FLIGHT TEST PROCESS MANAGEMENT USING STOCHASTIC PETRI NETS

Edmar Thomaz da Silva
EMBRAER – Empresa Brasileira de Aeronáutica
Edmar.thomaz@embraer.com.br

Luís Gonzaga Trabasso
ITA – Instituto Tecnológico de Aeronáutica
gonzaga@mec.ita.br

Abstract. *This paper describes a modeling and simulation tool based on Stochastic Petri Nets (SPN) theory applied to flight tests process management. [1]*
The problem might be stated as follows: there is a set of process (flight test campaigns), which share a common pool of resources such as prototypes[1], flight test instrumentation and systems availability. When the resources (or tokens) are available, the flight test can be carried out (a transition takes place). The terms in parenthesis corresponds to the SPN terminology.
The simulation tool is therefore, an adaptation of the general Petri Net theory for flight test campaign planning and management. The paper shows the necessary steps to render the adaptation possible. The derived model is used for simulating real flight test campaigns.
Comparing the situation with and without the modeling and simulation tool, it is possible to conclude that the risk of severe mistakes caused by faulty assumptions of time, costs and safety can be reduced significantly, since the dynamic of process can be checked.

Keywords*. Simulation, process management, flight tests, stochastic petri net.*

1. Introduction

Aircraft development process is characterized by high complexity and ever-decreasing lead times throughout, aiming at shortening time-to-market with the required quality and low costs.

Development delays experienced in single systems, systems integration and those due to design oscillations (design non compliances identified during flight tests that demand re-design and new test campaign) represent the main treats for the aeronautical industry to accomplish the market and certifying boards schedules.

In spite of the fact that the organization might be structured for applying concurrent engineering concepts and tools [2][3], several factors impair the expected benefits of the concurrent engineering. It is usual that the phases prior to certification experience some delays however, the dead line for the certification phase remains the same. Consequently, the flight tests campaign is compressed in terms of time length for avoiding the delays in the overall schedule.

Flight tests activities are complex, expensive and eventually hazardous. Costs and schedule completion criteria have to be balanced out with safety and quality when a flight test campaign is planned.

Flight test certification process is a set of flight test campaign that have to be done in order to certify an specific issue or system of the aircraft e.g.: handling qualities, auto pilot, communication systems, among others. Planning and managing activities in this highly dynamic scenario are very complex tasks. Since there is a great number of uncertainties related with systems maturity and aircraft behavior (in terms of availability for flight).

A modeling and simulation tool that provides the flight tests managers with the possibility to simulate the impact of these uncertainties in the flight test campaign evolution is extremely welcome. This tool has to take into account the inherent characteristics of the flight tests, which are concomitant, asynchronous, parallel and stochastic. Besides, this tool has to have a graphical, user-friendly interface, in order to make possible a visual modeling technique of the concurrent activities of the tests campaign. It has been searched and found that a Colored Petri Net (CPN) suporting stochastic process allows for modeling and simulating the necessary characteristics mentioned above.

The modeling and planning tool has been designed and tested on the certification flight test program typical scenario. This was done through a commercial CPN tool - the ExSpect 6.41® - and a set of relevant flight test campaigns. Comparing the situations with and without the simulation tool, it is possible to conclude that the risk of severe mistakes caused by faulty assumptions for the flight test campaign evolution can be reduced significantly, since a stochastic approach can simulate the real process.

The manuscript is organized as follows: section 2 describes the key characteristics of the flight test process. Section 3 reviews the Petri-Net (PN) concepts. Section 4 describes the selection of a PN tool. The adaptation of a PN tool to fit the flight test campaign management is shown in section 5. Section 6 describes the results and analysis of the simulation, followed by the key conclusions in section 7.

[1] Prototype in this manuscript, is an aircraft that is used for certification purposes.

2. Flight test process definition.

Flight test planning is a process that is detailed concurrently with the aircraft program development. It is divided in three activities: initial planning, high level planning and detailed planning. Described in the next paragraphs.

2.1. Initial planning.

It is a planning made at the beginning of the aircraft or system development. The main idea of this activity is to identify the major activities, efforts, costs and schedule of the new product development.

2.2. High level planning.

High level planning is the arrangement in time of all tests related with the same system or subject. In this work, these tasks are called flight test campaign (eg. Handling qualities, flutter tests, radio and navigation etc).

Usually, the high level planning is very dynamic and has to be change at the measure of the knowledge about the prototype increases.

The objective of this work is related with this topic and aim, by means of simulation, to became this planning more predictable and less subject to change and mistakes.

2.3. Detailed planning.

Consists of detailing the flight test activity at its level more accurate and try to answer the following questions: which test point will be made in which flight using which prototype and when this flight will take place.

2.4. Others relevants concepts related with flight tests and used in this work.

In order to understand the flight test activities, it is necessary to explain the following activities and concepts:

- **Flight test proposal formulation** - the outcome of this activity is the Flight Test Proposal Report (FTPR), produced by all engineering areas that need to test their systems in flight. The FTPR depicts, in a formal and detailed way, all the information necessary to plan a flight test campaign such as: procedures, requirements, expected results, risks, flight test instrumentation, and so on.
- **Flight test point -** The flight test point or simply (TP) states the necessary conditions to be met in order to start up a given flight test. TP is defined by:
    - Initial condition – is the condition (altitude, velocity) and configuration (weight, center of gravity position) the aircraft will have to be in the space;
    - Action – is the action the pilot will have to carry out to change the initial condition for observing and collecting the data related to the aircraft and its systems behavior.
- **Flight test order** – (FTO) is the document that defines one flight and authorizes the crew to execute it. It consists of a set of TP´s ordered according to criteria of safety, time optimization, aircraft configuration and resources allocation.
- **Flight test campaign** – (FTC) is a set of FTO related with the tests necessary to demonstrate that one specific aircraft system or subject complies with the requirements defined in the FTPR.
- **Aircraft request** – (AR) is the document that informs to mechanician and flight test instrumentation personnel what will be the resources necessary to the flight and to configure the aircraft. When an aircraft request is approved, it means that all the resources necessary to execute the flight are available and the aircraft is ready to fly.

3. Petri net basic concepts.

PN´s consist of four primitive elements: tokens, places, transitions, and arcs showed in the Figure 1 and rules that govern their operation [4]. PN´s are based on a vision of tokens moving around a network. Tokens represent the objects or entities in a system.(e.g. Prototype, flights, flight test campaign). Places represent the locations where the tokens await processing (E.g. number of flights, AR available, etc). Transitions represent processes or events (e.g., execution of a flight). Finally, arcs represent the paths of objects through the system. Arcs connect places to transitions and transitions to places; an arrowhead at the end of the arc indicates the direction of the path.

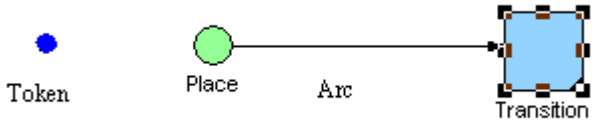

Figure 1. Depiction of PN Primitives.

PN firing rules specify the behavior of transitions; i.e., the conditions under which processes or events can occur. Three rules govern the transition firing:

- When all upstream places are occupied by at least one token, the transition is enabled;
- Once enabled, the transition fires;
- When a transition fires, one token is removed from its upstream place and put at a downstream place.

The concepts described above are applied to ordinary PN. Stochastic Petri Nets (SPN) differentiate from ordinary PN because their tokens might carry data values, time and conditions for activation and can, hence, be distinguished from each other. This contrasts with the tokens of an ordinary PN, which by convention, are drawn as black, "uncolored" dots.

4. Selection of a petri net tool.

There are a great number of Petri Nets tools available, in order to select one, three basic components were considered: software evaluation, kind of PN supported and components tools.

4.1 Software evaluation.

Two basic restrictions were imposed for software selection:

- Supported by **windows plataform**;
- **Shareware**.

The other aspects that were considered are the following:

- **SW stability** – capacity to functions without crashes the PC;
- **SW scalability** – capacity to support model expansion in order to evaluate the model as close as possible of the real problem;
- **SW documentation** – publications and help available about the SW operation.

4.2 PN supported.

PN theory was largely explored in the past years so, a great number of approachs was develop in order to adapt the theory to different kinds of problems. The approachs that were considered are the following:

- **High-level Petri Nets** - Coloured Petri Nets and Predicate/Transition Nets;
- **Object-oriented Petri Nets** - Petri Nets with object-oriented extensions such as classes and objects;
- **Stochastic Petri Nets** - Nets with stochastic time that typically can be used for performance analysis;
- **Petri Nets with Time** - nets with time that can be used for performance analysis.

4.3 Tool components supported.

Every kind of PN has a set of typical components that expands the possibility to evaluate the system behavior.

- **Graphical Editor** - Graphical user interface which supports editing of nets in a graphical representation;
- **Token Game Animation** - Simulation with animation of the flow of tokens;
- **Fast Simulation** - Simulation without graphics to allow maximum simulation performance;
- **State Spaces** - Reachability graphs/trees and occurrence graphs;
- **Simple Performance Analysis** - Simulation with time;
- **Advanced Performance Analysis** - Performance analysis such as Markovian chains;
- **Interchange File Format** - File format for exporting and importing data and model diagrams to and from other tools.

4.4 Comparison among Petri Nets tools: adherence to specific requirements.

Considering the requirements specified in the previous paragraphs, small models were implemented and tested in the following tools:

- CPN tools 1.29 - University of Aarhus;
- Exspect 6.41 - Deloitte & Touche Bakkenist and Eindhoven University of Technology;
- HPSim - Henryk Anschuetz;
- PNSim 1.0 - Georgios Markatatos;
- POSES++ - GPC mbH;
- Simulaworks - Mohamed Ali Sobh.

The evaluation is presented in Table 1 below.

Table 1. PN tool selection.

Features	CPNtools	ExSpect	HPSIM	PNSIM	Poses++	Simulaworks
Stability	0	9	8	8	5	10
Scalability	0	7	5	0	10	5
Documentation	5	8	10	5	5	5
High level	7	10	0	0	10	0
Object oriented	0	10	0	0	5	7
Stochastic	0	10	0	0	5	7
Timed	5	10	10	0	10	10
Graphical editor	2	10	10	10	7	7
Token game	5	10	10	10	10	7
Fast simulation	5	10	5	0	10	10
State space	0	7	0	0	5	0
Simple performance	5	10	10	10	10	10
Advanced performance	0	10	5	0	10	7
Interchange file	5	0	0	0	10	7
Final punctuation	39	121	73	43	112	92
Final punctuation. (%)	28	86	52	31	80	66

ExSpect is the tool that best fits to the scope of this work. In particular, the important aspects that can be pointed out are the capacity to construct and change the model in a straightforward way. The software's libraries give the capacity to expand the possibility to model different aspects of the problems. Graphical interface is another good feature of the software since it allows for tracking the simulation parameters in a dashboard. Software instabilities specially when running the simulations and the lack of adequate documentation which impairs a better utilization of the features available are the negative points of the software.

POSES++ is a powerful tool designed to simulate large systems in real time since it is based in C++ language. The software tool has been built in multi client/multi server system connected via TCP/IP inside a LAN and over INTERNET. This approach makes the tool heavy to be operated in a single PC, which affects negatively the tool characteristics requirements.

Simulaworks is a tool similar to Exspect however it does not have all the features supported by the latter.

HPSIM and PNSIM are tools developed as students' assignments in universities. HPSIM supports only small didactical examples of single PN. PNSIM has a great number of problems in the software installation process thus, it was not possible to evaluate this software.

CPNTOOLS is a software tool still under development with a great number of unresolved problems. It generates great instabilities in the host PC and requires a huge amount of RAM memory to run small models.

5. Flight test – simulation model.

Usually, a flight test certification process has up to 300 flight test campaign. The basic problem for the flight test manager to make an accurate high level planning is related with the following uncertainties:

When the prototype will be ready for flight ?

The different systems maturity is reached during the flight test. It is not possible to predict exactly when one system will be ready to be test. It have to be taken into account yet that at the beginning, new system´s malfunction can cancel the flights.

After starting the flights what will be the best estimation for the flight rates?

Since the system are not in the final configuration, there is a great number of problems related with reliability that naturally, are corrected during the development, but, during the flight tests can impair the flights´ evolution and the prototype availability for flight.

The flight test manager has to guess about the mentioned problems. Usually the “guess” is based in a very strong experience in flight test and make sense for small programs and programs derived from other one previously tested. For certification process of new products the huge number of different activities make this approach not applicable (a set of dependent good guess is not a good guess).

In order to fit the flight test problem into the SPN tool it has been designed a simplified model that has a set of integrated modules described in the next paragraphs.

Figure 2 depicts the system through its components and functions.

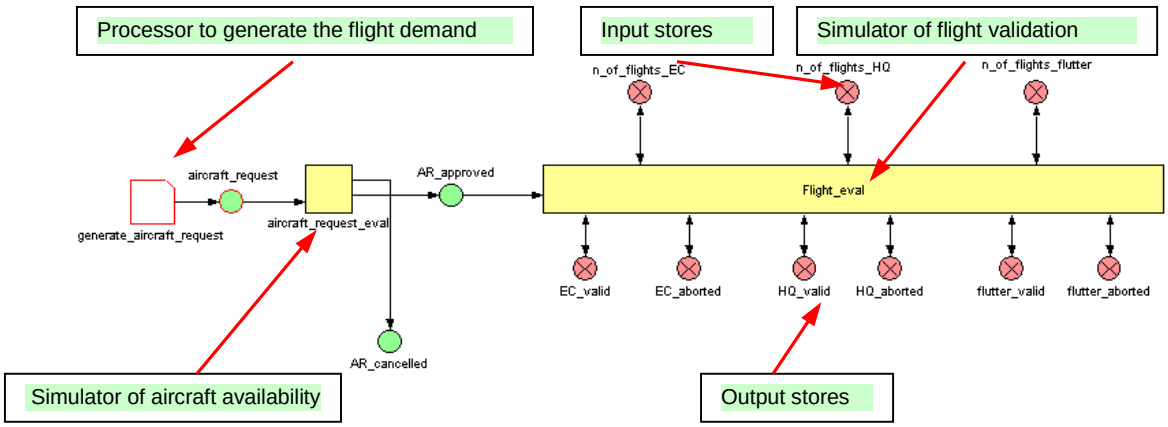

Figure 2 System to evaluate the aircraft availability for flight.

The implementation of every component is explained in details in the next paragraphs.

5.1 Processor to generate the flight demand.

This processor programs and adjusts the arrival time between two generated tokens (every token represents one possible flight), according to the simulation phase. The arrival time is modeled through the statistic function “nexp”. This function generates tokens with a specified mean and variance of the arrival time. By adjusting its parameter, it is possible to simulate periods when it is expected good and bad flights rates (eg. good or bad weather conditions which define flight executions).

The basic components of this processor are shown in Figure 3.

.

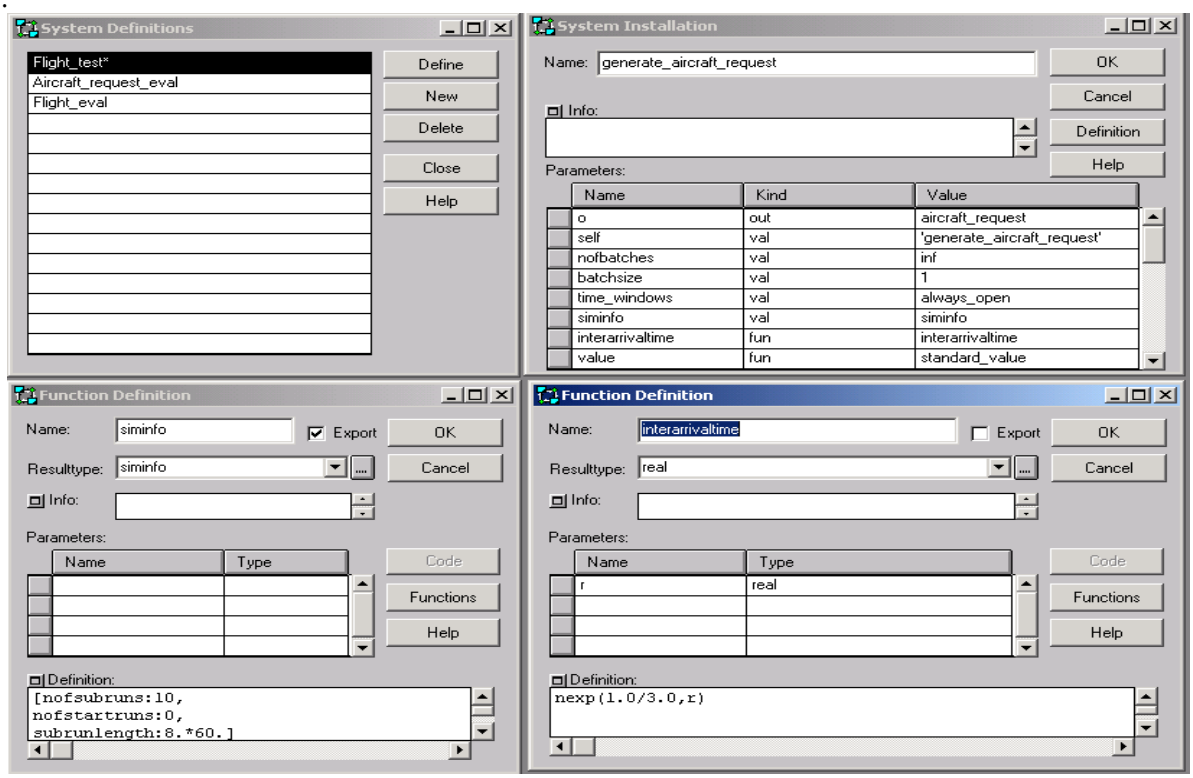

Figure 3 Construction’s details of the processor to generate flight demands.

5.2 System to simulate the aircraft availability.

After generating the flight request, the system "aircraft request eval" simulates the aircraft availability for flight. The aircraft availability is modeled through a probability function that sums up the real conditions observed during the prototype evolution. In the first flights, it's observed a great number of maintenance problems that impair the flight execution, then, this situation improves along with the prototype evolution.

The probability of execution is programmed inside the processor "*aircraft_request_validation*". The processor is shown in Figure 4. The probability of executions is represented by a threshold value that is compared with a random value. This value is generated by a random store that produces random values. The values follow an uniform distribution between 0 and 1. The random store is synchronized with the arrival of tokens in the aircraft_request channel. Flights that are approved are sent to the channel "*aircraft_request_approved* and those that are rejected are sending to "*aircraft_request_cancelled*". Both channels are constructed with special features that allow for data exchange with external spreadsheets for further analysis.

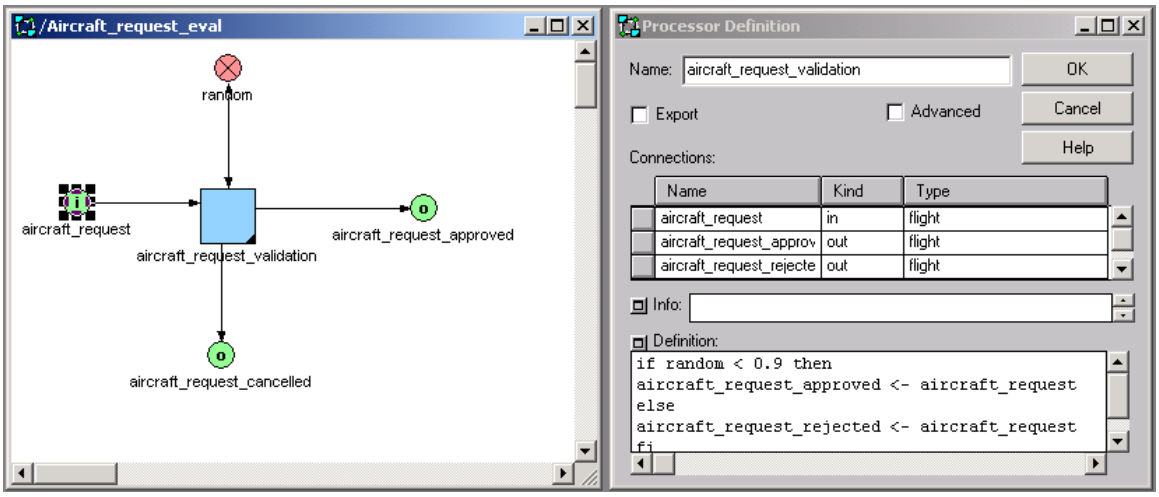

Figure 4 Depict of the system "aircraft_request_eval".

5.3 System to simulate the flight validation.

If the aircraft request is approved, the request enters in the module that simulates the flights validation described in the Figure 5.

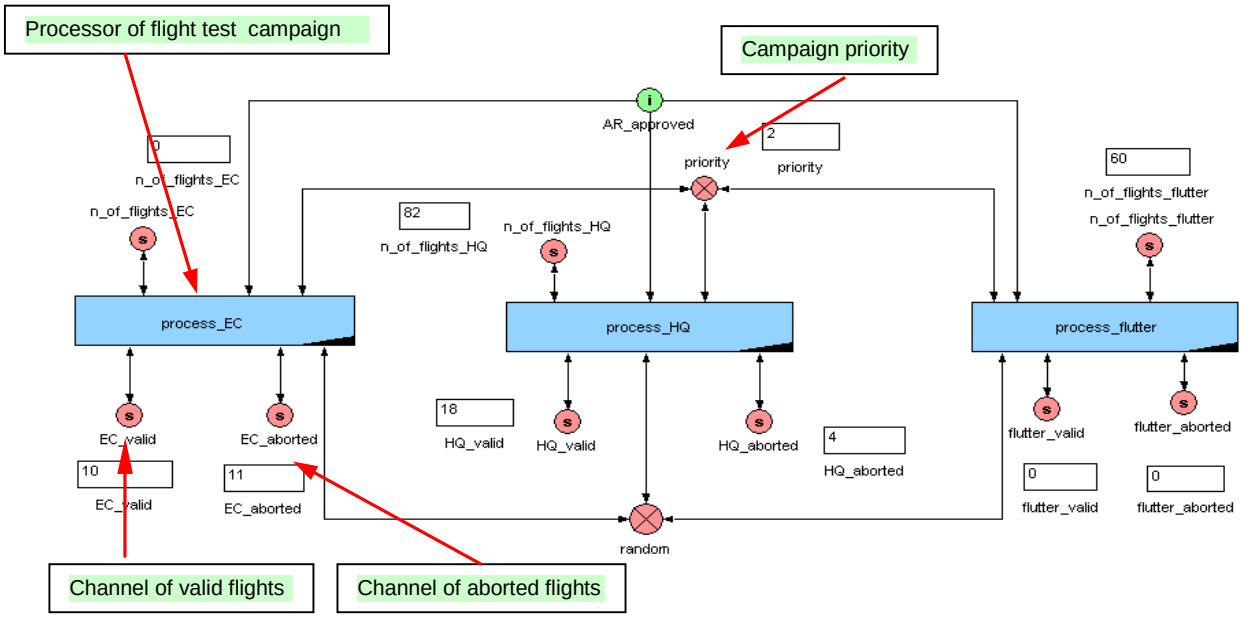

Figure 5 Module which simulates the flights validation.

The basic components and functions of the module are described below.

5.3.1 Processors.

A set of three processors (named *process_xx*) simulates the average flight rate expected for every specific flight test campaign. The probability of the flight validity differs among the campaigns. This is due to three main factors:

- Pilot's skills to make the maneuver. Some test points has very tied limits for stabilization of some parameters such as velocity and altitude.
- Flight test instrumentation sensibility. Due to the great number of complex data acquisition systems, it is possible that some test points become invalid due to fails in these systems;
- Fail in aircraft systems – especially at the beginning of the tests.

There is a probabilistic function implemented in the processor that defines if one flight has been accomplished with success. The details of process definition are shown in Figure 6.

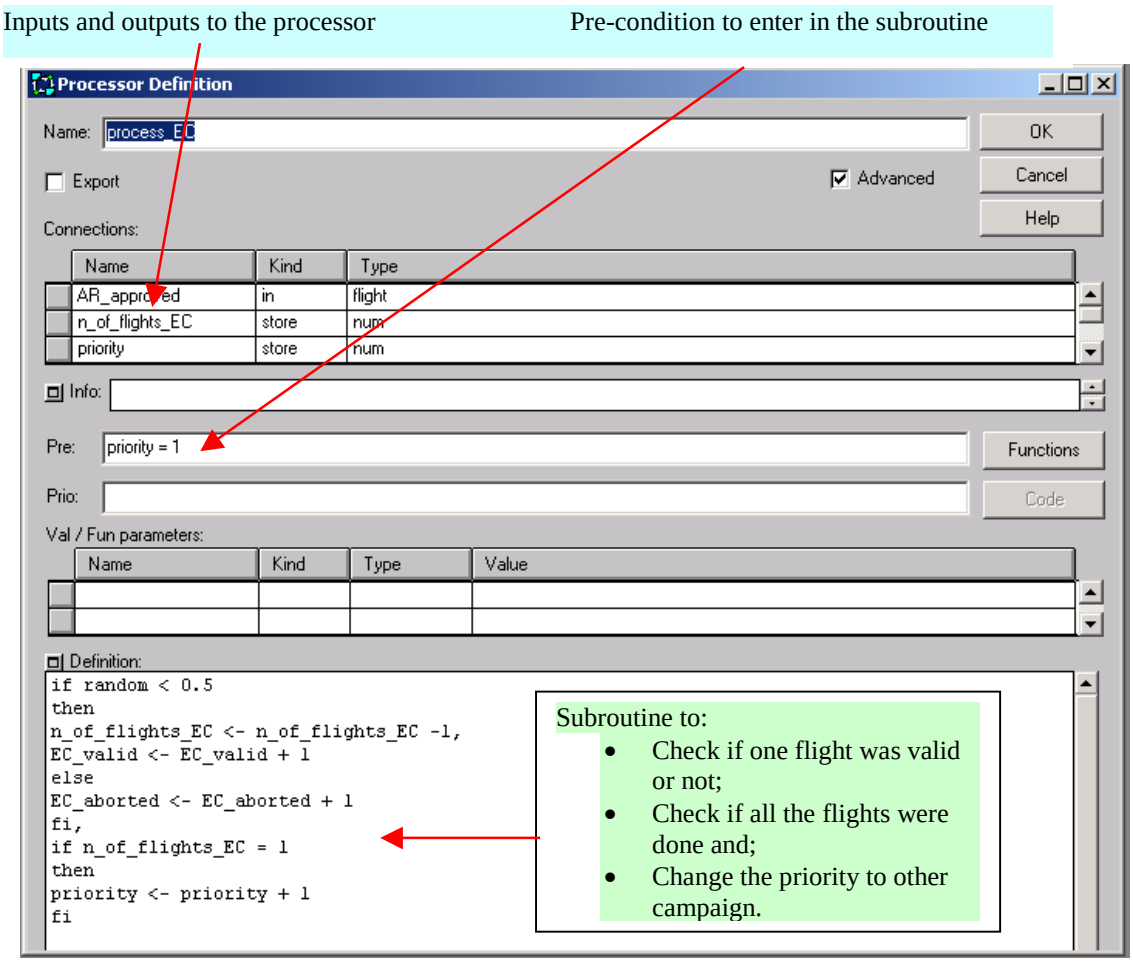

Figure 6 Flight validation processor.

If the flight is valid, the total number of flights is decremented (channel "*n_of_flights_xx*" – Figure 5*)* and the channel "*xx_valid*" receives one token. Otherwise, the channel "*xx_aborted"* receives one additional token. The token identification and arrival time are sent to a spreadsheet for posterior analysis.

When the flights of a specific campaign finish, the priority changes and the flights of the campaign with next priority are enabled.

6. Results and analysis of the simulation.

It was simulated three representatives flight test campaign:

- Flight envelope clearance – the initial flights that explore the flight envelope (velocity, altitude and systems operations);
- Handling qualities – this is usually the biggest flight test campaign and demand a great effort to be accomplished;
- Flutter test – a hazardous flight test campaign.

The results are described in the next topics.

6.1 – Prototype availability evolution.

The aircraft availability for flight can be measure by the parameter aircraft request cancelled.
The figure 7 bellow shows the evolution of this parameter.

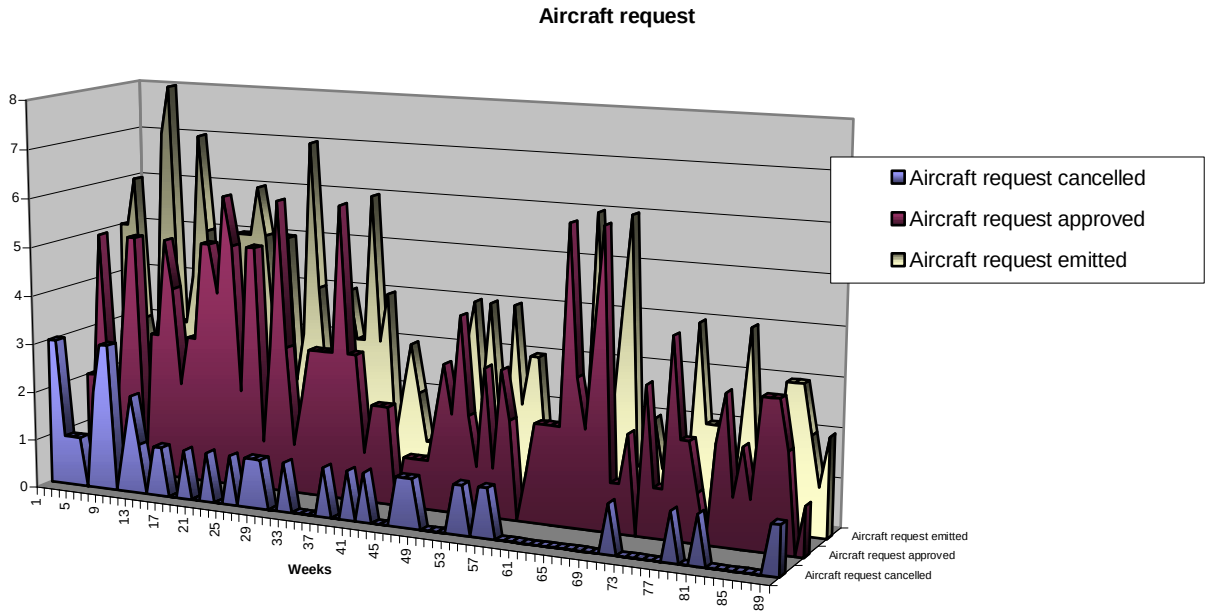

Figure 7 – Aircraft request evolution.

The block “aircraft request eval” was modeled in order to represent the aircraft maturity with time.

The sequence aircraft request cancelled shows that at the beginning of the flights the prototype availability is low. There are a great number of flights canceled not only by the aircraft systems malfunction but also by systems equipments that support the flights, specially the flight test instrumentation.

The natural tendency is that this kind of problem became scarse with time. At the end it is a minor problem.

One of the most important parameter used to evaluate the aircraft quality is its dispachability, so the project has to become robust.

An accurate definition of prototype availability is very important for the flight test manager since one mistake in this estimation can impair strongly the certification board schedule.

6.2. Scenario 1 – First flights.

The simulation of this flight test campaign represents the beginning of the prototype operation. The results are showed in the figure 8.

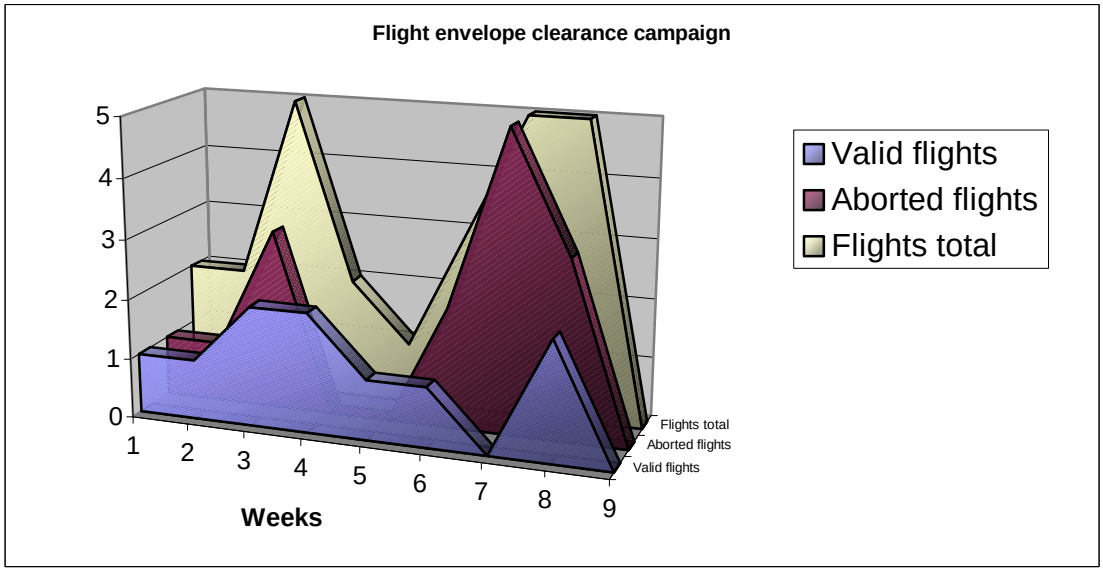

Figure 8 Evaluation of the prototype behavior at the beginning of operation.

The beginning of prototype operation is characterized by a low dispachability associated with a low evolution in terms of rate of flights.

In this example it was expend 9 weeks to accomplish the task and the number of aborted flights (15 - useless flights) is greater than the success ones (10).

In order to planning this phase, it is necessary to avoid good expectations. It is better not try to explore the flights but instead, to plan a low number of flights and gather the necessary data to help engineers to became the systems project more robust for the subsequent phases.

6.3. Scenario 2 – Steady state behavior and external factors impacts.

The steady state is represented by the handling qualities campaign. At this point the major problems found in the first flights were solved and the prototype became more stable with time. Figure 9 shows the simulations results.

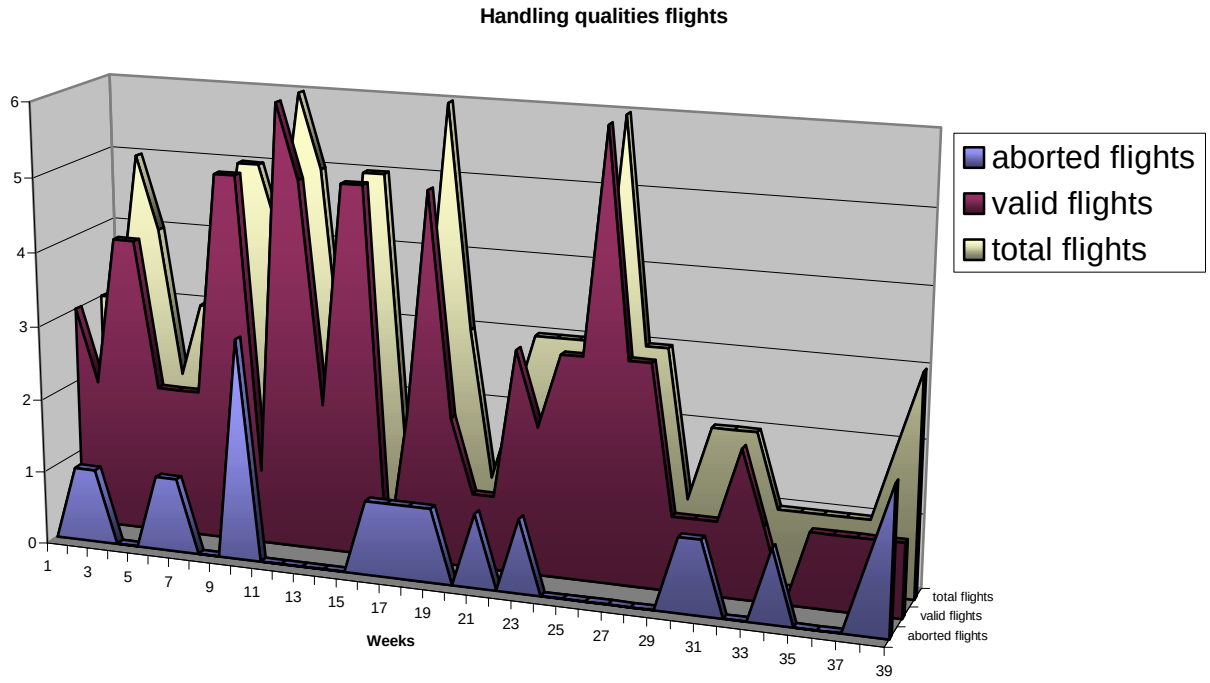

Figure 9 – Steady state behavior and external factors impacts.

At this phase it is possible to observe a better correlation between the “intention” to flight and the valid flights. The flight rate is good (4 flights against 1 per week in the previous case).

This situation is stable until week 29 when the average returns to 1 flight per week. In this case the reduction in the flight rate is going together with the aircraft request. This scenario represent the impact of external factors like bad wheather conditions and usually happens in summer (rainning days).

Since the prototype is stable, for this phase the manager has to make efforts to maximize the number of flights in the days not impacted by the weather conditions.

6.4. Scenario 3 - Hazardous flight test simulation.

Hazardous flight tests requires the support of chase aircraft and the readiness of a search and rescue helicopter. This introduces additional problems that can lead to low dispachability. In addition there is a work of data reduction that demand more time between flights. As consequence, the flight rate is reduced in this phase.

The results are showed in the next figure.

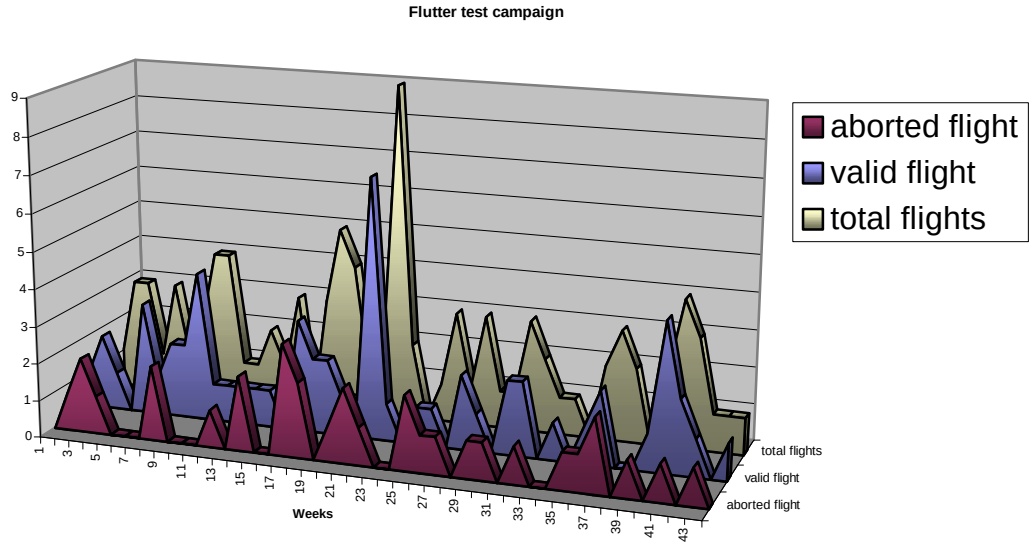

Figure 10 – Hazardous flight test evolution.

The flight rate is low at the beginning due to the necessity to adjust the flight test instrumentation and increase until week 21. This represents the flights that are made inside the envelope with low risk.

The reduction represents the exploration of points out of the envelope when the time to analyse data and give the clearance for the next flight increase.

7. Conclusions.

This paper has demonstrated that the Stochastic Petri Nets theory is suitable to model and simulate the flight test process for aircraft certification.

It has been used the CPN tool ExSpect 6.41 and three scenarios that has to be analysed in order to have an accurated planning where simulated.

As a graphical technique the CPN tool make it easy to visualize the system being modeled and to communicate the resulting model. This means that the information can be obtained in a fast, clear and direct way, allowing the flight test manager to take complex decisions considering strong restrictions of safety, costs and time.

8. References.

[1] Aalst, W. V., Oberweis, A.,2000, “Business Process Management”. Vol. 1, Springer Verlag.
[2] Huang, G.C.1996 “Design for X - Concurrent Engineering Imperatives”, Chapman&Hall.
[3] Pahl, G., Beitz, W.,1996, “Engineering Design : A Systematic Approach” , Springer Verlag,
[4] Jensen, C.,2000 “An Introduction to the Practical Use of Colored Petri Nets”, Department of Computer Science, University of Aarhus.
[5] Petri net WWW pages. URL: http://www.daimi.aau.dk/PetriNets/.