DALculus – Theory and Tool for Development Assurance Level Allocation

Pierre Bieber[1] , Rémi Delmas[1], Christel Seguin[1]
[1] ONERA, 2 avenue Edouard Belin,
31055 Toulouse, France
{Pierre.Bieber, Remi.Delmas, Christel.Seguin} @ onera.fr

Abstract. The Development Assurance Level (DAL) indicates the level of rigor of the development of a software or hardware function of an aircraft. We propose a theory formalizing the DAL allocation rules found in the ARP4754a recommended practices. A tool implementing this theory was developed in order to assist the safety specialists when checking or optimizing a DAL allocation.

Keywords: Dependability Assessment, Aerospace Systems, Avionics

1 Introduction

The Development Assurance Level (DAL) indicates the level of rigor of the development of a software or hardware function of an aircraft. The DAL guides the assurance activities that should be applied at each stage of development. These activities aim at eliminating design and coding errors that would have a safety effect on the aircraft.

The revised version of Aeronautical Recommended Practices ARP4754a [1] establishes rules to allocate the DAL to functions. The allocation is primarily based on the severity of the effects of a function implementation error. But new rules introduce the possibility to downgrade the DAL levels based on the independence at requirement, implementation or hardware deployment level.

It can be tedious to check that downgrading rules were applied correctly. Furthermore, designers are trying to allocate the smallest DAL possible to functions in order to decrease development costs. Consequently, we have investigated means to assist the safety specialists when checking or optimizing a DAL allocation.

We undertook the formalization of DAL allocation rules as constraints, that link the maximal allowed reduction of DAL and the independence of functions appearing in the minimal combination of function errors leading to an aircraft function loss. Optimization criteria are also defined to help minimizing the allocated DAL and the number of functions required to be independent.

This problem is solved using very efficient constraint solvers. A valid DAL allocation and a set of function independence requirements are extracted from the

solution found by the solver and are proposed to the designers. The approach also takes into account user provided constraints that help the tool to focus on more interesting allocations.

The following chapter describes the Development Assurance Level Allocation Process according to ARP4754a. Then, in chapter 3, we detail the proposed theory of DAL allocation. Finally, we explain how the approach is implemented and the tool was applied on various critical aircraft systems including the Electrical Generation and Distribution System. The paper concludes with the first lessons learnt from these experimentations.

2 Development Assurance Level Allocation Process

2.1 Aims of the DAL

The design of aeronautics safety critical systems deals with two families of faults:

- random faults of equipments, such as an electrical wire rupture that would cause the loss of power supply for a computer cabinet,
- systematic faults in the development of the equipment, which include errors in the specification, design and coding of hardware and software. An instance of a development fault could be a specification of the "wheel on ground" function that would not make a difference between the landing gear being compressed or decompressed resulting in a incorrect indication of whether the aircraft is on ground or flying.

Two very different approaches are used when assessing whether the risk associated with these two types of faults is acceptable. Quantitative requirements (thresholds of fault occurrence probabilities) are associated with random equipment faults whereas non-quantitative design assurance requirements (Development Assurance Level) are associated with development faults. This approach is not unique to the aeronautics industry, it also applied in other safety-critical domains such as space, nuclear, railway or automotive industries [2].

In the aeronautics industry, a DAL ranging from E to A (where A is the higher level) is allocated to functions, software and hardware items. According to the ARP4754, "*The Development Assurance Level is the measure of rigor applied to the development process to limit, to a level acceptable for safety, the likelihood of Errors occurring during the development process of Functions (at aircraft level or system level) and Items that have an adverse safety effect if they are exposed in service.*"

The DAL associated with a software item guides the assurance activities that have to be performed during its development following DO178B [3]. The higher the DAL, the more detailed and rigorous are the assurance activities to be performed. For instance, the following table describes three objectives of the Software Coding and

Integration Process. It indicates which objectives are applicable at a given DAL level. A cell containing R means that this is a Required objective at this level, a blank cell means the objective is not required and a cell containing I means that the objective should be achieved with independence. Independence is achieved when the activity is performed by a team different from the software development team.

Table 1. DO178B Software Coding Objectives (extract).

Id	Objective	DAL Applicability			
		A	**B**	**C**	**D**
1	Source Code complies with low-level requirements	I	I	R	
2	Source Code complies with software architectures	I	R	R	
3	Source code is verifiable	R	R		

At level E nothing is required. At level D none of these three objectives has to be achieved. Traceability at this level of detail (between source code and requirements) is not needed, but other traceability (between high-level and system requirements) is required. At level C, objectives 1 and 2 are required without independence. Independence is almost never required at level C. At levels A and B, the three objectives have to be fulfilled. The difference between these two levels is that more objectives shall be achieved with independence at level A than at level B.

High DALs require a great number of assurance activities. The increase in the level of rigor, level of detail and the need to involve independent teams increase the development cost of software and hardware items. Consequently, designers aim at allocating a DAL to software and hardware as low as possible, within the bounds imposed by safety regulation, in order to reduce the development cost of their systems.

2.2 DAL Allocation rules according to ARP4754a

DAL allocation is a part of the System Safety Assessment Process which comprises several steps. First, the Functional Hazard Analysis identifies the Failure Conditions (e.g. safety critical situations of the system) and assesses their severity on a scale going from No Safety Effect (NSE) to Catastrophic (CAT). Then, during the Preliminary System Safety Assessment, fault-trees (or alternatively fault propagation models [4]) are built and analyzed. A fault-tree describes the way that item faults propagate inside the system architecture in order to cause a Failure Condition. Minimal combinations of item faults leading to the Failure Conditions are extracted from the fault-tree, theses combinations are called Minimal Cut Sets (MCS). These combinations are used to compute the mean probability of the Failure Condition in

order to assess whether the designed architecture is safe enough. Usually the mean probability of a Failure Condition whose severity is Catastrophic shall be smaller than 10^{-9} per flight-hour.

DAL allocation is based on a qualitative assessment of the minimal cut sets computed for the Failure Conditions of the system. We consider that an item fault contributes to a Failure Condition if it appears (through its failure modes) in one of the MCS of the failure condition. A DAL is associated with an item according to the classification of the most severe Failure Condition that this item fault contributes to. Actually, due to DAL downgrading rules, it is not necessarily the most severe FC that constrains the DAL allocation the most.

Table 2 gives the basic DAL assignment rules. **Sev** is the severity of the most critical Failure Condition an item fault is involved into. According to this table, an item is assigned DAL B if the most severe consequence of this item fault is classified as Hazardous (HAZ).

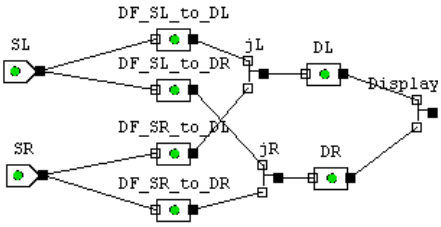
Table 2. Basic DAL Allocation.

Sev	NSE	MIN	MAJ	HAZ	CAT
DAL	E	D	C	B	A

New DAL allocation rules introduced in the revised ARP4754a allow to downgrade the original DAL allocated using the basic allocation rule, in cases when items involved in the minimal cut sets are known to be pair-wise independent (we will discuss in detail the notion of independence in section 2.3). Each minimal cut set is analyzed using one of the 2 following downgrading options:

- **Option 1**: The original DAL of one item in the minimal cut set is preserved and the DAL of remaining items may be downgraded of at most two levels.
- **Option 2**: The original DAL of one pair of items in the minimal cut set may be downgraded of at most one level and the DAL of remaining items may be downgraded of at most two levels.

Once all minimal cut sets are analyzed the lowest DAL consistent with the authorized downgrading by the selected option may be allocated to the items.



1. DL, DR
2. SL, SR
3. DF_SL_to_DL, DF_SL_to_DR, SR
4. DF_SL_to_DL, DF_SR_to_DL, DR
5. DF_SL_to_DL, DR, SR
6. DF_SL_to_DR, DF_SR_to_DR, DL
7. DF_SL_to_DR, DL, SR
8. DF_SR_to_DL, DF_SR_to_DR, SL
9. DF_SR_to_DL, DR, SL
10. DF_SR_to_DR, DL, SL
11. DF_SL_to_DL, DF_SL_to_DR, DF_SR_to_DL, DF_SR_to_DR

Fig. 1. Data Display (Graphical view and Minimal Cut Sets)

Example: To illustrate the various DAL allocation rules, we introduce a very basic Data Measurement and Display example that is graphically described in figure 1. Data is measured by two sensors SL (Left sensor) and SR (Right Sensor) anddisplayed on two Display units DL and DR , measured data is sent by the sensors to both displays (via data flows named DF_SL_to_DL, DF_SL_to_DR, DF_SR_to_DL and DF_SR_to_DR). Each of these items may be lost. Measured data can be displayed on one display unit as long as the display unit is working and data measured by at least one sensor is transmitted to the display. The Failure Condition of interest is the loss of data on both displays, and it is classified Hazardous.

Figure 1 also provides the list of the 11 Minimal Cut Sets leading to the loss of data on both displays. MCS 1 states that the loss of both displays leads to the failure condition, and MCS 11 states that the loss of 4 data flows leads also to the failure condition.

As the Failure Condition is classified Hazardous, the original DAL allocated to each of the items of the Data Measurement and Display system is equal to B.

Downgrading rules can be applied if we suppose that the groups of Left items (SL, DL, DF_SL_to_DL, DF_SL_to_DR) and Right items (SR, DR, DF_SR_to_DL and DF_SR_to_DR) are mutually independent. We can check that each minimal cut set of each FC contains at least a pair of faults from items that are mutually independent. For instance MCS 1 contains the faults of items DL and DR that are mutually independent. Hence, if we apply option 1 then we may allocate DAL B to DL and downgrade the DAL of DR to D or, conversely, downgrade the DAL of DL and allocate DAL B to DR. After analyzing all the minimal cut sets, allocating DAL B to all Left items and DAL D to all Right items would be allowed by option1.

If we now apply option 2 and we analyze MCS 1 then the DAL of both DL and DR may be downgraded to C. When MCS 11 is analyzed, as DF_SL_to_DR and DF_SR_to_DL are independent their DAL may be downgraded to C and the DAL of remaining items DF_SL_to_DL and DF_SR_to_DR may be downgraded to D. When MCS 10 is analyzed, as DL and SL are not independent but DL and DF_SL_to_DR are independent, the DAL of DL and DF_SL_to_DR may only be downgraded to C. Consequently, there is a conflict between the analysis of MCS 11

and MCS 10 with respect to the downgrading of the DAL of DF_SL_to_DR. To solve this conflict, the higher DAL allocation has to be used. Here, DAL C should be allocated to DF_SL_to_DR .

But, if we go back to the analysis of MCS 11, it was also possible to downgrade the level of DF_SL_to_DR and DF_SR_to_DL to D and the level of DF_SL_to_DL and DF_SR_to_DR to C. In that case the analysis of MCS 10 and 11 would not be conflicting.

As it was shown by this simple example, verifying that a DAL allocation is correct is not an easy task, and generating a correct DAL allocation is even more difficult when systems become large. For each minimal cut set there are several downgrading possibilities depending on the selected option. Furthermore, the downgrading possibilities have to be consistent for all minimal cut sets. We think that a DAL allocation and verification tool would be very helpful for the safety analyst. The first step in developing such a tool is to formalize the principles of DAL allocation.

2.3 Item Independence according to ARP4754A

Independence plays a crucial role in the DAL allocation process as it is required in order to apply DAL downgrading. According to the ARP4754a "*item Development Independence ensures that the development of items on which the Function(s) is(are) implemented, should not suffer from a common mode Error.*"

Item Development Independence is achieved by relying on items that use different types of software and hardware technologies to implement a Function. For instance, a general-purpose computer-based LCD screen and an ad-hoc electro-mechanical display could be used to display the fuel quantity on-board. These two set of items would be considered as independent as they do not depend on similar technologies. On the contrary, several occurrences of a screen with the same technology (LCD screen for instance) could be considered as non-independent.

From a system development perspective, the use of independent items increases the development costs. In some cases, it might just be impossible to achieve item independence as only one technology is available to implement the items. So system designers will be interested by architectures with a limited use of independent items.

Example: Let us consider again the Data Measurement and Display system studied in the previous section. We now suppose that the four data flows of the Data Measurement and Display System are no longer mutually independent because the same communication protocol is used to transmit them. In that case, when MCS 11 is analyzed, we cannot downgrade the DAL level of the data flows as they are not independent. Consequently DAL B would be allocated to data flows (DF_SL_to_DL, DF_SL_to_DR, DF_SR_to_DL and DF_SR_to_DR)

As it was shown by the previous example, various assumptions on item independence lead to very different DAL allocation. This makes the DAL allocation even more

complex. So in order to assist the DAL allocation process we should also help the safety analyst to find which items have to be independent.

3 DAL Allocation as a Constraint Satisfaction Problem

We formalize DAL allocation as a sequence of two constraint satisfaction problems. The first problem consists in identifying a minimal set of necessary independence relations between items. The second problem allocates the DAL to items taking into account the various downgrading options enabled by the item independence relations identified by solving the first problem.

3.1 Independence Identification

Besides the classical quantitative safety requirement, a qualitative safety requirement is also usually associated with a failure condition. This qualitative requirement could be of the form "*no combination of item faults of size strictly less than NSev leads to the failure condition*", where the relation between NSev and the severity of the failure condition is given by Table 2.

To check the qualitative safety requirements associated with a failure condition, we want to establish that the size of each minimal cut set is greater than or equal to NSev. The size of an MCS including N item faults is not necessarily equal to N because these item faults could be non-independent. If none of the items appearing in the MCS are independent, a single common cause fault could lead to the failure condition. According to Table 2, this would not be acceptable for failure conditions whose severity ranges from MAJ to CAT. We consider that the size of a minimal cut set is equal to the cardinal of the maximal subset of mutually independent items contained in the minimal cut set. So for a MAJ or HAZ failure condition, each minimal cut set should contain at least a pair of independent items. For a CAT failure condition each minimal cut set should contain at least three items that are mutually independent. We name this maximal subset the ***core*** of the MCS from now on.

Table 2. Qualitative Safety Requirements.

Sev	MIN	MAJ	HAZ	CAT
NSev	1	2	2	3

Example: According to the definition of the size of a minimal cut set, MCS 1 of the Data Measurement and Display system is of size 2 if `DF` and `DL` are independent. MCS 11 is of size 4 if the four data-flows are mutually independent but it is of size 2 if only `DF_SL_to_DL` and `DF_SR_to_DR` are independent. Both independence relations would be acceptable as the failure condition if HAZ.

In the following we describe the Constraint Satisfaction Problem (CSP) that has to be solved in order to identify independence relations between items so that qualitative requirements are enforced. The CSP is defined by a set of constants that are inputs of the problem such as the set of minimal cut sets, a set of variables whose values are searched by the solver, and by the constraints that should be satisfied on these variables. We also use abbreviations to make the constraints more readable. Finally we propose a quantitative criterion that is optimized by the solver during its search.

• Constants
– I: set of items,
– MCS: set of minimal cut sets for the failure condition,
– Nsev: minimal size required for minimal cut sets to satisfy the qualitative requirement.

•Variables
–indep(p,p') is true if items p and p' are independent,

• Abbreviations

- The size of a minimal cut set mc is greater than or equal to n iff there exists a set of n items in mc that are mutually independent:

$(size(mc) \geq n) == (\exists m \subseteq mc, card(m)=n \ \& \ indepm(m)))$

- Items in the set m are mutually independent iff all pairs of distinct items (p,q) in m are independent.

$indepm(m) == \forall (p,q):m*m, \ p \neq q \Rightarrow indep(p,q)$

• Constraint

- The size of each minimal cut set shall be greater or equal to Nsev:

$\forall mc:MCS, \ size(mc) \geq Nsev$

• Optimization Criterion
Minimize the number of pairs of independent items:

$\Sigma_{(p,q):I*I} \ indep(p,q)$

Example: Let us consider again the Data Measurement and Display system. A minimal and valid independence relation found by the solver has cardinal 7 :

```
indep(DR, DL),
indep(SL, SR)
indep(SR, DF_SL_to_DR)
indep(SL, DF_SR_to_DL)
indep(SL, DF_SR_to_DR)
indep(DR, DF_SL_to_DL)
indep(DF_SL_to_DR, DF_SR_to_DR).
```

3.2 DAL Allocation

This section describes the CSP that formalizes the DAL allocation problem. Table 3 shows the correspondence between DALs and numerical values used to encode the order relation between DALs and simplify the expression of constraints.

Table 3. Numerical values for the DAL.

NDAL	0	1	2	3
DAL	D	C	B	A

• Constants
– `I`: set of items,
– `MCS`: set of minimal cut sets,
– `indep`: independence relation,
– `NDAL`: numerical value of the DAL of FC

•Variables
– `ndal(f):` numerical value of the possibly downgraded DAL of item f

•Constraints

- The DAL level of an item cannot be downgraded by more than two levels:

$\forall mc:MCS, \forall p:mc, ndal(p) \geq NDAL - 2$

- **option 1.** For each MCS and each of its items, either the DAL is not downgraded, or the DAL is downgraded. In this case a core of mutually independent items of size NSev must exist in the considered MCS such that one of its elements has at least the DAL NDAL, and the downgraded item must be part of this core:

$$\forall mc:MCS, \forall p:mc,$$
$$ndal(p) \geq NDAL \text{ or}$$
$$(ndal(p) < NDAL \Rightarrow$$
$$(\exists m \subseteq mc, indepm(m) \,\&\, card(m) = NSev \,\&\, p:m \,\&$$
$$(\exists q:m, ndal(q) \geq NDAL))$$

- **option 2**. For each MCS and each of its items, either the DAL is not downgraded, or the DAL is downgraded. In this case a core of mutually independent items of size NSev must exist in the considered MCS such that two of its elements have at least the DAL NDAL-1, and the downgraded item must be part of this core.

$$\forall mc:MCS, \forall p:mc,$$
$$ndal(p) \geq NDAL \text{ or}$$
$$(ndal(p) < NDAL \Rightarrow$$
$$(\exists m \subseteq mc, indepm(m) \,\&\, card(m) = NSev \,\&\, p:m \,\&$$
$$(\exists (q,r):m*m, q \neq r \,\&\, ndal(q) \geq NDAL-1 \,\&\, ndal(r) \geq NDAL-1))$$

•Criterion: Minimize the sum of numerical values of allocated DALs:

$\Sigma_{f\,:\,I}$ ndal(f)

Example: We consider the minimal cut sets of the Data Measurement and Display system and the `indep` relation presented in the previous section. If we apply option 1, an optimal DAL allocation is:

```
dal(SR)= dal(DL)= D
dal(SL)= dal(DR)= B
dal(DF_SL_to_DR)= B
dal(DF_SR_to_DL)= dal(DF_SL_to_DL)= dal(DF_SR_to_DR)= D
```

If we apply option 2, an optimal DAL allocation is:

```
dal(SR)= dal(DL)= C
dal(SL)= dal(DR)= C
dal(DF_SL_to_DR)= C
dal(DF_SR_to_DL)= dal(DF_SL_to_DL)= dal(DF_SR_to_DR)= C
```

4 Tool Support and Experimentations

4.1 Pseudo-Boolean Constraint Solving

The two constraint satisfaction problems presented in the previous section can be solved very efficiently by solvers such as Sat4J [6] or WBO [7] that deal with pseudo-Boolean logic (also known as {0,1} linear integer constraints). In this section we explain how the DAL allocation problem is modeled using pseudo-Boolean logic.

In pseudo-boolean logic, the DAL level allocation is encoded by a predicate hasDal(p, i) which is true if and only if ndal(p)>=i. Let Subsets(mc, Nsev) denote the collection of all possible subsets of size Nsev of the minimal cut set mc, called the Nsev-subsets of mc. We introduce the predicate dalOk(mc, s) to represent that the subset s of mc satisfies the chosen DAL allocation constraints.

For each MCS mc, there must be at least one of its Nsev-subsets such that the dalOk predicate holds. This is modeled in pseudo-boolean logic by instantiating the following constraint for each MCS, where {s1, ..., sn} = Subsets(mc, Nsev):

$$\text{dalOk(mc, s1)} + \ldots + \text{dalOk(mc, sn)} \geq 1;$$

For each item, the DAL is downgraded by at most two levels with respect to the default case :

$$\text{hasDal(f, NDAL-2)} \geq 1;$$

Listed below are the constraints instantiated for each Nsev-subset to encode its dalOk semantics when Nsev=2, for each possible DAL allocation options.

DAL option1 is satisfied for subset s = {p1, p2} in MCS mc if and only if:

- Items of the subset are mutually independent:

  $-1*dalOk(mc, \{p1, p2\}) + 1*indep(p1, p2) \geq 0;$

- One item in the subset has the Default DAL:

$-2*dalOk(mc, \{p1, p2\}) + 2*hasDal(p1, NDAL) + 2*hasDal(p2, NDAL) \geq 0;$

- Each item p3 in mc - {p1, p2} has the default DAL:

  $-1*dalOk(mc, \{p1, p2\}) + 1*hasDal(p3, NDAL) \geq 0;$

DAL option2 is satisfied by subset s = {p1, p2} in MCS mc if and only if:

- Items of the subset are mutually independent:

  $-1*dalOk(mc, \{p1, p2\}) + 1*indep(p1, p2) \geq 0;$

- A pair of items in the subset has at least the default DAL minus 1:

$-2*dal\_ok(p1, p2) + 1*hasDal(p1, NDAL-1) + 1*hasDal(p2, NDAL-1) \geq 0;$

- Each item p3 in c\{p1, p2} has the default DAL:

  $-1*dalOk(mc, \{p1, p2\}) + 1*hasDal(p3, NDAL) \geq 0;$

4.2 The DALculator

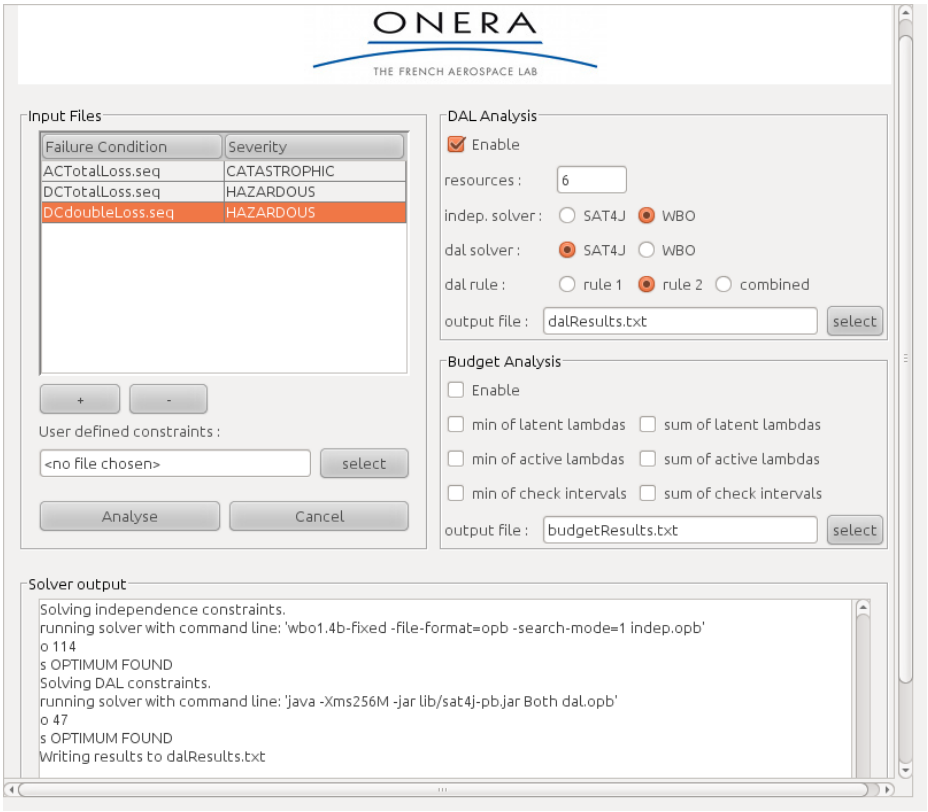


Fig. 2. Screen Capture of the DALculator GUI

A tool called the DALculator supporting independence identification and DAL allocation was developed. Figure 2 shows its graphical user interface. The user first selects one or several files containing the minimal cut sets of failure conditions then and indicates the failure condition severity. The user also selects the DAL downgrading option and the solver to be used. The tool parses the minimal cut sets and first generates a pseudo-Boolean instance encoding the independence identification problem and solves it. If a solution is found, the tool extracts the list of all mutually independent cores from it and generates a pseudo boolean instance encoding the the DAL allocation problem, and solves it using the chosen solver. If a solution is found, the DALculator proposes a DAL allocation to the user in a text file.

4.3 Lessons Learnt from the First Experimentations

We tested the DALculator on two classes of examples. The first class contains simple examples in the style of the Data Measurement and Display system. They were used in order to validate the results of the DALculator. The size of the minimal cut sets in this class ranges from 11 to 51, making it possible to check by hand that the results are correct. The second class contains more complex systems extracted from industrial models. This includes minimal cut sets computed for an Electrical Distribution and Generation system as well as a Flight Control system. These models were used in order to test the performance of the tool.

The first experiments showed that DAL allocation and independence proposed by the DALculator can be optimal but not very intuitive. For instance, in the Data Measurement and Display system we first hand-built a solution with 16 independent pairs, considering that all Left items are independent from the Right items. This is far from the optimal solution with only 7 independent pairs found by the tool and presented in section 3.1.

To help the user explore various independence relations and DAL allocations, the tool can load directives that will be taken into account in the constraint problem. So far, the user can specify that two items have to be independent, or that they shall not be independent. Similarly, the user can specify that the DAL of an item shall be upper bounded to a given level.

Thanks to these user directives it is also possible to use the DALculator to check that an existing allocation is correct. The user enters a complete independence relation or a complete DAL allocation and if the resulting constraints are consistent it means that the proposed allocation is consistent with the DAL allocation rules.

Table 4. Performances of DALculator.

Name	**MCS**	**order**	**Sev**	**Perfs**		**Idp**	**DAL**		
				Idp	**DAL**		**max**	**R1**	**R2**
DataDisplay0	11	4	HAZ	0.004s	0.004s	7	12	12	8
DataDisplay1	39	6	HAZ	0.011s	0.005s	7	9	9	9
DataDisplay2	51	9	HAZ	0.039s	0.007s	7	12	12	8
Elec	165	3	HAZ	0.007s	0.006s	16	18	15	11
Elec	447	4	CAT	0.060s	0.066s	102	70	60	56
FlightControl	3746	3	CAT	0.093s	0.445s	903	183	168	155

Table 4 gives performance figures on representative examples. The **MCS** column gives the number of cut sets of the considered failure condition. The **order** column gives the maximal cut order. The **Sev** column gives the classification of the failure condition. The **Perf** column gives the solving time for the independence constraint problem (**idp** column) and the DAL allocation problem (**DAL** column). Despite the combinatorial nature of the problem, the run times are relatively low, even on real world examples. This validates our choice of constraint formalism and solving

technology, and indicates a good scalability potential to real world problems. The **Idp** column gives the optimal number of independence requirements generated by the independence analysis. This number can be surprisingly lower than in trivial hand made solutions, which demonstrates the added value of optimization. Last, the **DAL** column lists the values of the criterion that is optimized on the **DAL** allocation analysis: the max column gives the worst value for a valid allocation encountered by the solver during optimization. **O1** and **O2** columns give the optimal solution computed by the solver for option1 and option2 respectively. Again, the quantitative distance between valid allocations, from the worst encountered to optimal, can be quite important, demonstrating the added value of automatic and formal analysis over hand built solutions.

As often in standards, there is room for interpretation of the rules. In the case of ARP4754A downgrading rules, we estimated that two notions were subject to interpretation:

- How many items in the minimal cut set have to be independent in order to apply the downgrading rules?
- What does the "*remaining items*" cover in the rules that state that "*all remaining items may be downgraded*"?

Our formalization proposes an interpretation of these two notions. We considered that the number of items that needs to be independent is defined by Table 2. And we considered that "remaining items" that can be downgraded are items that are independent from the non-downgraded item, other items shall not be downgraded. We plan to discuss with people in the aeronautics industry and in the certification authorities in order to check that our interpretation is valid.

5 Conclusion and Perspectives

Related work

In [8] several authors including Y. Papadopoulos have proposed an approach for the automated allocation of Automotive Safety Integrity Level (ASIL). The ASIL has a role similar to DAL. Their approach is quite different from ours as it computes the ASIL directly on the basis of the fault-tree structure. We have not considered this approach because we wanted to support safety assessment that use safety propagation models instead of fault-trees. Nevertheless this approach seems interesting to overcome one of the drawback of the approach based on minimal cut sets analysis. The number of MCS can become huge for complex systems and could cause problems for the constraint solvers that we use.

Another interesting feature of this work is that ASIL is allocated to an item fault and not to a function as for the DAL. Consequently different ASIL could be allocated to the loss of an item and to its undetected erroneous behavior. This fine grained allocation could help to select the level of rigor of development techniques; For instance, it could be interesting to use formal techniques to check the absence of

errors leading to a fault with a high SIL and use more conventional techniques when considering other faults of the item that were allocated a smaller SIL.

The paper does not describe whether the independence between items is taken into account in the allocation process. In the aeronautics safety process the identification of item independence is very important and cannot be neglected.

In [9] the author describes an approach to allocate the SoftWare Assurance Level SWAL. Again the SWAL plays in Air-Traffic Management a role similar to DAL. The approach analyzes the dependencies among the entities and resources of a distributed system related with their interactions. Then inequalities between the SWAL of these entities are defined. The resolution of these inequalities leads to the allocation of SWAL levels for the components of the distributed system. This approach needs a more detailed description of the system design than what is used in our approach (e.g. minimal cut sets). It could be used at a later stage of the design of the systems, when the software design and implementation is established, in order to check that dependencies are consistent with the DAL that were allocated at the preliminary design stages.

Perspectives

We are currently working on a similar approach for the allocation of Quantitative budgets to function and item faults (see [5]). The tool analyzes minimal cut sets and proposes a mean probability of occurrence for each item fault such that the probability of the failure condition is acceptable according to its severity. The approach also formalizes the allocation as a Constraint Satisfaction Problem but we use Mixed Integer Linear Programming tools instead of pseudo-Boolean tools to solve the CSP.

More generally one of our research objectives would consist in investigating the integration of models and tools based on constraint satisfaction problems for avionics design assistance. This could contain tools for independence and DAL allocation, allocation of Modular avionics modules and communication path [11], real-time scheduling of tasks [12].

Acknowledgements. The research described by this paper was partially supported by the European Union FP7 project MISSA. The authors would like to thank Jean-Pierre Heckmann and Alessandro Landi for their explanations about DAL allocation rules.

References

1. SAE S-18 and EUROCAE WG-63 committees: ARP4754a - Guidelines for Development of Civil Aircraft and Systems, SAE aerospace (2010)
2. P. Baufreton, J.-P. Blanquart, J.-L. Boulanger, H. Delseny, J.-C. Derrien, J. Gassino, G. Ladier, E. Ledinot, M. Leeman, P. Quéré , B. Ricque : Multi-Domain Comparison of Dependability and Safety Standards, in proceedings of ERTS 2010, http://www.erts2010.org, (2010)
3. RTCA SC167 and EUROCAE WG-12 committees: RTCA/DO-178B - Software Considerations in Airborne Systems and Equipment Certification, RTCA Inc. (1992)

4. M. Bozzano, A. Villafiorita, O. Åkerlund, P. Bieber, C. Bougnol, E. Böde, M. Bretschneider, A. Cavallo, C. Castel, M. Cifaldi, A. Cimatti, A. Griffault, C. Kehren, B. Lawrence, A. Luedtke, S. Metge, C. Papadopoulos, R. Passarello, T. Peikenkamp, P. Persson, C. Seguin, L. Trotta, L. Valacca, G. Zacco: ESACS: an integrated methodology for design and safety analysis of complex systems, in proceedings of ESREL 2003,Balkema publisher. (2003)
5. Bieber, P., Delmas, R., Seguin, C.: Derivation of Qualitative and Quantitative Safety Requirements. To appear in: ESREL 2011, Balkema (2011)
6. V. Manquinho, R. Martins and I. Lynce: Improving Unsatisfiability-based Algorithms for Boolean Optimization, in Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT), (2010).
7. SAT4J: http://www.sat4j.org
8. Y. Papadopoulos, M. Walker, M.-O. Reiser, M. Weber, D.-J. Chen, M. Törngren, D. Servat, A. Abele, F. Stappert, H. Lönn, L. Berntsson, R. Johansson, F. Tagliabo, S. Torchiaro, A. Sandberg: Automatic allocation of safety integrity level, st Workshop on Critical Automotive Applications: Robustness & Safety, CARS 2010 (EDCC Workshop), Valencia, Spain, 27 April 2010 (2010)
9. Pecchia, A.: Una metodologia per la definizione dei livelli di critcità dei componenti di un sistema software complesso, Master Thesis, Università degli Studi di Napoli Federico II, Italy (2008)
11. L.Sagaspe, P. Bieber: Constraint-Based Design and Allocation of Shared Avionics Resources, 26th AIAA-IEEE Digital Avionics Systems Conference, Dallas, (2007).
12. Aleti A., Bjoernander S., Grunske L., and Meedeniya I., ArcheOpterix: An extendable tool for architecture optimization of AADL models, in Model-based Methodologies for Pervasive and Embedded Software (MOMPES), Workshop at ICSE 2009 ACM and IEEE Digital Libraries,(2009)