Synthesis of software architecture for the control of embedded electrical generation and distribution system for aircraft under safety constraints: The case of simple failures

LAURENT ZIMMER[1], PIERRE-ALAIN YVARS[2]

[1] DASSAULT AVIATION
78, Quai Marcel Dassault, 92252 Saint Cloud, France
laurent.zimmer@dassault-aviation.com

[2] InstitutSupérieur de Mécanique de Paris (SupMeca)
Laboratoire QUARTZ
3 rue Fernand Hainaut, 93407 Saint Ouen Cedex, France
pierre-alain.yvars@supmeca.fr

***Résumé* – Dans l'aéronautique, le respect des exigences de sûreté de fonctionnement est le facteur déterminant pour la définition des architectures de systèmes. A l'avenir, les architectes systèmes devront s'aider d'outils d'assistance à la génération d'architectures correctes par construction. Dans ce contexte, nous nous sommes intéressés à la modélisation formelle des exigences et à la résolution informatique de ces modèles formels pour trouver des architectures admissibles. Dans ce papier, nous proposons une approche et un ensemble d'outils pour au moins vérifier puis synthétiser une architecture à sûreté de fonctionnement intégrée d'un système embarqué avion de production et de distribution de puissance électrique. Nous avons développé une approche à base de modèle qui utilise le langage DEPS (DEsign Problem Specification), un langage dédié à la modélisation et à la résolution des problèmes de conception. Les résultats obtenus montrent qu'il est possible de modéliser des exigences complexes de sûreté de fonctionnement au niveau requis par l'architecte système et que leur prise en compte pendant la résolution génère des solutions correctes vis-à-vis de celles-ci.**

***Abstract* – In aeronautics, compliance with safety requirements is the main driver in the definition of system architectures. In the future, system architects will have to use tools to assist in the generation of correct architectures by construction. In this context, we are interested in formal requirements modeling and in the computer resolution of these formal models to find eligible architectures. In this paper, we propose an approach and a set of tools to at least verify and then synthesize a fail-safe architecture of an on-board aircraft electric power generation and distribution system. We have developed a model-based approach using the DEsign Problem Specification (DEPS) language, a language dedicated to modeling and solving design problems. The results obtained show that it is possible to model complex dependability requirements at the level required by the system architect and that taking them into account during the resolution process generates solutions that are correct with respect to them.**

***Mots clés* - système embarqué avion de génération et de distribution électriques, sûreté de fonctionnement, synthèse à base de modèle, programmation par contrainte, vérification.**

***Keywords* – on board aircraft electric power generation and distribution system, safety, model based system synthesis, constraint programming, verification.**

1 INTRODUCTION

In this period of ecological and energy transition, several industrial sectors are being led to rethink their concepts to ensure a reduction in greenhouse gases. This is notably the case in the aeronautics sector, where we are talking about a more electric, hybrid electric or all electric aircraft. These concepts, concerning the systems architecture domain, aims to replace hydraulic and/or pneumatic systems with electrical systems performing the same functions. On the one side, the expected benefits are numerous: better power rationalization, weight savings, improved aircraft availability and reduced maintenance costs. All three will increase the profitability of commercial aircraft and meet current and future environmental requirements. On the other side as explained in [Menu et al, 2018] the increasing use of electrical components and electrical power systems gives rise to new constraints, particularly on the electrical power generation and distribution system architecture embedded in the aircraft. The replacement of hydraulic and pneumatic networks will result in a shift in reliability and safety requirements to the electrical network.

Engineers will have to design and verify new and more complex architectural solutions in order to respond to these requirements while limiting the impact in terms of mass [Giraud, 2014].

In this paper, we do not consider the sizing of system but we present through a simplified use-case inspired from an industrial system architecture how we can take into account the safety requirements during the design process:

In this paper, firstly we will present the issue, secondly our approach, thirdly some related work, fourthly we will present the use-case that is to say the embedded power generation and distribution system that we are studying as well as the associated safety requirements.

Fifthly we will present the DEPS problem modeling language [Yvars and Zimmer, 2014], dedicated to the formalization of system synthesis problems. The modeling of the problem in DEPS will then be detailed. The models will be compiled and solved under the DEPS Studio modeling and solving environment associated with the DEPS language.

Finally, some perspectives of evolution will be evoked.

2 ISSUE

In aeronautics, compliance with operational safety requirements is the main driving force behind the definition of system architectures.

These system-level requirements are generally the result of preliminary reliability analyses that determine the failure rate of functions critical to operational safety.

To ensure that failure rates are above the desired thresholds, system architects must implement architectures with a number of hardware and functional redundancies.

In particular, these dependability requirements will lead architects to develop back-up functions that will replace flight-critical functions when they fail. This is known as fail-safe architecture.

These architectures are difficult to develop and verify because redundancy clashes with other design criteria such as weight and cost, which favor the sharing of components. Failure of components common to normal and backup functions causes the fail-safe nature of the architecture to be lost. In order to avoid the problem it is mandatory to ensure that the realizations of normal and backup functions are segregated (i.e. they have no material elements in common).

In the following, we propose an approach and a set of tools to at least verify and later synthesize a fail-safe architecture of a power generation and distribution system.

More precisely, we will simultaneously address the following problems:

- to check that the electrical hardware architecture meets the the operational safety requirements with regard to the various failures that may occur during a flight;
- to synthesise the software architecture needed to control the reconfiguration of the electrical system when a failure occurs.

3 APPROACH

We adopt a model-based systems engineering approach with two levels of modeling: one level for the functional architecture of the system and one level for the physical architecture.

At the functional level, the power supply system services are organized in safety patterns that group together the normal and backup power distribution operating channels. It is at this level that the properties of physical segregation between channels will be expressed.

At the hardware level, we find the components that make up the physical architecture of the system, including: AC generators, bus bars, contactors, AC/DC converters and calculators.

We formalize the problem of verifying the conformity of the architecture to safety requirements as a generalized deployment problem (or resource allocation problem).

Generation, Distribution, Conversion, Command and Supervision functions, which are part of safety patterns, must be deployed on generators, distribution paths, convertors and calculators in a way that respects the segregation constraints.

It is important to point that these segregation constraints have to be expressed at the functional level even if they act at the hardware level. On the one hand, these segregation constraints are expressed on functions and channels and, on the other hand, we would like to have some form of genericity and reusability of the models.

In the following, we detail how we use DEPS a modelling language with constraint to set and solve the checking and/or synthesis problems of fail-safe architecture.

4 RELATED WORK

The majority of the published work in the field of embedded electrical power system design focuses on solving problems by satisfying some functional requirements rather than formalizing the problem and requirements. Thus, [Giraud, 2014] proposes an approach designed to solve the problem of load allocation on an aircraft electrical network. It involves sizing the energy sources and generating the connection paths between the energy sources and the loads in such a way that the energy demand is satisfied. The load allocation calculation is done using the implementation of a genetic algorithm. No formalization of the load allocation problem using an adapted description language is proposed.

Other researchers practice design by simulation of embedded electrical architecture as in [Yang et al, 2015]. Here again there is no abstract modeling of the problem. The performance of a given system is evaluated by simulation. If the result is suitable, the architecture is accepted; if not, it is necessary to manually modify some parameters of the system and then simulate it until a satisfactory solution is found. It can take a long time, especially when there are multiple requirements to be met. On the other hand, it is possible to achieve a very fine level of granularity for the physical models used.

From the point of view of taking into account safety requirements, [Menu et al, 2018] propose a two-step design process: the use of a graphical formalism intended to represent the architecture of the electrical system and a low level of variability reduced essentially to the range of possible values for the cardinality of the system components. A generation of the cartesian product of the candidate architectures is then carried out and fault trees are used to evaluate the failure probabilities of the different generated systems, with the designer having to choose the satisfactory solution. The limits of this work concern on the one hand the weakness of the expressible variability and on the other hand the evaluation of the solutions.

However, as early as 2010, [Becz et al, 2010] and [Pinto et al, 2010] emphasize the need for both problem modeling formalism at a sufficient level of abstraction and synthesis tools to represent and solve complex system design problems in embedded aeronautics.

In our case, we will see that we have a typical case of formally representing and solving a problem of synthesis of software architecture for the control of the embedded electrical generation and distribution network in compliance with the safety requirements of this network.

[Creff et al, 2020] point out the difficulties of using formalisms such as SysML (System Modeling Language), initially designed to represent totally defined systems to model problems. They propose in their work to use the Clafer [Bak et al, 2014] formalism associated with the Choco [Lorca et al, 2014] constraint programming library to model and solve a problem of allocating calculators to embedded tasks. Clafer is a feature oriented modeling language [Kang et al, 1990] with very limited reusability capabilities. As a result, it remains a language dedicated to the configuration of software product lines. On their side, [Leserf et al, 2015] propose to add a first level of variability to the SysML language intended to be more universal. The approach is coupled with the Choco library to solve simple configuration problems. This work has so far remained at the research stage. The current limitations of these two approaches are the low level of variability that can be taken into account, the use of a solver handling essentially discrete constraints and a weak coupling between the formalism and the solver, which means that the development of models is carried out in case of a problem in the solver language and not in the Clafer or SysML language.
All these limitations have already been pointed out by Shah [Shah, 2010] and [Shah et al, 2012]. We will see in this paper that the DEPS language (DEsign Problem Specific language) [Yvars and Zimmer, 2014] and its recent integrated modeling and solving environment DEPS Studio [Yvars and Zimmer, 2019] are an attempt to address these limitations.

5 PROBLEM DESCRIPTION

5.1 System description

The case study is an aircraft electrical Power Distribution System (PDS). A PDS is responsible for the distribution of energy from the various generators to the various on-board loads (Figure 1).
From an electrical point of view, the system is composed of the following hardware elements:
- Three generators (GLC1, GLC2, RAT) which provide the electrical power;
- Eight bus bars (LH MAIN AC, LH ALT AC, LH MAIN DC, LH ESS DC, RH MAIN AC, RH ALT AC, RH MAIN DC, RH ESS DC) that provide electrical distribution services to the loads connected to them;
- Three converters to transform alternating current into direct current (T1, T2, SBT);
- Seventeen power contactors to connect and disconnect certain parts of the network in order to conduct or interrupt the transfer of power between these parts.

The network topology of the power generation and distribution system can evolve dynamically under the effect of calculators that control the contactors by means of their control ports. The calculators and the connection between the ports and the contactors are not shown on Figure 1.
Note that the system has been sized by electrical engineering specialists so that GLC1 or GLC2 alone can provide the electrical power required by all the loads.

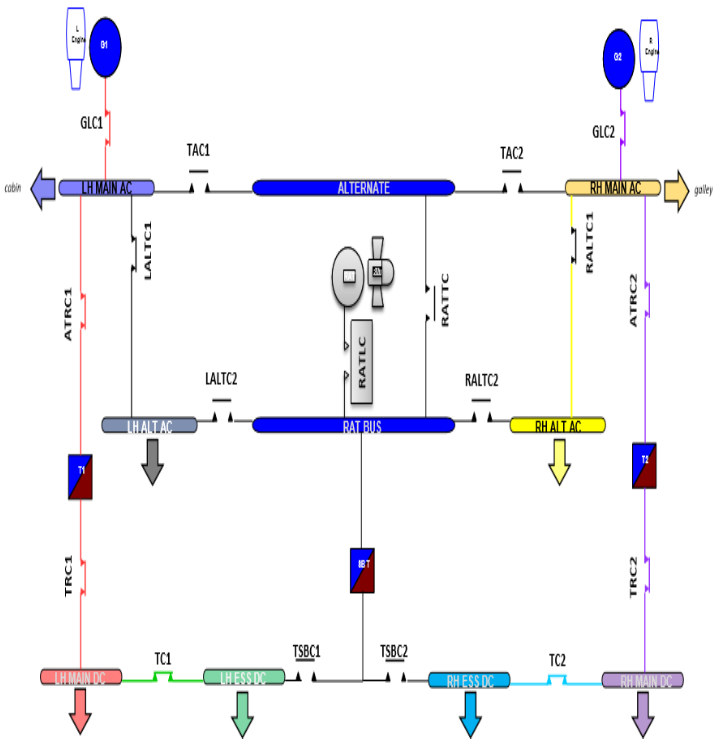

Figure 1. Aircraft electrical generation and distribution architecture

5.2 Control system description

To ensure the connection between the generators and the bus bars, the contactors must be connected or disconnected. To do this, they are controlled by means of processing programs running on calculators.
A calculator includes:
- a power supply unit (PSU),
- a microcontroller (μC).

A μC is a kind of processing unit on which are implemented the programs controlling the contactors. It interfaces with the environment via control ports.
It should be noted that for technological reasons the routing of contactor commands must respect the following property: a contactor can only be controlled by one and only one control port (P1).
It should be noted too that a PSU may fail frequently. Therefore a calculator can fails just like any other hardware equipment.
The role of the control system is twofold:
- It has to distribute electrical power to the bus bars by establishing distribution channels;
- It has to change the distribution channel in case of failure.

So we will have two kinds of μC programs; those controlling the distribution channels and those monitoring the switch from one channel to another.

Distribution channels are therefore made up not only of contactors but also of their treatments. In addition to this we have to consider transition channels made up of switching treatments.

5.3 Adressing safety requirements

Each bus bar can be seen as a system distributing power services to loads.
In normal operation, each bus bar of the PDS functions normally and distributes power via a so-called *normal distribution channel*.
If a power service is critical for the survival of the aircraft, then additional channels are added to the related system in order to guarantee service in the event of a failure. In this case a bar bus system will be composed of a normal distribution channel, a safety distribution channel and the related transition channel.
It is important to note that these safety channels are the result of preliminary safety studies. Therefore they are functional inputs of the problem we address.
In the following we will only consider the safety channels defined for single failures. Other safety channels exist in the case of multiple faults but there are out of the scope of the present paper.
In the case of single failures, the overarching requirement is as follows:
(R) In the case of a single failure occurring on equipment all bus bars must continue to be powered after reconfiguration of the system.
In this respect, it should be remembered that calculators are part of this equipment. Therefore calculator failures have to be considered.
In our case study, *R* means that the height bus bar system will have the same safety pattern made of: a normal distribution channel, a safety distribution channel and a transition channel.
To this pattern we must add the segregation constraints in order to avoid the hardware common points.
In an obvious way, the normal and safety distribution channels have to be segregated including the programs controlling the distribution.
Less obviously the normal and transition channels have to be segregated too. Indeed, if the program controlling the normal distribution and the transition program are on the same calculator then a calculator failure prevents switching to the safety distribution channel. The Figure 2 resumes the resulting pattern.

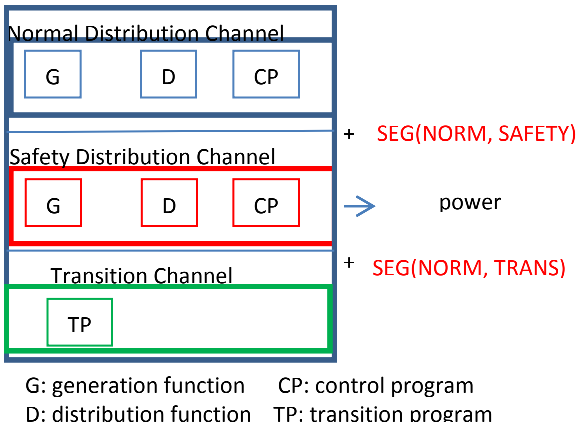

Figure 2. Example of safety pattern

Let's take as an example the LH MAIN AC system.

In the topology of Figure 3, the "LH MAIN AC" bus bar is powered by generator G1 via contactor GLC1. In the event of a single failure on generator G1, contactors TAC1 and TAC2 must be activated to allow continuous power supply to the bus bar via generator G2.

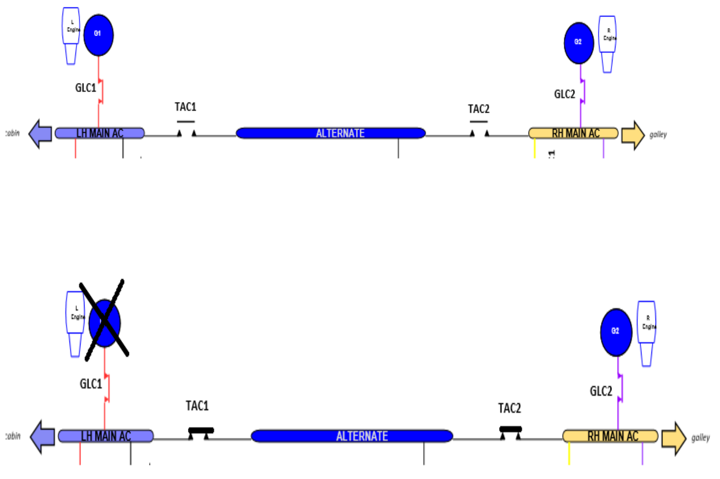

Figure 3. Example of normal mode and simple failure

Functionally, the LH MAIN AC system (see Figure 4) is composed of:
- A channel norm: G1, GLC1 and the control program,
- A channel safety: G2, GLC2, TAC2, TAC1 and the control program,
- A transition channel: the transition program.

Figure 4 shows a material deployment of LH MAIN AC.

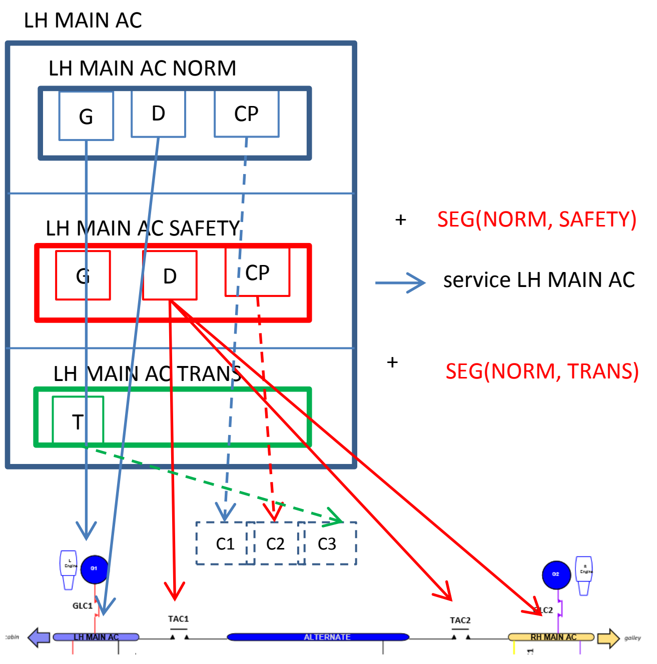

Figure 4. LH MAIN AC generation and distribution functions deployment

It should be noted that:
- For the generation (G) and distribution (D) functions of LH MAIN AC the deployment is known (NORM on G1 and GLC1, SAFETY on G2 and GLC2, TAC2, TAC1);

- For the control and transition programs the calculators and the ports used for the deployment are not known.

This is why in the following we put the emphasis on the deployment of the control and transition programs on calculators and on the routing between calculator control ports and contactors.

5.4 *Software architecture synthesis*

Thus, with a fixed electrical architecture, it remains the problem of determining:
- the necessary and sufficient number of calculators,
- how to allocate calculators to the control and transition programs,
- how to allocate the contactor commands to the calculator ports.

This last point requires specifying the relationships existing between contactor commands and contactors:
- R1: a contactor is controlled by one and only one contactor command of a calculator
- R2: a contactor command of a calculator controls one and only one contactor

All this has to be done in such a way that all safety requirements are met.

Finally it is not a question here of verifying that an existing control architecture verifies a posteriori the dependability requirements imposed on the electrical architecture, but rather of producing a control architecture that verifies by construction the dependability requirements. In other words, it is about to use the dependability requirements to build a solution. We are thus faced with a synthesis problem, requiring a formal model of the problem to be solved as well as a solving tool adapted to use this formal description to generate a correct architecture by construction [Pinto et al, 2010].

6 DEPS LANGUAGE

6.1 *Paradigm*

The Design Problem Specification (DEPS) language is commonly referred to as a Domain Specific Language (DSL). The target application domain is the specification and resolution of engineering problems, particularly those encountered in product or system design: sizing, configuration, allocation and architecture generation. The industrial interest is to use a unique formalism and tooling to model and solve all these categories of synthesis problems [Yvars, and Zimmer, 2021]..

DEPS is an external (as opposed to internal or embedded) dedicated language. The source code is therefore independent of any host generalist language.

The DEPS language can be seen as a combination of a software or system modeling language and a mathematical programming language. From the former have been borrowed the features of structuring and abstraction which allow to represent the elements and the system under study. From the latter were borrowed the mathematical concepts necessary to solve the engineer's problems: unknowns, equations and inequalities.

This combination makes it possible both to represent design problems and to pose and then solve or optimize the systems of equations and inequalities that govern them [Yvars and Zimmer, 2019]. DEPS is supported by the non-profit organization DEPS Link (www.depslink.com).

DEPS has been used on problems of robot design [Yvars and Zimmer, 2014], battery synthesis [Diampovesa et al, 2020] and embedded avionics system synthesis [Zimmer et al, 2020].

6.2 *Main characteristics*

6.2.1 The Model

The fundamental feature of the language is the Model. Any Model is defined using the keyword *Model* followed by its name and its (possibly empty) list of arguments. It contains in order: a declaration-definition area for *Constants*, a declaration area for *Variables*, a declaration-creation area for *Elements* and a definition area for *Properties*. The definition of a DEPS Model ends with the keyword *End* (Figure 5).

The properties of a Model are the equations and inequalities that relate to the constants and variables of that Model. All the algebraic operators of the IEEE754 standard are available to build the properties. A Model therefore contains all the ingredients necessary to set the constraints that govern an instance of this *Model*. Some specialized constraints can also be set as constraints on data catalogs. Any instance of a Model will necessarily contain the set of constants, variables and Elements expressed in the Model and will necessarily have to verify the set of properties of the Model..

```
Model B(arg1, arg2) extends A
Constants
arg1 : Integer ;
Variables
CPU : CpuIndex;
Elements
arg2 : C[Integer, CpuIndex];
Elt1 : C(1, 2);
Properties

End
```

Figure 5. DEPS model example

As in an object language there is inheritance and composition: a Model can be extended and inherits constants, variables, elements and properties from another Model. Elements are instances of Models. They are either built inside the Model and in this case you have to call the constructor of the reference model with values given to its arguments or are passed as arguments to a Model creating an aggregation link with it. In this last case, the argument elements are named in the list of model arguments and declared in the Elements zone, specifying the signature of the model to which they refer. *Constants* can also be passed as arguments to a *Model*, allowing the creation of Parametric Models. DEPS also supports polymorphism.

Thus in Figure 5, we have a Model B that inherits (*extends*) from a Model A. Model B has two arguments: argument *arg1* which is an integer (*arg1: Integer;)* and argument *arg2* which is an instance of Model C which must necessarily have two arguments: the first being an integer, the second being a *CpuIndex*. All this is specified by the signature of the Model C (*arg2: C [Integer, CPuIndex] ;*). Finally, Model B is composed of an instance of Model C called *Elt1* and created thanks to the call *C(1,2).*

Modeling a problem to be solved is therefore like as building DEPS Models and assembling elements in a main model without arguments expressed using the keyword *Problem*.

6.2.2 The Quantity

In DEPS, both constants and variables are associated with types of physical or technological magnitudes called quantities (Quantity). They are necessary in Systems Engineering (Figure 6).
A Quantity has:
- A basic type of quantity (called QuantityKind). For example, Real, Integer, Length;
- A min (resp. max) bound that represents the minimum (resp. maximum) value that can be taken by any constant or variable having the defined quantity as its type;
- A unit of the quantity. For example the meter *m* for a length and *u* for quantities without unit.

A QuantityKind has:
- a basic type (integer or real),
- a min and max bound,
- a dimension in the sense of the dimensional analysis of the quantity. For example L for a length or U for a dimensionless quantity;

```
QuantityKind Integer
 Type : integer ;
 Min  : -maxint;
 Max  : +maxint;
 Dim : U ;
End
```

```
Quantity CpuIndex
 Kind : Integer ;
 Min  : 1 ;
 Max : 4 ;
 Unit : u ;
End
```

Figure 6. QuantityKind and Quantity

6.2.3 Implementation

The DEPS Studio integrated modeling and solving environment associated with the DEPS language includes model editing functions, project management functions based on a package mechanism, a compiler and a solver.
An integration approach rather than a model transformation approach has been chosen. Indeed, in the case of solving a sub-defined system synthesis problem, it will be necessary in case of unsatisfactory computation results to perform a model tuning in the DEPS language. By deliberately opting for an integration approach, we choose this fine-tuning process.
The computational methods we use are taken from the work on the resolution of CSP [Tsang, 1993]. The solver implements the well-known revised HC4 (HC4Rev) propagation method [Benhamou et al, 1993] on equations and inequalities. We have extended the HC4Rev algorithm, initially designed to handle open real intervals, to the following three types of domains: integer intervals, enumerated sets of floating values and enumerated sets of signed integer values.
The object-oriented architecture of the solver has been designed so that it can be extended to other propagation and/or resolution methods.

7 MODELING THE PROBLEM IN DEPS

We briefly present here some models extracted from the synthesis problem explained in the chapter 5. They concern the model of the LH MAIN AC system and they have been chosen to highlight some technical points.

7.1 Modeling the subdefinite systems

The LH MAIN AC system follows this model (Figure 7). We recognize the channels NORM, SAFETY and TRANS and the segregation constraints between NORM and SAFETY and NORM and TRANS.
The channels NORM and SAFETY have the set of contactors as parameters since they act on them.

```
Model S1 () extends ThreeChannelSystem[ContactorSet]
Constants
Variables
Elements
ch1: ChS1Norm(ContSet);
ch2: ChS1Safety(ContSet);
ch3: TRANSChannel();
seg1 : SEG(ch1, ch2);
seg2 : SEG(ch3, ch1);
Properties
End
```

Figure 7. DEPS LH MAIN AC model

7.2 Modeling a channel

This model of channel (Figure 8) is composed of a generation function, a distribution function and a control program for commanding contactors belonging to the set of contactors.

```
Model ChS1Norm () extends GDChannel[ContactorSet]
Constants
Variables
Elements
GenF:   S1NormGenFunction();
DistF:   S1NormDistFunction (ContSet);
ProcF:  S1NormProcFunction(ContSet);
Properties
End
```

Figure 8. DEPS channel model

7.3 Modeling a function

This control function (Figure 9) commands one contactor. It extends a processing function with a constraint in the property zone which expresses that the control program and the contactor control are on the same calculator.

```
Model S1NormProcFunction() extends
OneContactorProcFunction[ContactorSet]
Constants
Variables
Elements
Properties
L1.ProcIndex = ContSet.GLC1.ProcIndex;
End
```

Figure 9. DEPS function model

7.4 Modeling a segregation

This model segregates two control functions (Figure 10). The first one commands one contactor and the second three contactors. The second function is composed of three logic subprograms each of them controlling a single contactor.

```
Model SEG(ProcF1, ProcF2) extends
SEG[ProcFunction[ContactorSet], ProcFunction[ContactorSet]]
Constants
Variables
Elements
ProcF1 : OneContactorProcFunction[ContactorSet];
ProcF2 :  ThreeContactorProcFunction[ContactorSet];
Properties
ProcF1.L1.ProcIndex <> ProcF2.L1.ProcIndex;
ProcF1.L1.ProcIndex <> ProcF2.L2.ProcIndex;
ProcF1.L1.ProcIndex <> ProcF2.L3.ProcIndex;
End
```

Figure 10. DEPS segregation of two control functions

All the segregation requirements are expressed in the bar bus systems (Fig 7).

7.5 Modeling the problem

Finally, we create the full problem containing all the contactors of the electrical architecture (TheContactors), the eight bus bars (F1, F2, ..., F8) as well as complementary segregations (seg1, seg2, seg3, seg4) necessary for the complete expression of the problem and allowing to express the segregation of the NORM channels of the bus bars between the left and side parts of the aircraft (Fig 11).

```
Problem FailSafeVerification
Constants
Variables
Elements
 TheContactors: ContactorSet();
  F1: S1(TheContactors);  F2: S1(TheContactors);
  F3: S1(TheContactors);  F4: S1(TheContactors);
  F5: S1(TheContactors);  F6: S1(TheContactors);
  F7: S1(TheContactors);  F8: S1(TheContactors);
  seg1: SEG(F1.ch1, F3.ch1);
  seg2: SEG(F2.ch1, F4.ch1);
  seg3: SEG(F5.ch1, F7.ch1);
  seg4: SEG(F6.ch1, F8.ch1);
Properties
End
```

Figure 11. DEPS model of the Problem

7.6 Solving the problem

After a compilation step, the DEPS solver produces fail-safe architectures with a control system requiring at the best four calculators.
The results of allocating the calculators to the contactors are shown in Table 1.
The results of the allocation of the calculators to the different contactor control programs for the normal and safety modes of the bar bus systems as well as their transition management programs are shown in Table 2.

Table 1. Routing between contactors and calculators

Contactors	Calculator Index
GLC1	1
ATRC1	1
TRC1	1
LALTC1	4
GLC2	2
ATRC2	2
TRC2	2
RALTC1	3
RALTC	1
RATTC	1
TAC1	3
TAC2	4
LALTC2	3
RALTC2	4
TC1	2
TSBC1	3
TSBC2	3
TC2	1

Thus, if we remember that for the LHMAIN AC bus bar, the NORM channel has to control contactor GLC1 (Fig. 3), we read in table 2 that the control program for this contactor is deployed on calculator 1. In the same way, the contactors to be actuated for the SAFETY channel are contactors TAC1, TAC2 and GLC2 which control programs are deployed on calculators 3, 4 and 2. The transition program TRANS is deployed on computer 3.
The whole is coherent since we can see in table 1 that contactor GLC1 is routed to calculator 1 and that TAC1, TAC2 and GLC2 are routed respectively to calculators 3, 4 and 2.

Table 2. Deployment of control and transitions programs on calculators

Bus bar system	NORM calculator Index	SAFETY calculator Index	TRANS calculator Index
LHMAIN AC	1	3, 4, 2	3
LHMAIN DC	1, 1	4, 3, 3, 2	2
RHMAIN AC	2	4,3,1	3
RHMAIN DC	2, 2	3, 4, 3, 1	1
LHALTAC	4	3, 1, 3	1
LHESSDC	2	4, 3, 3	1
RHALTAC	3	4, 1, 4	1
RHESSDC	1	3, 4, 3	2

8 CONCLUSION

In this article we have shown that it is possible to capture safety requirements that are difficult to formalize directly without a suitable formal language. We used the DEPS language which allowed us to model the right abstractions at the right level to describe a sub-defined model of architecture software architecture for the control of embedded electrical generation and distribution system for aircraft on the one hand, and safety requirement models on the other hand. The deployment problem was solved by applying constraint satisfaction methods to the properties of the problem with DEPS Studio environment. The structuring features offered by the language facilitate the reuse of the models developed.

Future work will focus on taking into account additional safety requirements as double faults and ultimate rescue mode and additional components as fault sensors. The evolution of the models will follow those of the DEPS language in particular on the possibilities of manipulation of collections of objects in the future versions of DEPS and DEPS studio.

9 REFRENCES

Giraud, X. (2014). *Méthodes et outils pour la conception optimale des réseaux de distribution d'électricité dans les aéronefs.* Phd Thesis INSA Toulouse.

Yang Z, Qu J &Shi X (2015). Modeling and Simulation of Power Distribution System in More Electric Aircraft. *Journal of Electrical and Computer Engineering*.

Menu, J., Nicolai, M., & Zeller, M. (2018.). Designing Fail-Safe Architectures for Aircraft Electrical Power Systems. *Proc of AIAA/IEEE Electric Aircraft Technologies Symposium.*

Becz, S., Pinto, A., Zeidner, L. E., Banaszuk, A., Khire, R., & Reeve, H. M (2010). Design System for Managing Complexity in Aerospace Systems. *13th*

AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.

Pinto A., Becz S. & Reeve H.M (2010). Correct-by-construction design of aircraft electric power systems. *Proc of AIAA Aviation Technology, Integration, and Operations Conference*.

Creff S, Le Noir J, Lenormand E & Madelénat S. (2020). Towards Facilities for Modeling and Synthesis of Architectures for Resource Allocation Problem in Systems Engineering. *Proc of 24th Systems and Software Product Line Conference*. Montreal.

Bąk K., Diskin Z., Antkiewicz M., Czarnecki K. & Wąsowski A. Clafer: Unifying class and feature modeling. *Software and Systems Modeling, 2014.*

Lorca X., Prud'homme C. & Fages J-G (2014). *Choco3 Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S.

Kang, K.C., Cohen S.G., Hess J.A., Novak, W.E. & Peterson, A.S. (1990). *Feature-oriented domain analysis (FODA) feasibility study*. Technical Report CMU/SEI-90-TR-021, SEI, Carnegie Mellon University.

Leserf P., Saqui-Sannes P.,Hugues J. & Chaaban K. (2015). SysML Modeling For Embedded Systems Design Optimization: A Case Study. *Proc of 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD).*

Shah A.A. (2010). *Combining mathematical programming and SysMl for component sizing as applied to hydraulic systems*. Master Thesis, Georgia Institute Of Technology.

Shah A.A., Paredis C.J.J., Burkhart R. & Schaefer D. (2012). Combining Mathematical Programming and SysML for automated Component Sizing of Hydraulic Systems. *Journal of Computing and Information Science in Engineering (JCISE).*

Yvars, P.-A. & Zimmer, L. (2014). DEPS Un langage pour la spécification de problèmes de conception de Systèmes. *Proc of the 10th International Conference on Modeling, Optimization & SIMulation (MOSIM).* Nancy.

Yvars, P.-A. & Zimmer, L. (2019). DEPS Studio : Un environnement intégré de modélisation et de résolution de problèmes de conception de systèmes. *Proc of 8ème Conférence en ingénierie du logiciel (CIEL)* Toulouse.

Yvars, P-A & Zimmer, L (2021). Integration of constraint programming and model based approach for design synthesis. *Proc of IEEE Systems Conference SYSCON 2021*, Vancouver.

Diampovesa S, Hubert A & Yvars P-A (2020).Modélisation d'un problème de conception en vue de réutilisabilité. Exemple d'une batterie Li-ion. *Proc of Symposium de Génie Electrique (SGE),* Nantes.

Zimmer L, Yvars P-A & Lafaye M (2020). Models of requirements for avionics architecture synthesis: safety, capacity and security. *Proc of the 11th Complex System Design and Management (CSDM) conference.*

Tsang E (1993). *Foundations of Constraint Satisfaction.* London and San Diego: Academic Press.

Benhamou F., Goualard F., Granvilliers L. & Puget J.F. (1993). Revising Hull and Box consistency. *16th International Conference on Logic Programming.*