats are given and discussed.*

A. INTRODUCTION

Traditional Intelligence, Surveillance and Reconnaissance (ISR) space systems consist of a limited number of very expensive monolithic satellites, performing optical, infrared or radar observations, or electronic and communications intelligence. Satellite tasking is performed by dedicated Users Ground Centers (UGC), also known as mission centers. On a periodical basis, UGCs send their requests to a Command and Control Center (CCC) which computes mission plans. Those plans are then uploaded to the satellites thanks to a network of Tracking, Telemetry and Command (TT&C) ground stations. This network is also used by the CCC to monitor the state of the satellites. Once raw data have been gathered by the satellites, they are downloaded back to the UGC through specific Receiving stations.

ISR space systems are considered as particularly sensitive and require high availability and robustness. In order to meet those requirements despite the natu-ral space environment hazards and the limited recovery options, the traditional design approach consists in implementing redundant equipments in each monolithic satellite, along with local Fault Detection, Isolation and Recovery (FDIR) algorithms. This leads to complex integration, validation and verification processes and thus to very expensive satellites.

Despite those efforts, ISR satellites remain vulnerable to an emerging class of highly unpredictable threats: a wide range of *space negation capabilities*, including cyber-attacks and directed energy[1] or kinetic energy[2] weapons, are being developed throughout the world [9, p.149-160]. In the meantime, collision risks caused by orbital debris become a growing concern for operational satellites [9, p.27-43]. As traditional space systems are not designed to resist such threats, their chances to be damaged or destroyed are thus increasing.

In order to deal with this new paradigm, we propose to apply the traditional strategy (redundancies + FDIR) to a higher design level, in the context of fractionated spacecraft [2]. This leads us to define two complementary concepts to improve the robustness of space systems: *passive* and *active robustness*.

Passive robustness focuses on the physical architecture of the space segment of the system, *i.e.* the types, number and orbits of satellites. Its objective is to minimize immediate consequences of aggressions. Inspired by the fractionation concept [2], passive robustness is based on networked constellations of redundant small satellites, called "*networked constellations*" (NCs) [7]. In NCs, the military reconnaissance mission is achieved by a constellation of "payload (P/L) satellites" on very low Earth orbits (VLEO). This constellation is supported by another constellation of "support satellites" on higher

[1] lasers or micro-waves
[2] *i.e.* antisatellite missiles

orbits, enabling P/L satellites operations (communication relays, data handling, computation functions...) P/L and support constellations are two layers of a dynamic space network. Each layer behaves like a distributed virtual sub-system and the connections between P/L and supports, performed by intersatellite links (ISLs), create dynamic virtual satellites (Fig.1).

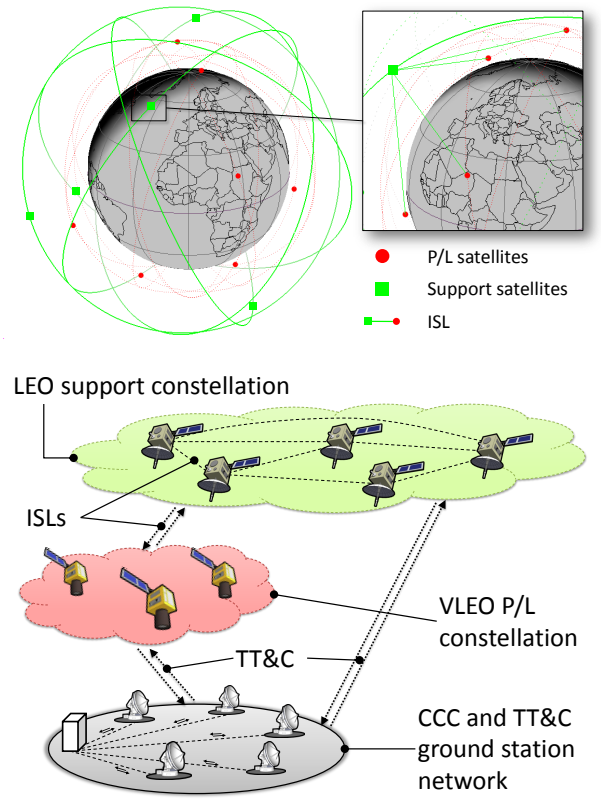

Fig. 1 – *Example of NC configuration: orbital (top) and functional (bottom) configuration.*

Active robustness aims at increasing the system availability and efficiency in the aftermath of an aggression. This is achieved thanks to short-term recovery capabilities accomplished by autonomous onboard algorithms, ground control, or a combination of both. The detection of off-nominal conditions, the isolation of the failure to a specific subsystem and the recovery of nominal or degraded capabilities are performed thanks to FDIR strategies [4, 14].

NC configurations combined to some FDIR strategies have been modeled and simulated thanks to a set of classical computer simulations, using ephemerids generated with *Satellite Tool Kit 8* ®, the space mission simulator developed by *AGI*, and *Excel* ® and *MATLAB* ® computations. The operational performance of the NCs (revisit rate, accessibility...), their communicability (network density, resource sharing conflicts...) and their robustness (operational consequences of satellite failures) have been evaluated and compared [5, 6, 7]. Network metrics [1], such as routing delays or traffic overload caused by FDIR information exchange, have also been assessed. The preliminary results show that threat-tolerance of sensitive space systems can be drastically improved thanks to NC architectures and autonomous FDIR strategies. However they also reveal plenty of concurrency issues between the satellites within NCs, that have to be modeled and assessed. Moreover, in the simulations described above, we only considered *satellite losses* to evaluate the degradation of operational performance of nominal systems. Even if this rough approach is sufficient for a preliminary analysis of threat tolerance of NCs with a large number of satellites, a more accurate description of consequences of aggressions is required, based on satellite functions or equipment losses, to design redundancy architectures and FDIR strategies.

In the remainder of the paper, networked constellations equipped with on-board autonomy for mission achievement and FDIR will be called Autonomous Networked Constellations (ANCs). An ANC will be considered as a *hierarchical and cooperative multiagent system*, including *entity-agents* (ground stations, P/L satellites, support satellites) and *resource-agents* (communication, AOCS[3] or payload equipments...)

The ultimate purpose of this study is to make a trade-off between several ANC hardware (number, types and distribution of entities...) as well software (autonomy organization pattern, FDIR strategies...) architectures. To this end, we must be able to simulate and evaluate nominal and degraded performance of various ANC configurations in order to rank ANCs' architectures, which requires to:

- easily create, handle and study several ANC hardware and software configurations, which are large networked space systems with heterogeneous entities;
- manage *transient communication sessions* between these entities, which are ruled by space dynamics laws; interfacing with realistic precomputed epheme-rids is strongly considered;
- study *post-aggression failure propagation* within ANCs, from resource-agents to entity-agents;
- assess *several reconfiguration strategies*, which imply upward and downward interactions between entities and resources.

This paper is organized as follows: in section B., we will detail the hierarchical multiagent description of ANCs. In section C., we will describe the specific nets-within-nets approach chosen for our work, namely *Reference nets*, as well as the tool, *Renew*

[3] Attitude and Orbit Control System

2.2. Finally in section D., we will show a preliminary example of an ANC configuration modeled as a set of *Reference nets* with *Renew 2.2.*

B. ANC MODEL FEATURES

In this section, we describe the detailed features of an ANC and specify what is needed for its modeling.

1. ANC model structure

The static structure of an ANC model should include the following levels (Fig.2):

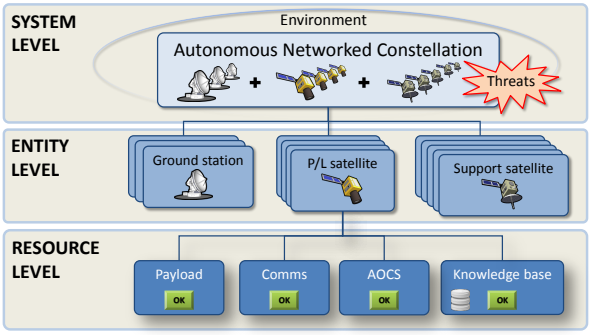

Fig. 2 – *Multilevel breaking down of the ANC model structure.*

"System" level. It is the highest level of the ANC model. It represents the ANC in its environment and contains a reference to each satellite and ground station of the ANC. In this paper, we only consider a limited model of the environment of ANCs: the Earth, which provides its gravity field and enables orbital moves, the Earth surface, which is observed by the P/L satellites and on which ground stations are set up, and the near-Earth space where satellites orbit. Threats, that can strike any part of an ANC, are also part of the environment.

"Entity" level. This level represents the state of each mobile (P/L or support satellites) or fixed entity (ground stations), within the "system". Each entity performs a part of the global ISR mission (data collection or storage, communication, mission planning, system monitoring...) and requires low-level resources to achieve those goals. The number of P/L satellites, support satellites and ground stations is set for each considered ANC configuration [5].

"Resource" level. This level describes the availability of low-level physical or logical components which are considered as resources for "entities": payload, communication, AOCS...It only comprises the relevant functions to be conside-red for threat-tolerance analysis, *i.e.* the most likely to be targeted by aggression means. The "knowledge" of each entity is also stored in a resource called "knowledge base". This knowledge may be about its current state, its environment or the state of other entities.

2. ANC model dynamics

The dynamic features of the different levels are as follows:

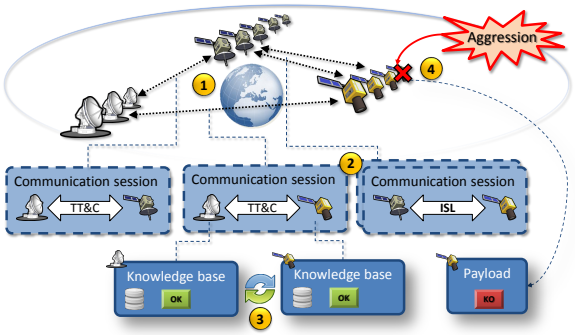

Fig. 3 – *Examples of interactions in an ANC.*
①: inter-entity accesses ruled by the space dynamics laws.
②: ongoing communication sessions (TT&C and ISL).
③: synchronization of two knowledge bases thanks to a communication session.
④: aggression on a P/L satellite - its payload becomes unavailable.

Environment. As it is ruled by the space dynamics laws, the ANC environment may be considered as deterministic: access periods between entities are foreseeable and can be computed thanks to ephemerids that sample position and velocity of each entity over time (Fig.3.①). As for threats, their activity and their precise effects remain highly unpredictable. Any aggression may, however, result in either damages at the resource-level and thus modify the availability of the entity functions (Fig.3.④), or destroy an entire entity.

Entity and resource-agents. As described in section 1., an heterogeneous set of agents is considered in our model: three types of entity-level agents, also called *entities*, (ground stations, P/L and support satellites) and several types of resource-level agents, also called *resources*.

Although those agents are cooperative to achieve the ISR mission, concurrency may emerge, especially between entities (P/L satellites *vs* support satellites, or satellites *vs* ground stations). For example, a ground station may have a simultaneous access with two satellites. For this concurrent situation, either the station has two available antennas and can communicate with both satellites at the same time, or it has to "choose" between them, according to rules (order of priority, first-come/ first-served...) or routing tables. It is noticeable that there is no concurrency between entities to use re-

sources: we assume that entities, which are mainly satellites, are correctly designed and have enough resources to achieve their part of the mission in nominal conditions. In case of an aggression, those resources may however be affected and failures may propagate within the whole system: for instance, the jamming of the communication sub-system (a resource) of a satellite (an entity) may impair the whole system communication pattern. Among the other resources, one should note the specific role of the "knowledge base": it is used to store the current state of each entity with regard to their other resources, as well as the state of the other entities. This knowledge is updated through communication sessions between entities (Fig.3.③).

Communications. By definition, ANCs form dynamic (but deterministic) mesh networks and each entity is a node of this network [5]. Like in any low Earth orbit space system, communications between the ANC entities are necessarily transitory. Those connections are established either by ISLs for communications between satellites, or by TT&C links for communications between satellites and ground stations (Fig.3.②).

Entities can interact in a deterministic and periodical manner [5]. In nominal situations[4], communication periods are determined by space dynamics as well as communication rules and protocols: in order to exchange messages, two satellites must firstly be in mutual visibility, which is determined by their orbits, and secondly be planned to communicate. As we will only focus on those access/no-access periods, we will consider ANCs as discrete-event dynamic systems, without taking into account continuous-time dynamics.

Interactions between entities mainly consist in message passing (concerning the entity current state, operational information, new assignments, communication relays...) Those messages may be used to update each entity's knowledge base, to assign tasks to entities or to broadcast reconfiguration orders.

3. Requirements for a modeling tool

As shown in the previous two paragraphs an ANC model includes several types of entities with embedded resources and subjected to different kinds of events coming from themselves or from the environment. Moreover different ANC configurations including several tens of entities must be modeled easily so that their performance and robustness to aggressions should be assessed and compared.

Therefore a modeling tool for ANCs should have the following features:

- allow easy entity creation or removal (*i.e.* when a satellite is destroyed by an aggression);
- allow a compact representation of the state of an ANC, given the states of the entities, resources and communications;
- represent communicating objects;
- represent event-driven state changes (begin/end of visibility; begin/end of communication; aggressions and failures; reconfiguration orders...)
- represent synchronization and concurrency;
- allow a hierarchical representation of objects (*i.e.* entity-embedded resources) with state propagation within the hierarchy:
  - horizontal state propagation: entity $\longrightarrow$ entity;
  - vertical state propagation: top-down (entity $\longrightarrow$ its resource(s)) and bottom-up (resource $\longrightarrow$ its entity).

These requirements made us choose the "nets-within-nets" Petri net extension which will be reminded in the next section. By the way although Petri nets have been widely used to study complex systems [8] as well as multiagent and multirobot systems [3, 16], very few references are available concerning space system design [14].

C. FROM NETS-WITHIN-NETS TO REFERENCE NETS

Before modeling and simulating an ANC as a set of nets-within-nets, we have to determine the precise nets-within-nets formalism as well as the software we will use.

1. Reference Nets and *Renew 2.2*

Over the past decade several "nets-within-nets" approaches have been introduced to model hierarchical multiagent distributed systems: Elementary Object Petri Nets [17], Nested Petri Nets [15], Reference Nets [11] or Mobile Object Net Systems [10].

Each formalism features different inter-net connection schemes: indeed the number of hierarchical levels may be limited (*e.g.* two levels [17]), the ability to communicate or synchronize transition firings may be restricted in order to preserve some formal properties (*e.g. decidability* [15]), or specific semantics may be considered (*e.g. value semantics* [10]). Those limitations or specific features may appear problematic to model ANCs. Moreover there are few available software tools to design and run such "nets-within-nets".

[4] *i.e.* with no failures and no aggressions.

As for Reference nets, they implement the nets-within-nets concept thanks to a special inscription language using Java expressions, which may control transition enabling and can also create new instances of subnets. A given net instance can communicate with another one provided it has a reference of this instance and that those nets have linked transitions thanks to so-called "synchronous channels" inscriptions.

Those features have been implemented in an academic open source Petri net simulator, called *Renew*[5], whose current version, *Renew 2.2*, was released in August 2009 [12, 13]. The developer team still provides support.

2. Some *Renew 2.2* specific features

Net instances. *Renew 2.2* differentiates typical static nets that are drawn in the graphical editor and considered as templates, and net instances that are dynamically created from net templates during the simulation [12, p.43].

As presented previously (section 1.), ANCs are composed of several "copies" of a limited number of different entities (and resources): P/L satellites, support satellites, ground stations... Therefore each type of entity will be modeled as a specific Petri net template, and their instances will represent the different P/L satellites, support satellites or ground stations within the considered ANC configuration.

Transitory communication links (section 2.) may also be considered as transitory Petri nets: thanks to *Renew 2.2* features, an unlimited number of communication nets can be instantiated "on the fly".

Finally, the hierarchical structure of ANCs is immediately obtained when instantiating nets within other nets.

However instances cannot be erased or removed from the simulation: if all transitions of an instance are disabled, this instance is "forgotten" by the simulator, thanks to a garbage collector. Consequently we have to design nets very carefully in order to be sure that unused instances will not remain active and disturb the simulation run.

Synchronization schemes. *Synchronous channels* are one of the specific features of reference nets: they enable nets to influence each other [12, p.43-47]. This feature is mandatory to simulate communicating entities or event-driven net evolutions, or to synchronize net activities. Synchronizations will be extensively used to model ANCs.

Within the hierarchical architecture of ANCs, we have to consider top-down, bottom-up and horizontal synchronizations. Even if they are made possible by *Renew*, they require different types of techniques:

- *Top-down synchronization* (Fig.4) is the simplest one as the master "knows" the slave-instance it creates;

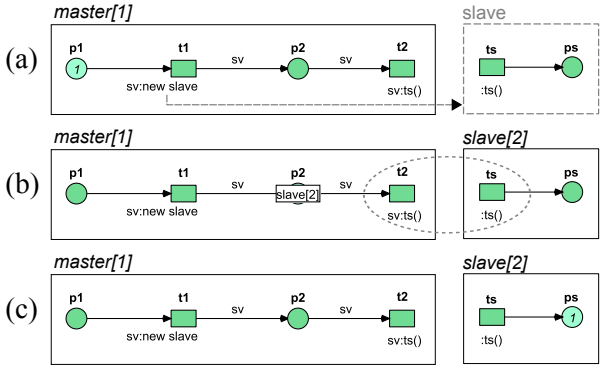

Fig. 4 – *Top-down synchronization.*
(a): Thanks to the inscription sv:new slave*, firing* t1 *creates one instance of the* slave *net (*slave[2]*). A reference to* slave[2] *is stored in variable* sv*.*
(b): t2 *and* ts *are synchronized thanks to inscriptions* sv:ts() *and* :ts()*. Note the net-token* slave[2] *in* p2*.*
(c): Final marking.

- *Bottom-up synchronization* (Fig.5): the slave net must "know" its master: a reference to the master must be passed to the slave instance;

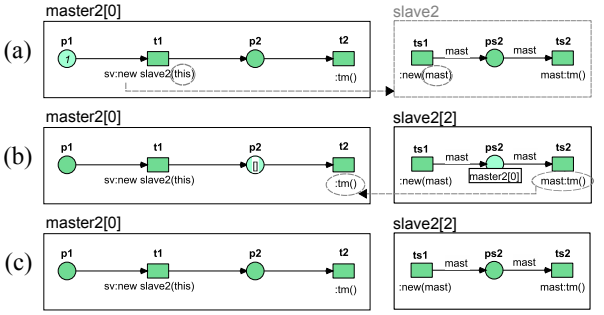

Fig. 5 – *Bottom-up synchronization.*
(a): Thanks to inscription sv:new slave2(this)*, firing* t1 *creates one instance of* slave2 *net (*slave2[2]*) and passes a reference to* master2[0] *to* slave2[2] *thanks to keyword* this*.* slave2[2] *stores this reference in variable* mast*.*
(b): t2 *and* ts2 *are synchronized thanks to inscriptions* :tm() *and* mast:tm()*. Note the net-token* master2[0] *in* ps2*.*
(c): Final marking.

- *Horizontal synchronization* (Fig.6) is quite similar to the bottom-up synchronization; each slave must "know" the other one.

[5] for REference NEts Workshop

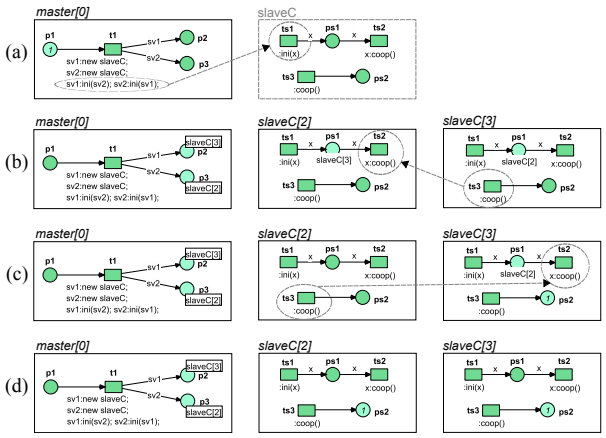

Fig. 6 – *Horizontal synchronization between two slave-nets.*
(a): Thanks to inscriptions sv1:ini(sv2) *and* sv2:ini(sv1)*, each* slaveC *net instance receives a reference to the other* slaveC *net instance (transition* ts1*,* :ini(x)*).*
(b): Transitions slaveC[2].ts2 *and* slaveC[3].ts3 *are synchronized thanks to inscriptions* x:coop() *and* :coop()*.*
(c): Transitions slaveC[3].ts2 *and* slaveC[2].ts3 *are synchronized thanks to inscriptions* x:coop() *and* :coop()*.*
(d): Final marking.

D. APPLICATION

In this section, we present an example of a simplified ANC modeled with *Renew 2.2* as a set of Reference nets.

1. A simplified ANC

The simplified ANC configuration we are considering here is composed of the following entities: 2 P/L satellites, 1 support satellite and 1 TT&C ground station.

Each entity embeds the following resources:

- 1 communication resource + 1 knowledge base;
- for P/L and support satellites: 1 AOCS resource;
- for P/L satellites only: 2 payload resources.

Concerning the dynamic behavior of the ANC, we have made the following simplifying assumptions:

- inter P/L satellite communications are not allowed;
- the support satellite is a sharable entity: it can simultaneously establish ISLs with the two P/L satellites;
- the TT&C ground station is an unsharable entity: it can only establish a link with one satellite at a time;
- communication sessions are ruled by a simple exchange protocol: mutual identification of entities with no acknowledgment;
- neither time nor real space dynamics are implemented in our model yet.

As for threats, they can either aggress any kind of resource, or destroy an entire entity. They are triggered manually.

2. Nets description

Each ANC item described in section B. is modeled as a Petri net.

System net (Fig.7, page 7). It is the highest level net of the ANC and has two main functions: setting up the simulation by creating entity-net instances (static structure of the ANC) and managing the system's dynamics (aggression triggering, ISL and TT&C accesses).

Simulation initialization (Fig.7, page 7, frame 1). The manual transition Init_Simu creates instances of entity nets of the considered ANC configuration thanks to several *creation inscriptions*. For example, creation inscription sat1:new satellitePL(this,1) creates a new instance of the P/L satellite net and passes the parameter list (this,1) to this instance, which is assigned to variable sat1. sat0, sat2 and gst0 are instances of, respectively, satelliteSupp net, satellitePL net and groundStation net. An instance of the net init[6] distributes the references of those instances so that they can interact. Those references are stored in places Satellite_Indices, Satellite_List, Station_Indices and Station_List. Those places are then used to manage accesses via *virtual places* which can be seen at the bottom of the system net in ISL and TT&C accesses management areas (frames 3 and 4), with the V. prefix. When Start_Simu transition is fired, the simulation run starts.

Threat management (Fig.7, page 7, frame 2). This area is used to manually trigger aggressions on resources or entities. The upper part is dedicated to satellite destruction management, which is an irreversible process (*e.g.* attack of an antisatellite missile). As net instances cannot be removed (see section 2.), P/L satellites and support satellites are dealt with differently: the payload activity in satellitePL instances is disabled thanks to inscription listSat[sat]:stopPL().

The lower part is dedicated to resource threat management. It is composed of three frames, respectively for threats on communications (*e.g.* jamming), payload (*e.g.* dazzling) and AOCS (*e.g.* GPS

[6] not detailed here; this net has only a functional role.

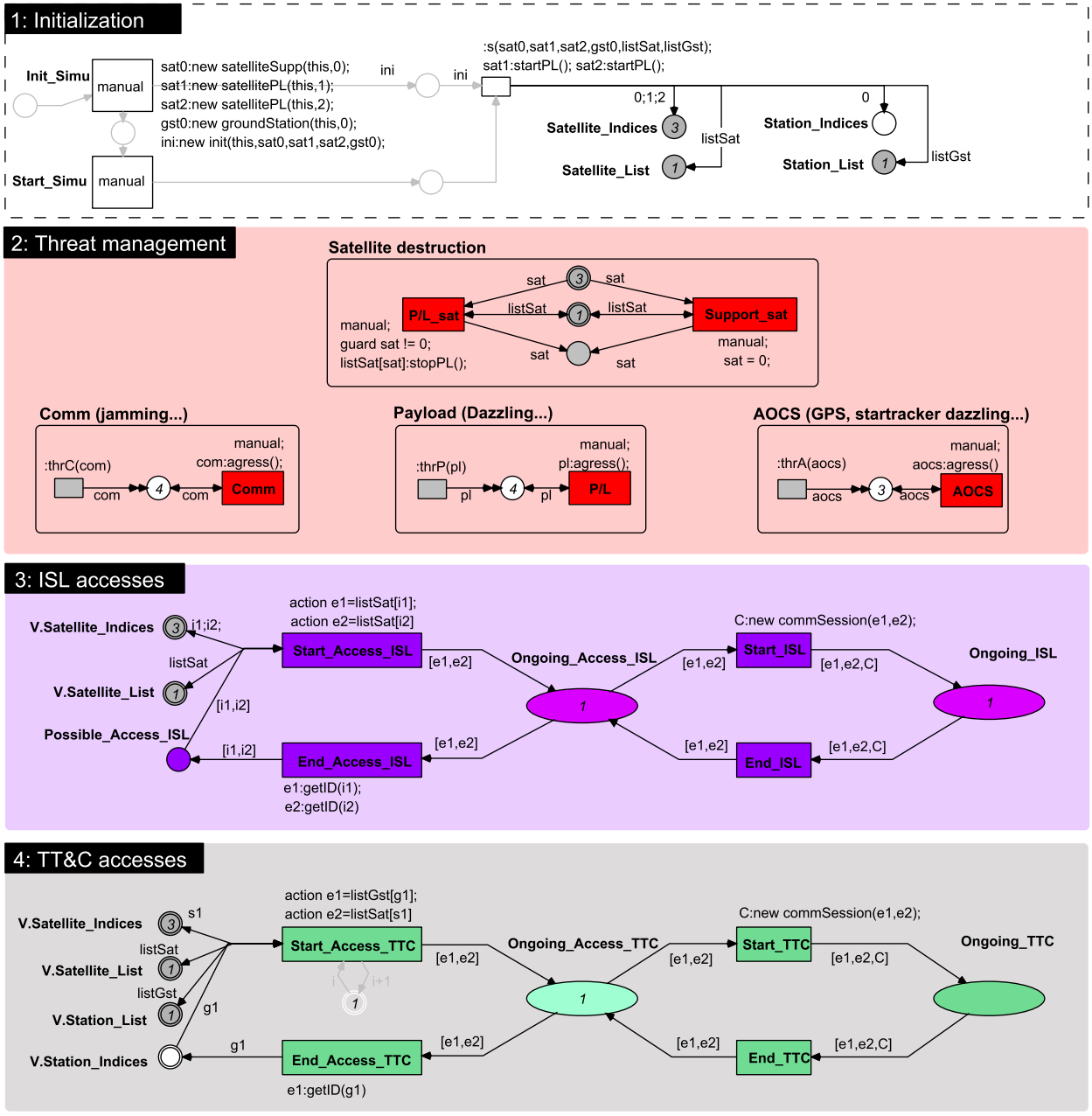

Fig. 7 – *An instance of* system net. *The marking is composed of colored tokens which may be integers (e.g. place* Satellite_Indices*, 3 tokens in the current marking), tuples (e.g. place* Ongoing_ISL*, 1 token), Java objects (e.g. place* Satellite_List*, 1 token) or net references (e.g. places in payload, communications or AOCS threat management, respectively with 4, 4 and 3 tokens). Only the number of tokens is displayed, not their values.*

jamming). Transitions with inscriptions thrC(com), thrP(pl) and thrA(aocs) are fired during the initialization phase. Then a reference to each resource net is stored in the central place of each net. When transition :agress is fired manually, one resource is made unavailable until reconfiguration (manual transition in resource nets).

Access management (Fig.7, page 7, frames 3 and 4). Those nets simulate access and communication periods between entities. As neither time nor real space dynamics are implemented yet, a very simple dynamic model is considered: for ISLs, place Possible_Access_ISL contains a list of possible ISL accesses, determined externally thanks to satellite ephemerids; as for TT&C links, accesses to ground stations are considered as concurrent. When transition Start_Access_ISL fires, an access period between two satellites (e1 and e2) begins (place Ongoing_Access_ISL) and transition Start_ISL is enabled. If it fires, a communication session between e1 and e2 is established (inscription C:new commSession(e1,e2)). Transitions End_Access_ISL and End_ISL put an end to ISL communications and access periods respectively. This description is also suitable for TT&C access management.

Entity nets have a common structure:

1. an *initialization area* that instantiates the suitable number and type of the entity embedded resources and stores their references in variables;
2. a *functional area* that manages the operational activities of the entity, *i.e.* how the resources are used.

The size of the functional area depends on the type of the considered entity. According to section 1., functional areas of support satellite and ground station nets are subsets of the P/L satellite net functional area, which is composed as follows: knowledge base, communication function, AOCS function and payload function (Fig.8, page 10). Support satellite and ground station nets are not detailed here. They have the same structure with less embedded resources: knowledge base, communications and AOCS for support satellite nets, knowledge base and communications for ground station nets.

Knowledge base: it is instantiated thanks to inscription kb:new resourceKB(this) and one token kb is put in place Knowledge_Base. Each function of the entity can update the knowledge base according to its state or its activity thanks to top-down synchronous channels (*e.g.* place Emit with inscription kb:read(msg) on Fig.8). When a communication session is established (inscription :initSession()), the knowledge base is accessed and a message is prepared.

Communication function: the communication resource is instantiated thanks to inscription c01:new resourceCO(this). Token co1 in place Comm_OK sets the resource as "available" for operational use. The right hand part of the communication resource area is dedicated to communication management thanks to transitions Emit and Receive, and to synchronous channels (top-down, with knowledge base: kb:rec(msg) and kb:read(msg); bottom-up, with communication session net: emit(msg) and receive(msg)).

Token co1 can be removed in case of an aggression on the communication resource: a bottom-up synchronous channel (:KO_Comm(), from resource to entity) is activated and the token moves to place Comm_KO. This firing also updates the knowledge base thanks to kb:updateS(0,0). This marking prevents further operational use of the communication function. The resource can be made avai-lable again thanks to channel :recovComm(). The availability of the AOCS and payload functions are managed in the same manner.

Payload function: in the example, it relies on two P/L resources assigned to variables pl1 and pl2 in place PL_OK. Transition Use_PL only requires one P/L resource: this is how we simulate a hot redundancy[7] reconfiguration scheme. The firing of this transition updates the knowledge base thanks to channel kb:updateA(1): the integer value 1 is sent to the knowledge base and increments a counter of operational activity, called *activity index*.

Resource nets (Fig.9). In the considered simplified ANC configuration, communication, payload and AOCS resources have the same net structure. The main places of the nets are OK, which is initially marked with a black token, and KO. This token moves to KO when an aggression occurs (inscription :agress()) and moves back to OK when recovery actions are carried out (*e.g.* cold redundancy activation[8], reset of an electronic component...)

[7] the second resource is immediately available in case the first one becomes unavailable.
[8] the second resource is powered off; it needs to be powered up to become available.

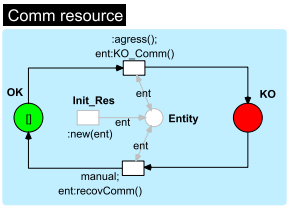

Fig. 9 – *A* communication resource net. *The place Entity stores the reference to the upper entity in order to enable bottom-up synchronous channels ent:KO_Comm() and ent:recovComm().*

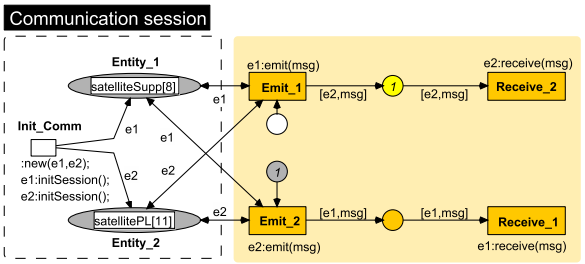

Fig. 11 – *An instance of a* communication session net *between satelliteSupp[8] and satellitePL[11]. One message msg is being sent by satelliteSupp[8].*

Knowledge base net (Fig.10, page 10). Its main role is to store the entity's local know-ledge about the other entities in a set of tuples that contains the states of all the entities of the ANC. It is structured as follows: identification of an entity, state of its resources (stored in a Java list object) and value of *activity index*. This index measures the operational activity of an entity over time: for example, each time a P/L satellite activates its payload, the activity index is increased by 1. It is particularly used during the synchronization phase to compare knowledge freshness between received and stored data.

This net enables interactions with other nets: top-down synchronization with the upper entity to update its activity index (transition Update_Activity) or resource state (transition Update_State) or bottom-up synchronization when a communication session is established between two entities to pass messages (*e.g.* transitions Prepare_Msg or Receive_Msg).

Knowledge is synchronized between entities when they communicate. Emission and reception activities are carried out by the lower part of the net. For emission, the formatted message is only read and sent to the communication function (:read(msg)). For reception, the received message is stored in a buffer (place Input_Buffer) and its data are sorted: the knowledge base only keeps up-to-date data about other resources (transition Write_Msg with guard inscription); out-of-date data or data about its own entity, which is necessarily out-of-date, are cleared (transition Clear).

Communication session net (Fig.11). It is used to model bidirectional communications between two entities. When a communication session starts, each entity receives the identity of the other one. Then, on a rotating basis, each entity sends the content of its knowledge base (transitions Emit_1 and Emit_2).

3. Simulation and first results

This section focuses on first results we have obtained with a scenario involving an aggression on the communication resource of the support satellite. As time is not implemented yet, it is approximated by a discrete counter of ground station accesses, which are regularly spread over time. In the following, one simulation *step* will be equivalent to one ground station access, disregarding other Petri net activities.

Scenario description. The simulation starts with the nominal simplified ANC. After approximately 1000 steps, the communication resource of the support satellite is jammed and permanently made unavailable. Therefore ISL communication sessions no longer enable knowledge base synchronization: P/L satellites keep on emitting messages but the support satellite remains quiet. The knowledge bases of all entities are recorded every 100 steps.

Metrics. Due to transitory accesses, the knowledge bases of the different entities always change: consequently at a given step, there are always differences between the data stored in each knowledge base. Those discrepancies are measured through the *activity indices* that assess the difference between the "real" activity index of each entity, stored and updated within each entity, and the "believed" activity index, stored in the other entities' knowledge bases. Let $A_{i,j}(s)$ be the activity index of satellite j stored in satellite i at step s and $A_{i,i}(s)$ the "real" activity index of satellite i at step s. For the simplified ANC, we have defined two *knowledge discrepancy* factors KD between P/L satellites "1" and "2" and ground station "g":

- $KD_{sat-sat}(s) = \frac{1}{2} \cdot [(A_{1,2}(s) - A_{2,2}(s)) + (A_{2,1}(s) - A_{1,1}(s))]$;
- $KD_{sat-grd}(s) = \frac{1}{2} \cdot [(A_{1,g}(s) - A_{g,g}(s)) + (A_{2,g}(s) - A_{g,g}(s))]$;

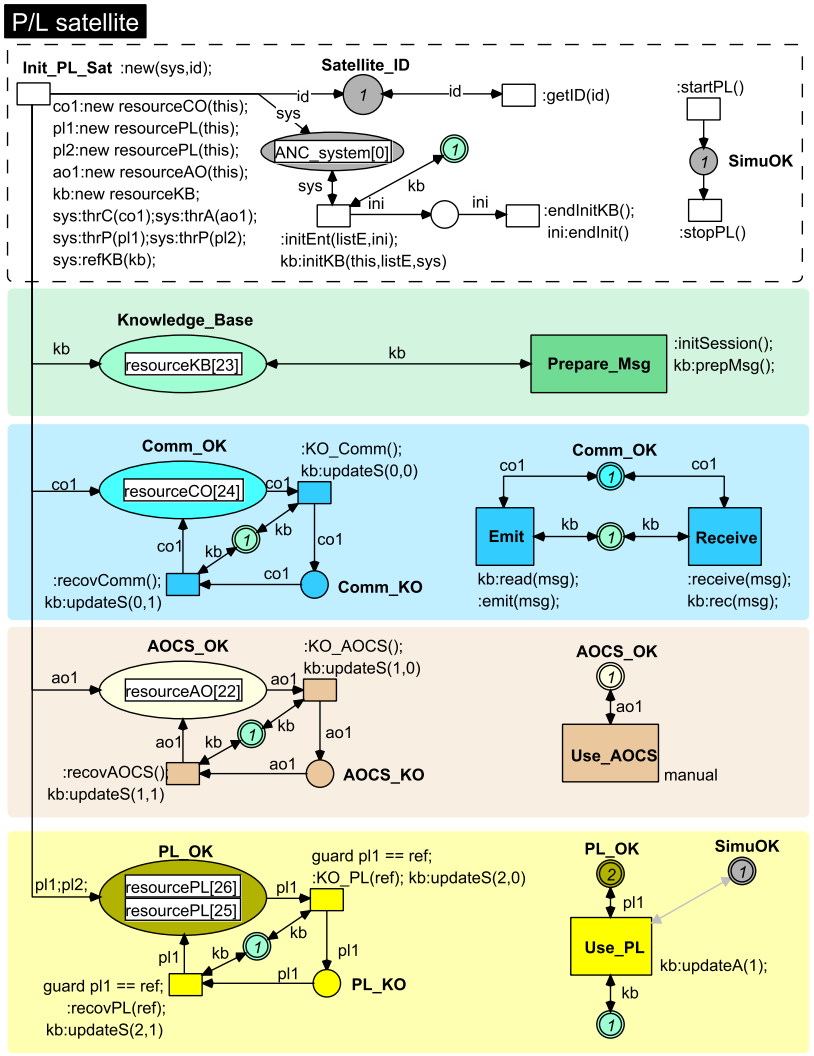

Fig. 8 – *An instance of* P/L satellite net.
At the top: initialization area. Below: functional areas. The current marking means that this instance is the number "1" P/L satellite (place Satellite_ID*) and that it was created by instance* ANC_system[0]*. Functions are provided by resources* resourceKB[23]*,* resourceCO[24]*,* resourceAO[22]*,* resourcePL[25] *and* resourcePL[26]*. They are all available.*

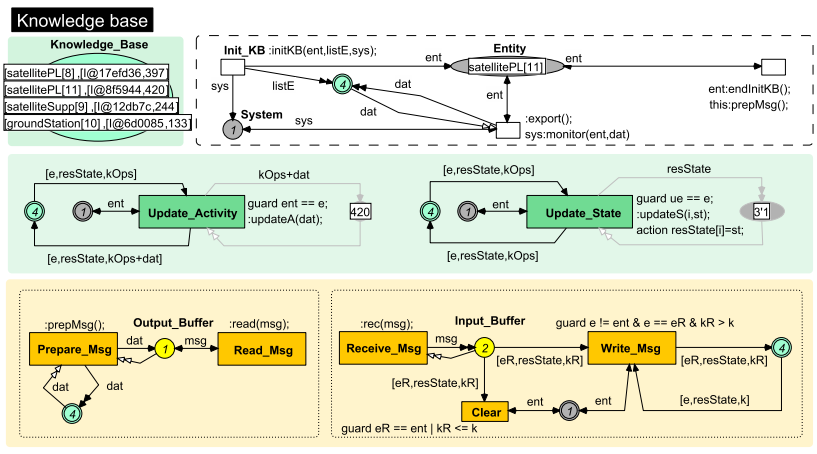

Fig. 10 – *An instance of a* knowledge base net. *This instance was created by* satellite[11] *(place* Entity*). Note the structure of the 4 tokens in place* Knowledge_Base*. The value of the activity index is 420, i.e. the payload of* satellite[11] *has been activated 420 times since the beginning of the simulation. The* Output_Buffer *is ready for the next emission. The* Input_Buffer *is processing a received message (2 tokens left, i.e. two tuples concerning two entities).*

Results. The KD results are displayed in Fig.12 and Tab.1.

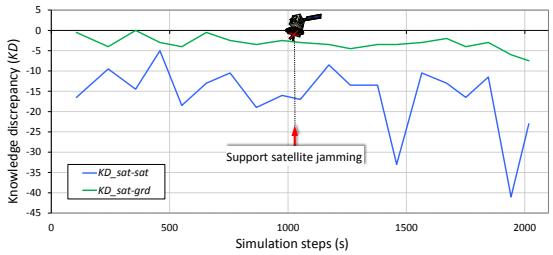

Fig. 12 – *Knowledge discrepancies over time.*

Tab. 1 – *Knowledge discrepancies evolution before and after the aggression. (mean temporal values)*

Knowledge discrepancy	Before aggression	After aggression	Deviation
$A_{1,2} - A_{2,2}$	-11.9	-18.4	-54.6%
$A_{1,g} - A_{g,g}$	-2.1	-4.3	-104.8%
$A_{2,1} - A_{1,1}$	-16	-18.4	-15.0%
$A_{2,g} - A_{g,g}$	-2.6	-3.8	-46.1%
$A_{g,1} - A_{1,1}$	-15.6	-13.1	+16.0%
$A_{g,1} - A_{2,2}$	-8.0	-6.2	+22.5%

One may notice that from the P/L satellites point-of-view, the aggression on the communication resource of the support satellite results in a deterioration of their knowledge: mutual knowledge drops of 35% on average (-54.6% and -15.0%), and knowledge of the ground station activity index drops of 75% (-104.8% and -46.1%). As for the ground station, it benefits from this loss: as it is unsharable as far as communications are concerned, it is more available to communicate with P/L satellites and thus its knowledge about them improves (+16.0% and +22.5%). Those preliminary results are coherent and comply with previous results [5, 6, 7]: communication jamming on support satellites results in a significant degradation of the overall performance of the ANC (here the knowledge discrepancy).

E. CONCLUSION AND FURTHER WORK

The nets-within-nets approach allows us to design and simulate clearly organized models of ANCs that are well adapted for further assessment of the performance and robustness of different ANC configurations subjected to different kinds of threats. Indeed nets-within-nets allow hierarchical influences to be repre-sented and the entity-resource hierarchy that we have shown paves the way for a multiple-level hierarchy with *e.g.* more detailed resources for a more precise assessment of threat impacts. The assessment metrics are directly linked to the semantics of tokens, that carry explicit knowledge of the ANC state.

Moreover reference nets, which are derived from object oriented programing, enable us to envisage quite easy manipulations as the whole structure of the model will not be affected by changes within nets, provided we manage to preserve interfaces with other nets.

Further work will focus on the improvement of the ANC model and on the setting up of more comprehensive simulation scenarios in order to assess several ANC configurations thoroughly, with various combinations of P/L and support satellites:

- Time: the nets need to be modified in order to make them suitable with the timed Petri net formalism which is a feature of *Renew 2.2*. Time will enable us to perform more realistic simulations and thus evaluate ANC performance more accurately;
- Real space dynamics: this will require the development of a plugin for *Renew 2.2* in order to fire some transitions according to real ephemerids read from external files;
- Knowledge bases: more complex data structures will be implemented, including knowledge on the environment or on the mission plan of the other entities;
- Communication protocols: different protocols will be implemented to manage communication session creations;
- Automatic export and processing of simulation traces produced by *Renew 2.2*.

REFERENCES

[1] C. Barnhart and R. Ziemer. Topological analysis of networks composed of multiple satellites. In *Tenth Annual International Phoenix Conference on Computers and Communications*. US Naval Res. Lab., Washington, DC, 1991.

[2] O. Brown and P. Eremenko. Fractionated space architectures: a vision for responsive space. In *4th Responsive Space Conference, Los Angeles, CA, USA*. AIAA, 2006.

[3] J. R. Celaya, A. A. Desrochers, and R. J. Graves. Modeling and analysis of multi-agent systems using Petri nets. *Journal of Computers*, 4(10):981–996, Oct. 2009.

[4] C. Cougnet, B. Gerber, and J.-F. Dufour. New technologies for improving satellite maintainability, flexibility and responsiveness. In *Toulouse Space*

Show 2010, TechnoDis Symposium, Toulouse, France, 2010.

[5] F. Cristini. Robust satellite networks: solutions against emerging space threats. In *18th IFAC Symposium on Automatic Control in Aerospace*, Nara, Japan, 2010.

[6] F. Cristini, C. Tessier, and E. Bensana. Satellite network architectures against emerging space menaces. In *Toulouse Space Show 2010, TechnoDis Symposium*, Toulouse, France, 2010.

[7] F. Cristini, C. Tessier, and E. Bensana. Satellite network architectures against emerging space threats. In *10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, Sapporo, Japan, 2010.

[8] C. Girault and R. Valk. *Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[9] C. Jaramillo, editor. *Space Security Index 2011, Executive summary*. Project Ploughshares and the McGill University Institute of Air and Space Law, Canada, 2011.

[10] M. Köhler. Mobile object net systems: Petri nets as active tokens. Technical Report 320, University of Hamburg, Department of Computer Science, 2002.

[11] O. Kummer. Introduction to Petri Nets and Reference Nets. *Sozionik Aktuell*, 1:1–9, 2001.

[12] O. Kummer, F. Wienberg, M. Duvigneau, and L. Cabac. Renew - User Guide. Release 2.2. Technical report, Theoretical Foundations Group, Departement of Informations, University of Hamburg, 2009.

[13] O. Kummer, F. Wienberg, M. Duvigneau, J. Schumacher, M. Köhler, D. Moldt, H. Rölke, and R. Valk. An extensible editor and simulation engine for Petri nets: Renew. In J. Cortadella and W. Reisig, editors, *Applications and Theory of Petri Nets 2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 484–493. Springer Berlin / Heidelberg, 2004.

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

[15] I. A. Lomazova. Nested Petri Nets: a formalism for specification and verification of multi-agent distributed systems. *Fundamenta Informaticae*, 43:195–214, 2000.

[16] R. Sánchez-Herrera, N. Villanueva-Paredes, and E. López-Mellado. High-Level Modelling of Cooperative Mobile Robot Systems. In R. Alami, R. Chatila, and H. Asama, editors, *Distributed Autonomous Robotic Systems 6*, pages 431–440. Springer Japan, 2007.

[17] R. Valk. Petri nets as token objects - an introduction to elementary object nets. In J. Desel and M. Silva, editors, *19th International Conference on Application and Theory of Petri nets, Lisbon, Portugal*, number 1420 in LNCS, pages 1–25. Springer, 1998.