Design and Analysis of Semaphore Precedence Constraints: a Model-based Approach for Deterministic Communications

Thanh-Dat Nguyen*, Yassine Ouhammou*, Emmanuel Grolleau*, Julien Forget†, Claire Pagetti‡ and Pascal Richard*
*LIAS, ENSMA and University of Poitiers, Futuroscope, France
Email: (thanh-dat.nguyen, yassine.ouhammou, grolleau)@ensma.fr and pascal.richard@univ-poitiers.fr
†LIFL, University of Lille1, Villeneuve d'Ascq, France - Email: julien.forget@lifl.fr
‡ONERA/DTIM, Toulouse, France - Email: claire.pagetti@onera.fr

***Abstract*—Architecture Analysis and Design Language (AADL) is a standard in avionics system design. However, the communication patterns provided by AADL are not sufficient to the current context of Real-Time Embedded System (RTES) in which some multi-periodic communication patterns may occur. We propose an extension of a precedence model between tasks of different periods (multiperiodic communication). This relies on the *Semaphore Precedence Constraint (SPC)* model that is inspired from the concept of *Semaphore*, and more specifically on the $m-n$ producer/consumer paradigm. We reinforce the SPC semantics by allowing cycles in the SPC precedence graph. We also present another viewpoint on the periodicity of tasks system using SPC based on a graph apart from the encoding technique presented in the SPC seminal work. An implementation of SPC in AADL and its associated analysis tool are also provided to study the temporal behaviour of systems using SPC.**

I. Introduction

An embedded system is a set of hardware and software components cooperating to accomplish a specific mission. When a system is subject to temporal constraints, it is considered real-time. Real-Time Embedded System (RTES) are used in many critical domains of our life such as telecommunication, automotive, avionic or nuclear.

RTESs often require communications between tasks. Many patterns for inter-task communications have been proposed by real-time operating systems, such as asynchronous patterns (blackboard or data module) and (loosely[1]) synchronous patterns (Mailbox, message queues, etc.). For synchronous communications, the common underlying model is the producer/consumer paradigm with or without losses. This model allows the $1-1$ communication patterns (i.e., one consumer execution for each producer execution), but it also allows the $m-n$ communication patterns (i.e., n consumer executions for m producer executions).

A. Problem statement

A system is *functionally deterministic* when its outputs are always the same for the same given inputs and *temporally deterministic* when its outputs are always produced within bounded time intervals. Besides, the temporal determinism ensures the repetition of tasks communication. Therefore, the behaviour of the system can be reliably predicted over time thanks to the schedulability analyses that are based on models capturing the temporal behaviour including the tasks communication. Nevertheless, most of the old deterministic communication models (e.g., *Simple Precedence Constraint*[1][2]) only supports $1-1$ communication patterns. Hence, they do not meet the current industrial needs (e.g., the example treated in the case study). Some of the existing work deals with multi-periodic communication (e.g., *Generalized Precedence Constraint* [3], *Repetitive Precedence Constraint*[4]) but are either less expressive or too complicated and lack a complete analysis framework. In a different context (i.e., throughput maximization in cyclic scheduling), *linear* precedence constraints have been proposed to model an infinite set of simple precedence constraints among jobs as a multi-graph [5]. A very powerful affine model of multi-periodic precedence constraints is proposed in [6], but it allows only analysis for specific offline schedulers.

[1]Synchronous communication by rendez-vous is usually avoided, therefore in the sequel, synchronous communication is considered loose.

B. Paper contribution

In this paper, we propose an extension of the multi-periodic precedence model [3], [7] that allows a precedence graph with cycles. We also present an implementation of this extended model in the architecture design language AADL [8], for which we have realized an extension to define these constraints, as well as a schedulability analysis tool.

C. Paper outline

The rest of the paper is organized as follows. Section II introduces different communication models related to the task precedence relationship and AADL language and discusses their limits. Section III presents the two contributions of the paper in terms of modeling and analysis of systems with task precedence relationships. Section IV illustrates the results obtained through a case study based on an industrial example. Finally, Section V concludes this paper.

978-3-9819263-0-9/DATE18/©2018 EDAA

II. Background

A. Deterministic communication patterns and representation models

Several communication patterns have been proposed where tasks have to be executed in a specific order. Figure 1 presents some examples of deterministic multi-rate (i.e., multiperiodic) communication patterns related to the precedence relationship between two tasks (τ_i and τ_j).

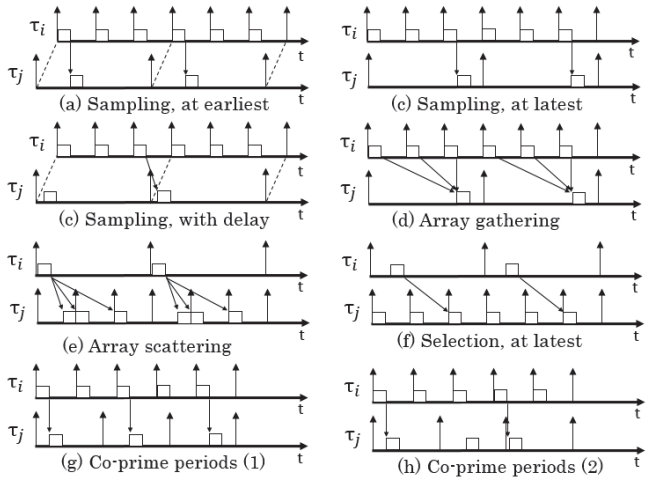

Fig. 1. Multi-rate communication patterns

Without (functional and temporal) determinism, the communication can never be validated, due to an unexpected functional behaviour, an unbounded complexity or an implicit temporal representation model. Several works have been proposed to make the communication deterministic. We detail some precedence models that have tried to express the communication patterns as follows (Figure 2 shows the precedence models presented in the sequel).

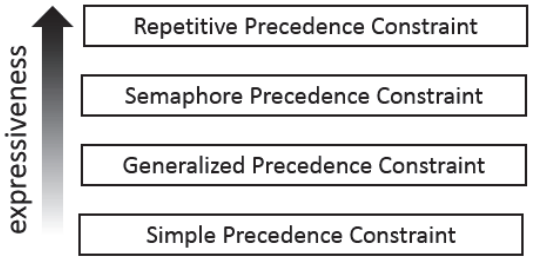

Fig. 2. Expressiveness of precedence models

Simple Precedence Constraint [1][2] was the first effort that tackled the problem of deterministic tasks communication. The notation of *Simple Precedence Constraint*: $\tau_i \rightarrow \tau_j$ has been firstly applied for jobs to express the job τ_i precedes the job τ_j and then extended to express precedence relations between two tasks of the same period where each job $\tau_{i.k}$ of task τ_i precedes the corresponding job $\tau_{j.k}$ of task τ_j. Two tasks involved in a *Simple Precedence Constraint* have to share the same period.

Generalised Precedence Constraint GPC is presented in [3]. GPC is an extension of *Simple Precedence Constraint* to support multiperiodic communication where the period of a task is a multiple of the period of the other. Authors of [3] focus on scheduling periodic tasks related by generalized precedence constraints given in an acyclic graph. The Generalized Precedence Constraint $\tau_i \rightarrow \tau_j$ requires that the number of started jobs S_j(t) of τ_j at any time t and the number of ended jobs E_i(t) of τ_i are such that:

$$E_i(t) \times T_i \geq S_j(t) \times T_j \quad (1)$$

where T_i (resp. T_j) denotes the period of task τ_i (resp. τ_j).

Repetitive Precedence Constraint RPC model [4] specifies a precedence constraint between two tasks τ_i and τ_j by using a predicate $Code_{ij}(n, m)$, which is true if and only if the job n of task τ_i precedes the job m of task τ_j (for n $\in [1..N]$ and m $\in [1..M]$, where lcm stands for least common multiple, N=$\mathrm{lcm}(T_i, T_j)/T_i$ and M=$\mathrm{lcm}(T_i, T_j)/T_j$). Therefore, the representation of the precedence relations uses a non-polynomial space. Moreover, no restriction is imposed on the precedence predicate, which may lead to inconsistent or redundant precedence constraints.

Semaphore Precedence Constraint (SPC) Semaphore Precedence Constraint [7] is inspired from the concept of Semaphore, and more specifically on the $m - n$ producer/consumer paradigm. Concretely, with SPC (denoted by $\tau_i \overset{h}{\rightarrow} \tau_j$), the behaviour of tasks depends on a counting semaphore whose initial value is denoted by h (with $h \in \mathbb{N}$ in [7], and extended to $h \in \mathbb{Z}$ in this paper). The source task of the constraint plays the role of the producer which adds a value equal to its period to the semaphore count at each execution. The target task of the constraint plays the role of the consumer taking a value equal to its period from the semaphore count at each execution. The key idea is that the consumer cannot take more value than the current semaphore count. Therefore, when the consumer tries to take more value than the semaphore count, it is blocked. Note that no cycle is allowed in the SPC precedence graph in the original work of [7].

B. AADL in a nutshell

Architecture **A**nalysis and **D**esign **L**anguage[2] is a domain specific language dedicated to design software and hardware architectures for real-time embedded systems. AADL provides components helping to define hardware (such as processor, bus and memory) and software concerns (such as thread, process and data). Interactions between components are expressed through their interfaces (i.e., ports, bus access, etc.). In addition, AADL provides a rich set of properties which can be easily extended. These properties make AADL designs become suitable pivot model since they allow to apply

[2]www.openaadl.org

several kinds of analyses (e.g., formal methods, schedulability analysis, energy consumption).

AADL provides two concepts: immediate and delayed data communication impose strong semantics to data exchange between tasks of the same period. An immediate communication means that the result of the sending thread is immediately available to the receiving thread. A delayed communication means that the result of the sending thread is made available only at its next release while the receiving thread only reads after its own release and ultimately when the data is required. Figure 3 represents two types of data communication expressed in AADL, where τ_i and τ_j have the same period, and τ_i precedes τ_j.

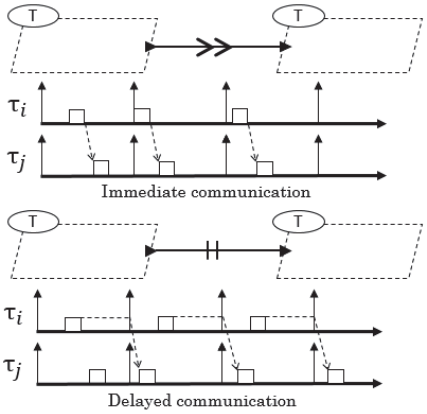

Fig. 3. Immediate and delayed communication in AADL

C. Discussion

SPC is based on a conventional set of concepts and a simple mechanism which ensures an easy and efficient implementation. In comparison to *Repetitive Precedence Constraint*, SPC is less expressive while it cannot express some communication patterns between tasks of co-prime periods. All the communication patterns presented in Figure 1, except the last one, can be expressed by SPC. Note that the case of co-prime periods (the last pattern) is very unusual and it can be completely avoided in the design phase. Moreover, regarding the temporal analysis aspect, a scheduling policy and also schedulability tests for fixed-task priority schedulers by adapting the *Audsley Priority Assignment*[9] are provided in [7] and a fixed-job scheduling policy is provided in [10]. Therefore, since the SPC is one the most powerful precedence constraints model among models having a polynomial representation while offering a good compromise between expressiveness and efficiency, we enhance it and extend it to support cycles.

On the other side, AADL provides the concept of *immediate* and *delayed* communications to offer deterministic communications, nevertheless they are limited to mono-rate communication. Indeed, in case of multi-rate communication, the semantics of these two patterns is not *deterministic* anymore. Thus, we extend AADL to enrich its modeling semantics and support SPC models.

III. Contributions

This section presents two main contributions:

- We reinforce the semantics of SPC by *(i)* allowing the initial semaphore value to be negative, and *(ii)* allowing cycles in the SPC graph.
- We implement SPC in AADL to support the multi-periodic communication. Also, we implement the associated analyses to study the schedulability of AADL system using SPC.

A. Enhancement of SPC semantics

1) Negative initial value of the counting semaphore: In [7], the counting semaphore has always a non-negative initial value, we found that the initial value of the semaphore can be negative, meaning that a certain amount of jobs of the source task (producer) have to occur before the value of the semaphore becomes non-negative and from this moment the communication acts periodically. Figure 4 presents an example of negative h.

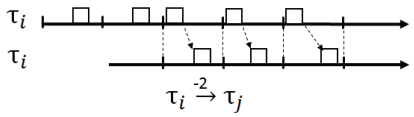

Fig. 4. Use of a negative initial value

It is useful in practical cases when the system is using a sensor such that the very first samples may not be reliable (e.g., a GPS receiver at cold start).

2) Unfolding graph: Analysis of SPC in [7], [10] is always based on an encoding technique which is efficient but not always easy to understand, in this part we introduce a new approach based on graphs for the analysis of SPC. To be comprehensive, we remind some notations and concepts.

- $\tau_{i.k} \rightarrow \tau_{j,k'}$ denotes a simple precedence constraint from $\tau_{i.k}$ to $\tau_{j.k'}$ (which means the k^{th} job of τ_i precedes the $(k')^{th}$ job of τ_j).

For the SPC $\tau_i \overset{h_{ij}}{\rightarrow} \tau_j$:

- Let $Pred_{i,j}(k') : \mathbb{N} \rightarrow \mathbb{Z}$, where $Pred_{i,j}(k')$ denotes the greatest integer k such that $\tau_{i.k} \rightarrow \tau_{j.k'}$ ($\tau_{i,k}$ is called the direct predecessor of $\tau_{j.k'}$ in τ_i). We have:

$$Pred_{i,j}(k') = \left\lceil \frac{(k'+1)T_j - h_{i,j}}{T_i} \right\rceil - 1 \qquad (2)$$

  Note that when $Pred_{i,j}(k') < 0$, this means that the job $\tau_{j,k'}$ is free from precedence from τ_i.
- Let $Succ_{i,j}(k) : \mathbb{N} \rightarrow \mathbb{N}$, where $Succ_{i,j}(k)$ denotes the smallest integer k' such that $\tau_{i.k} \rightarrow \tau_{j.k'}$ ($\tau_{j,k'}$ is called the direct successor of $\tau_{i.k}$ in τ_j). We have:

$$Succ_{i,j}(k) = \left\lfloor \frac{kT_i + h_{i,j}}{T_j} \right\rfloor \qquad (3)$$

A tasks set constrained by SPC can be represented by a *SPC graph* where each task is represented by a node and each SPC

is represented by a directed edge (see left part of Figure 5). The SPC graph can be unfolded into an infinite graph of jobs. In the unfolded graph, each node represents a job and each edge represents the simple precedence constraint between its ends.

Figure 5 presents an example of unfolding SPC graph to the infinite graph of jobs where each SPC constraint is replaced equivalently by a set of simple precedence constraints.

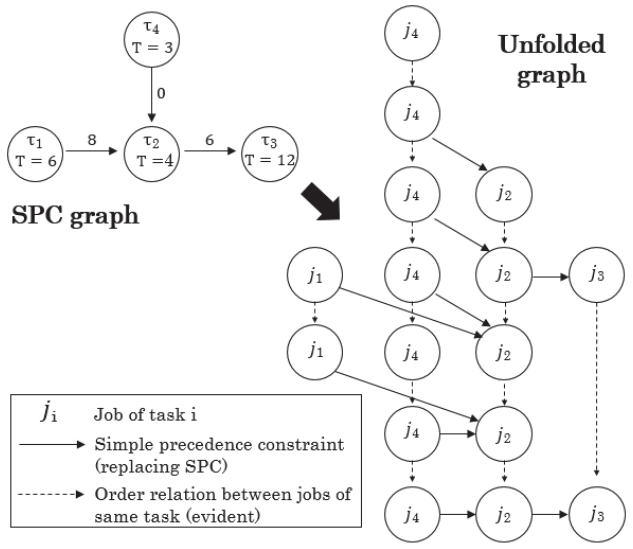

Fig. 5. Unfolding SPC graph on a time interval

Obviously, we cannot perform an analysis on the infinite graph. Fortunately, it was shown in [7] that for the original SPC, the infinite graph is ultimately periodic, that is to say: the infinite graph is composed of an initial non-repetitive part and a following repetitive part. Since we allow the initial value of the semaphore to be negative, we need to show that the unfolded graph of simple job precedence constraints is still ultimately periodic.

We consider an SPC $\tau_i \xrightarrow{h} \tau_j$, and we want to show that the corresponding unfolded graph of jobs simple precedence constraints is behaving periodically with a period $H_{i,j} = lcm(T_i, T_j)$. Algebraically, the arrows in the unfolded graph represent a bijective function $Succ_{i,j}(k) = k'$, whose inverse function is $Pred_{i,j}(k') = k$, represented in the graph by an arrow between $\tau_{i,k}$ and $\tau_{j,k'}$. The unfolded graph for one single SPC is periodic if and only if the functions $Succ_{i,j}$ and $Pred_{i,j}$ are periodic. By composition, the unfolded graph for an SPC graph is also periodic with a period given by the least common multiple of the periods of each of the single SPCs.

Lemma 1. $\forall k \in \mathbb{N} \wedge k \geq -\frac{h_{i,j}}{T_i}, Succ_{i,j}(k + \frac{H_{ij}}{T_i}) = Succ_{i,j}(k) + \frac{H_{ij}}{T_j}$

Proof. $Succ_{i,j}(k + \frac{H_{ij}}{T_i}) = \left\lfloor \frac{(k+\frac{H_{ij}}{T_i})T_i+h_{ij}}{T_j} \right\rfloor = \left\lfloor \frac{kT_i+H_{ij}+h_{ij}}{T_j} \right\rfloor = \left\lfloor \frac{kT_i+h_{ij}}{T_j} \right\rfloor + \frac{H_{ij}}{T_j}$. □

Note that the necessity for $k \geq -\frac{h_{i,j}}{T_i}$ comes from the fact that $h_{i,j}$ can be negative.

Lemma 2. $\forall k \in \mathbb{N} \wedge k \geq \frac{h_{i,j}}{T_j} - 1, Pred_{i,j}(k + \frac{H_{ij}}{T_j}) = Pred_{i,j}(k) + \frac{H_{ij}}{T_i}$

Proof. Obtained by algebraic equivalence. □

Giving the two lemmas, every $Pred$ and $Succ$ function is periodic, therefore the job precedence graph is also periodic and its period is the least common multiple of the periods of the tasks involved in the SPC graph.

We can observe in Figure 5 that the two fist jobs of τ_2 are free from τ_1, while after that, only one job every three jobs will be free from τ_1. This is because $h_{1,2} = 8$: the periodicity of the function $Pred_{1,2}$ starts only for job $k = \frac{h_{1,2}}{T_2} - 1 = 1$. On the contrary, if we look at the precedence constraint between τ_4 and τ_2, since $h_{4,2} = 0$, $Pred_{4,2}$ is periodic from the start.

3) Cycles in SPC graph: In [7], the SPC graph does not allow to have cycle. In this paper, we remove this constraint: the SPC graph is allowed to have cycles (see Figure 6) as long as the unfolded job precedence graph does not contain cycle. Obviously in case of fixed-task priority, the cycle in SPC graph (i.e., a task is successor and predecessor of another task at the same time) raises the priority assignment problem: According to [2], a task is always assigned a higher-priority than its successors. But for fixed-job priority, we found that it is possible to have cycles in an SPC graph. In this case, the priority assignment is taking place at the job level. In addition, an SPC defines relative occurrence order of target task regarding to source task but not the reverse, the cycle in an SPC graph in some case, allows to define total relative occurrence order between the jobs of the involved tasks. This strong coupling ensures the deterministic communication and eases the schedulability analysis of systems.

Figure 6 presents an example of cycle of SPC.

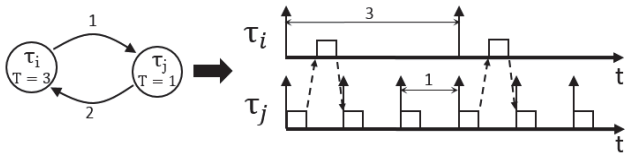

Fig. 6. Total order between jobs

Deadlock problem: An SPC graph can contain cycles, nevertheless, the underlying job precedence graph does not contain cycles only for certain insufficient count initial values in the SPC graph. Some initial counts can cause a deadlock (a cycle in the job precedence constraints). In order to be causal, the underlying job precedence graph (i.e., the unfolded graph) must be a *Direct Acyclic Graph* (DAG). The deadlock can be completely detected with a low complexity in comparison with other algorithms (i.e., $(O(V + E)$, V is number of vertices, E is number of edges)) by the topological sorting algorithm of Kahn [11], which checks if a graph is DAG.

B. Scheduling policy and schedulability analysis

1) Scheduling policy: a scheduling policy for the tasks set with precedence constraints must ensure two properties [2]: (prop-i) a job is never released before its predecessors and (prop-ii) a job has a higher priority than its successors.

The original paper of SPC provided two scheduling policies to schedule the system. The first one proposed in [7] is fixed-task scheduling policy which is based on offset adjustment and adapting the *Optimal Priority Assignment* [9] algorithm by assigning priorities in order. Nonetheless, as we discussed above, fixed-task scheduling policy cannot be used if the SPC graph possesses cycles. The second one proposed in [10] is fixed-job scheduling policy which is based on offset and deadline adjustments for EDF [1]. In case of EDF scheduling policy, the properties are then translated as follows:

Property 1. *(prop-i) can be ensured by replacing the release date of each job by the maximum release time among its predecessors[2] (offset adjustment process).*

Property 2. *Ensuring (prop-ii) consists in replacing the absolute deadline of each job by the minimum absolute deadline among its successors (deadline adjustment process).*

These original adjustments are computed by following the graph topological order at the level of tasks. For example of offset; the adjustment is carried out progressively from tasks which have no predecessors to tasks whose predecessors are already adjusted. Unfortunately, we cannot apply this order to an SPC graph with cycles because it always exists a task whose predecessor (or successor) is not yet adjusted.

We adapt the original process by changing the adjustment level of topological assignment order from task to job in the unfolded graph.

Thanks to Kahn's algorithm [11] that we have mentioned in III-A3, there are no precedence cycles between jobs in the unfolded graph. Consequently, the system is deadlock free.

2) Schedulability analysis: Once the SPC-constrained tasks set is adjusted, it acts like an independent tasks set. Therefore, we can perform the classic schedulability analysis (i.e., Demand Bound Function-DBF [12]) on the adjusted tasks set. We also implemented a simulator to study the temporal behaviour of the SPC-constrained task set.

C. Implementation of SPC in AADL

AADL provides an extension mechanism through *Property Sets*. The predefined *Property Sets* of AADL supports the basic properties for AADL components. Based on the predefined elements, users can define their own properties and component types to apply. We model SPC in AADL as a timing property of port connection. Figure 7 presents our extension in *Property Sets* to express SPC.

By default, the semaphore count does not have a unit, but the fact that the periods of source task and target task in the SPC can be different, adding unit to the account value allows to convert every value to the same unit to avoid ambiguity.

SemaphorePrecedenceConstraint.aadl

```
property set SemaphorePrecedenceConstraint is
    SPC : Timing_Properties::Time
    applies to (port connection);
end SemaphorePrecedenceConstraint;
```

Fig. 7. SPC implementation in *Property Set*

IV. Case study

A. Example description

We reuse an industrial example [7], which is the Flight Application Software (FAS) of the Automated Transfer Vehicle (ATV) designed by EADS Astrium Space Transportation for resupplying the International Space Station (ISS). Figure 8 provides an overview of the FAS software architecture. Each box stands for a high-level functionality (a task of a process) and the arrows represent the data-communications. The system first acquires data using dedicated functions: orientation and angular speed (Gyro Acq), position (GPS Acq and star-tracker Str Acq) and telecommands from the ground station (TM/TC). The *Guidance Navigation and Control* function (divided into GNC_US and GNC_DS) computes the commands to apply, while the *Failure Detection Isolation and Recovery* function (FDIR) verifies the state of the FAS and checks for possible failures. Commands are sent to the control devices: thrust orders (PDE), power distribution orders (PWS), solar panel positioning orders (SGS) and telemetry towards the ground station (TM/TC). Tasks frequencies range from 10Hz to 0.1Hz and tasks of different rates can communicate. There is also an intermediate deadline constraint, which requires data produced by GNC_US to be available at most 300 ms after the beginning of its period. (Note that, in [7], the task *TM/TC* had been split in two tasks because the communication model did not allow cycles).

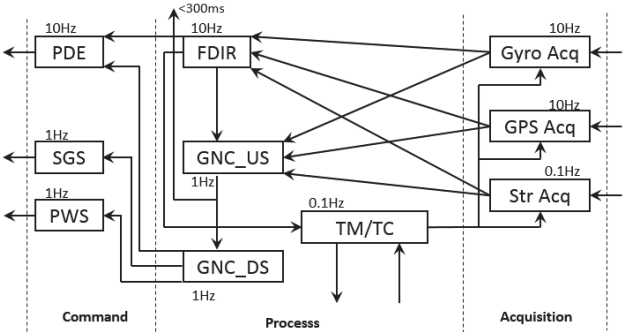

Fig. 8. FAS architecture

B. FAS model

The architecture of the FAS system expressed in AADL is presented in Figure 9, where each box is modeled as a thread, communication between FAS component is done via data port communications using our SPC extension. The detailed configuration of each FAS component and communication is presented in Table I.

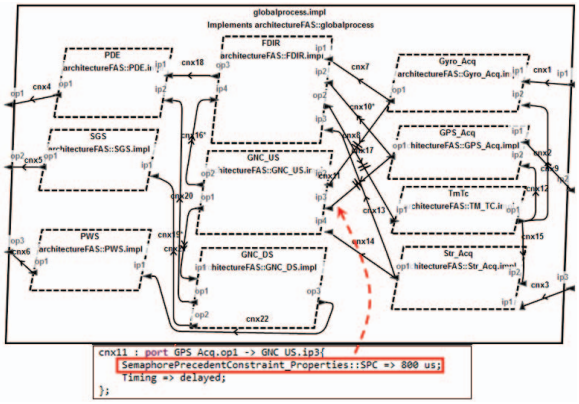

Fig. 9. Excerpt of FAS model in AADL

id	task	offset	deadline	wcet	period
τ_1	Gyro_Acq	10	100	10	100
τ_2	GPS_Acq	0	100	10	100
τ_3	FDIR	0	100	20	100
τ_4	PDE	0	100	10	100
τ_5	GNC_US	50	300	20	1000
τ_6	GNC_DS	0	1000	100	1000
τ_7	PWS	0	1000	20	1000
τ_8	SGS	0	1000	20	1000
τ_9	Str_Acq	1000	10000	200	10000
τ_{10}	TM/TC	500	10000	500	10000

TABLE I
TIMING PARAMETERS OF FAS TASKS

The SPC graph for the FAS system is presented in Figure 10. Note that the cycle of the unfolded graph contains 321 jobs.

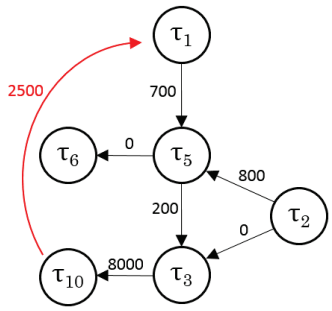

Fig. 10. SPC graph with cycle for FAS architecture

The FAS system is schedulable under EDF scheduling policy with the given configuration. Figure 11 is the screenshot of the developed tool showing the temporal behaviour generated by EDF. Note that the system cannot be analyzed with fixed-task priorities because there are cycles in the SPC graph.

V. CONCLUSION

In this paper, we extended a model of multi-periodic precedence constraints, called *Semaphore Precedence Constraints* (SPC). The constraints are expressed over tasks, but reflect precise precedence constraints over their jobs. A precedence

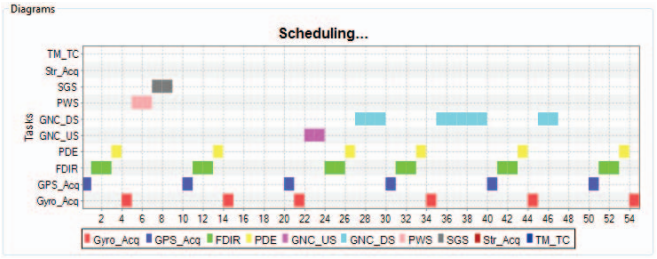

Fig. 11. FAS scheduled by EDF

constraint between two tasks τ_i and τ_j represents a periodic pattern of precedence constraints between the job, of period $lcm(T_i, T_j)$, after an acyclic phase depending on the initial value of the semaphore.

The SPC model allows to represent several "regular" precedence constraints patterns among the jobs enabling cycles in the SPC graph, as long as the unfolded precedence constraints on the jobs are a DAG. We also implemented a simple AADL extension in order to allow designers to use the SPC in their systems. Finally, we provided AADL-compliant schedulability analysis tool to analyze such a system.

REFERENCES

[1] J. Blazewicz, "Scheduling dependent tasks with different arrival times to meet deadlines," in *Proceedings of the International Workshop organized by the Commision of the European Communities on Modelling and Performance Evaluation of Computer Systems*. North-Holland Publishing Co., 1976, pp. 57–65.

[2] H. Chetto, M. Silly, and T. Bouchentouf, "Dynamic scheduling of real-time tasks under precedence constraints," *Real-Time Systems*, vol. 2, no. 3, pp. 181–194, 1990.

[3] P. Richard, F. Cottet, and M. Richard, "On-line scheduling of real-time distributed computers with complex communication constraints," in *Engineering of Complex Computer Systems, 2001. Proceedings. Seventh IEEE International Conference on*. IEEE, 2001, pp. 26–34.

[4] L. Cucu, R. Kocik, and Y. Sorel, "Real-time scheduling for systems with precedence, periodicity and latency constraints," in *Proceedings of 10th Real-Time Systems Conference, RTS02*, 2002.

[5] A. Munier, "The basic cyclic scheduling problem with linear precedence constraints," *Discrete Applied Mathematics*, vol. 64, no. 3, pp. 219 – 238, 1996.

[6] L. Besnard, A. Bouakaz, T. Gautier, P. Le Guernic, Y. Ma, J.-P. Talpin, and H. Yu, "Timed behavioural modelling and affine scheduling of embedded software architectures in the aadl using polychrony," *Science of Computer Programming*, vol. 106, pp. 54–77, 2015.

[7] J. Forget, F. Boniol, E. Grolleau, D. Lesens, and C. Pagetti, "Scheduling dependent periodic tasks without synchronization mechanisms," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*. IEEE, 2010, pp. 301–310.

[8] P. H. Feiler and D. P. Gluch, *Model-based engineering with AADL: an introduction to the SAE architecture analysis & design language*. Addison-Wesley, 2012.

[9] N. C. Audsley, *Optimal priority assignment and feasibility of static priority tasks with arbitrary start times*. University of York, Department of Computer Science, 1991.

[10] M. Forget, E. Grolleau, C. Pagetti, and P. Richard, "Dynamic priority scheduling of periodic tasks with extended precedences," in *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*. IEEE, 2011, pp. 1–8.

[11] A. B. Kahn, "Topological sorting of large networks," *Communications of the ACM*, vol. 5, no. 11, pp. 558–562, 1962.

[12] K. Jeffay and D. Stone, "Accounting for interrupt handling costs in dynamic priority task systems," in *Real-Time Systems Symposium, 1993., Proceedings*. IEEE, 1993, pp. 212–221.