IFAC-PapersOnLine, 48 (7): 9-14

Automating Allocation of Development Assurance Levels: an extension to HiP-HOPS

Ioannis Sorokos, Yiannis Papadopoulos, Luis Azevedo, David Parker, Martin Walker

University of Hull, Hull, HU6 7RX, UK
(+44 (0)1482 465981, e-mail: {I.Sorokos@2012, Y.I.Papadopoulos@, L.P.Azevedo@2012, D.J.Parker@, Martin.Walker@} hull.ac.uk).

Abstract: Controlling the allocation of safety requirements across a system's architecture from the early stages of development is an aspiration embodied in numerous major safety standards. Manual approaches of applying this process in practice are ineffective due to the scale and complexity of modern electronic systems. In the work presented here, we aim to address this issue by presenting an extension to the dependability analysis and optimisation tool, HiP-HOPS, which allows automatic allocation of such requirements. We focus on aerospace requirements expressed as Development Assurance Levels (DALs); however, the proposed process and algorithms can be applied to other common forms of expression of safety requirements such as Safety Integrity Levels. We illustrate application to a model of an aircraft wheel braking system.

Keywords: automatic, safety requirements allocation, DALs, HiP-HOPS.

1. INTRODUCTION

At the early stages of systems engineering, requirement identification and allocation is crucial in driving the development correctly and efficiently towards a timely, cost-effective project completion. Indeed, changes in requirements which occur late in the development lifecycle incur much greater costs to implement (Sharif, et al., 2012). Therefore, to minimize the impact of these changes on the project's schedule and budget, it is important to employ an effective requirements engineering methodology. In the case of safety critical systems, an effective determination and distribution of safety requirements is the basis of the development process recommended by numerous safety standards. Specifically, the concept of Safety Integrity Levels (SILs), first introduced in the IEC61508 (SC 65-A, 2010) standard, provides a means of describing, summarizing and assigning such requirements across the system. This concept is shared across numerous domain-specific standards such as ISO26262 (TC 22/SC3, 2011) for the automotive industry and ARP4754A (S-18, SAE, 2010) for the aerospace industry.

Manual methods have been employed widely in the past to deal with various phases of development, including requirements engineering. However, as the scale and complexity of subject systems grow, such approaches inevitably become ineffective and a potential liability against a project's completion. For instance, in the case of allocating SILs, determining an appropriate allocation might require evaluating hundreds of thousands of possibilities, each of which has a potentially different impact in terms of development time and effort required to implement. Note that many of those options which are often chosen in practice are non-optimal incurring unnecessary costs. Obviously, a process of exhaustive evaluation of all options would incur a prohibitive cost for most projects of non-trivial scale. By introducing tool support to specific development processes, such as requirements allocation, a high degree of automation can be achieved and costs alleviated. Automation would allow the development effort to be directed towards tackling more important, higher-level development issues, freeing up precious development resources. It also provides the opportunity to repeat the process efficiently, if required, as part of subsequent development iterations.

Recently, work has enabled automatic SIL allocation from system models annotated with failure logic in (Azevedo, et al., 2014) and DAL allocation from minimum cut sets in (Bieber, et al., 2011). In this paper, we describe a further development of this work to provide robust support within the HiP-HOPS tool for model-based automatic allocation of aerospace safety requirements in the form of DALs. The work is different from that described in (Bieber, et al., 2011) because our starting point is not a minimum cut set list but the system model. Furthermore, we use meta-heuristics, specifically Tabu Search, as a basis for obtaining optimal requirement allocations.

1.1 A summary of Development Assurance Levels

ARP4754A (henceforth referred to as 'the standard') provides guidelines towards the development of civil aircraft and is employed internationally. Its recommended methodology describes a set of safety assessment processes, which are designed to be carried out in parallel to a typical system development lifecycle. By performing the activities required by these processes, with the specified level of rigor, the system can be certified to meet its regulatory requirements. Development Assurance Levels (DALs) are the aerospace equivalent to SILs and are central to the

assessment framework described by the standard. DALs are meant to represent the 'level of rigor' (ARP4754A, 2010, p. 11) which safety assessment activities are required to be performed with. To understand the importance of DALs in the safety assessment process, it is necessary to outline the standard's view of a system's architecture.

The model upon which the standard bases its recommendations divides the system's architectural elements into three categories: functions, systems and items. Functions represent a high-level view of the system's functionality. For example, a likely function in any aircraft would be 'Flight Control'. Systems are an abstraction level immediately below functions, with each system providing, either on its own or with other systems, the behaviour described by a function; e.g. a 'Wheel Braking System' could support the 'Braking' function. Finally, items represent the lower-levels of the architecture, i.e., hardware or software components or small size sub-systems.

The initial assignment of DALs to functions occurs early in the development lifecycle, during the Aircraft Functional Hazard Assessment (FHA). The FHA is the first major stage in safety assessment and is more commonly known as Functional Hazard Analysis (ARP4754A, pg.12, 2010). This analysis identifies the potential Failure Conditions (FC) associated with each function, i.e., undesirable events that could occur during operation and that compromise the aircraft's safety. Each FC has a severity classification, stemming from its impact in the worst possible case, and an estimated probability of occurrence. During the Preliminary Aircraft Safety Assessment process, DALs are assigned based on each Function's hazard severity (ARP4761, pg.43, 1996), as seen in Table 1.

Table 1. DAL to Severity correspondence

Severity	DAL
Catastrophic	A = 4
Hazardous / Severe - Major	B = 3
Major	C = 2
Minor	D = 1
No Safety Effect	E = 0

Once DALs have been assigned to functions, and an architecture for the system has been defined, DALs are iteratively assigned to more refined architectural elements. The rules that guide the process of allocation utilize the concept of Functional Failure Sets (FFS). FFS contain the minimum combinations of function, system or item failures, termed Functional Failures (FF) in the standard, which are necessary and sufficient to cause a failure of the system containing said architectural elements (ARP4754A, pg.11, 2010). They are assigned with the DALs of the system failure they originate from. In turn, items whose failures belong to a given FFS can be, in principle, immediately allocated with its DAL. However, the standard allows for some items to receive less stringent DAL allocations. There are two options in doing so. Given a FFS with a DAL of k:

Option 1: a singular member is assigned a DAL of at least **k** and the other members of at least **k-2**.

Option 2: two members are assigned a DAL of at least **k-1** and the other members of at least **k-2**.

In the case of a FFS with only one member, option 1 is taken.

The allocation rules effectively state that in a set of items which, by failing together, cause a system failure, either one of the items must be developed at the system DAL with the rest developed at two DALs below, or two items must be developed one DAL below the system DAL with the rest developed two DALs below system DAL.

1.2 The issue of Independence in ARP4754A

In the standard, the concept of "Independence" is said to be 'a fundamental attribute to consider when assigning Development Assurance Levels' (ARP4754A, pg. 41, 2010). The standard uses independence as an attribute aiming to address the issue of common mode errors, which occur due to shared requirements amongst Functions or development processes amongst Items. It is important to explain our views on this matter and how we treat dependence and independence in the approach presented here.

The standard introduces two forms of independence, Functional and Item independence. The former refers to the presence of common causes of failure between separate Functions or Systems of the architecture, while the latter refers to separate Items. In both cases, identification of such common causes falls within the purview of the Aircraft/System Common Cause Analysis (CCA) process. The CCA process identifies such causes and includes them in the failure analyses performed at subsequent stages such as fault tree analysis.

In HiP-HOPS common causes are treated in two ways. They can be explicitly specified in the components and cause, via propagation of common energy, material or data errors, failure of more than one element in system models. Examples of such common causes are common power supplies or data sources that affect more than one element. Implicit common causes are events such as flood and fire which are typically examined in zonal analysis. They can also be specified at the model level and trigger simultaneous failure of more than one function or components directly and without explicit propagation of errors through the architecture. If the failure analysis determines that such failures indeed contribute towards the failure of seemingly independent Items, Systems or Functions, they will accordingly affect the DAL allocation to these elements. Therefore, our method correctly allocates DALs while addressing the issue of independence taking simultaneously into account all possible sources of failure in the system.

1.3 Challenges in requirements allocation

The allocation of a specific DAL (or SIL or similar safety requirement) to a function or item typically implies a development cost. Higher DALs imply a higher level of rigour and more costly development and assurance activities. This is clear in the standards where it is possible to observe, for instance, that the higher the DAL for a software item, the

higher the number of assurance objectives that must be met (Nordhoff, p. 7). It is apparent that allocation schemes which can achieve the required integrity for the system by assigning lower DALs to more items would be more economical and translate into less effort and time spent on assurance activities. It is precisely those cost-optimal allocations that one is interested in finding during the refinement of a system architecture under design. This problem can be more formally defined as a constrained optimization problem, with the decision variables being the DALs of each item; the constraints being the rules of allocation defined in the standard; and the objective of the optimization being to minimize the overall cost imposed by the allocation on the development process. This description can be summarized in the following expressions:

$$\underset{X}{\operatorname{argmin}} \sum_{i=0}^{n} Cost(X_i)$$

Where

X_i: the i-th allocated item DAL across all functions

Cost: the cost function, assigning each DAL a specific cost

We attempt to identify the allocation of item DALs across all functions which minimizes the total cost impact, subject to the following constraints:

$\exists X_{ik} \in X_k : [(X_{ik} \geq k) \cap (\forall X_{jk} \in X_k \geq k-2, i \neq j)]$

Or

$\exists X_{ik}, X_{jk} \in X_k : \{[(X_{ik}+1 \geq k) \cap (X_{jk}+1 \geq k)] \cap (\forall X_{mk} \in X_k \geq k-2, i \neq j \neq m)\}$

Where

X_{ik}: the i-th allocated item DAL contributing to a function with a DAL of k

X_k: the set of DALs for items of a function with DAL of k

The two constraints represent the two options available when allocating DALs (see section 1.1). The first constraint ensures that one member has a DAL of at least **k**, as in option 1, whereas the second that two members have DALs of at least **k-1**, as in option 2. In both cases, the remaining members must have DALs of at least **k-2**.

2. AUTOMATIC ALLOCATION OF DEVELOPMENT ASSURANCE LEVELS

2.1 HiP-HOPS

Hierarchically Performed Hazard Origin and Propagation Studies (Papadopoulos, et al., 2011) is a state-of-the-art model-based safety analysis software tool that largely automates the synthesis of fault trees and FMEAs from system models. Model-based development is a design paradigm in which the nominal behaviour of a system is developed using a common formal or semi-formal model of the system to facilitate communication of requirements and design between stakeholders in complex development processes. Model-based safety analysis extends this paradigm by enhancing the nominal behaviour of the system described in the model with component failure logic (Sharvia & Papadopoulos, 2011). This allows safety analyses to be conducted synchronously with the rest of the development activities and provide feedback earlier and more efficiently. HiP-HOPS requires a model of the system that is annotated with local failure logic for each component from which the tool then automatically synthesizes fault trees. These fault trees represent the failure logic of the system in the form of a tree, with the root of the tree being the 'top event' representing system failure and its leaves being base component failures. These are linked through a series of logical connectors such as AND and OR gates. Once the fault tree synthesis stage is complete, the tool analyzes the trees to produce useful safety artefacts, such as the system's Minimum Cut Sets. These sets contain the combinations of basic failure events whose occurrence is both necessary and sufficient to cause the overall system's failure. These sets are equivalent to the standard's FFS (ARP574-A, pg.41, 2010), therefore we can use them in applying the DAL decomposition rules to allocate DALs onto the system's components. A more detailed description of HiP-HOPS can be found in (Papadopoulos, et al., 2011).

2.2 Reduction Stage

The rules of DAL allocation allow multiple allocation possibilities when an FFS has more than one member. In large systems, these options can multiply leading to a combinatorial explosion. Indeed, in practice, many options exist for the allocation of function DALs to items of an architecture, often too many to consider exhaustively. The process would certainly benefit from an optimization algorithm that can efficiently search the large space of potential allocations to seek a cost-optimal allocation. We present such an algorithm in the next section which benefits from a pre-processing step for search space reduction.

Let us consider the following scenario in which each FFS contributes to a different function of DAL A or lower and therefore inherit said DAL:

FFS 1 = { FF1}

FFS 2 = { FF1, FF2 }

FFS 2 = { FF2, FF3 }

FFS 4 = { FF2, FF3, FF4 }

In this illustrative example, the relative costs implementing an element according to the different DALs are introduced in Table 2.

Table 2. Example Cost Function

DAL	A	B	C	D	E
Cost	50	40	20	5	0

An interesting phenomenon is occurring in this scenario. Each of the sets contain at least one member from a previous set and one member from the next, apart from the first and last. Additionally, each set only contains one member not belonging to a previous set. The cost function itself seen in

Table 4 is also interesting, as it is strictly increasing with regards to the DALs and non-linear.

Although there are multiple possible optimal solutions, finding one in this case does not necessarily involve enumerating all options. The solution given in Table 3, for instance, can be found using the following reasoning steps:

- FF1 was assigned A because it belongs to FFS1 and is the sole member, therefore inheriting its level.
- FF2 was assigned C because the other member of FFS2 is FF1 and has already been assigned level A, thus C is the lowest allowable level.
- FF3 was assigned A because the other member of FFS3 is FF2 and has already been assigned level C. Note that assigning FF2 and FF3 level B would result in a costlier allocation, due to the cost function, as C + A = 70 whereas B + B = 80. This is where the nature of the cost function chosen plays a particular role.
- Finally, FF4 was assigned C because another member of FFS4, FF3, has already been assigned level A.

Table 3. Example Optimal Allocation

FF1	FF2	FF3	FF4	Cost
A	C	A	C	140

Although this reasoning excludes the other possible optimal allocation, given in Table 4, it can still lead to an optimal solution (as shown) and does not require investigating other options that could be derived from the rules.

Table 4. Alternative Optimal Allocation

FF1	FF2	FF3	FF4	Cost
A	A	C	C	140

Let us now try to generalise the above example. Due to the high severity of aircraft hazards, the rules for DAL allocation are stricter than those found in other standards, allowing DAL reduction of only two levels at most. This allows us, when a model and the cost function exhibit certain properties, to reduce the possible allocations significantly by removing inefficient options, in some cases even eliminating all options of allocation down to one without loss of optimality. Even when there are still options remaining for optimization, the search space of the problem has been significantly reduced, thereby likely improving the effectiveness of the optimization technique employed subsequently.

These series of allocations can be applied when:

1) the cost function of each element is non-linear and strictly increasing with respect to the DALs of its FFs
2) there exist N FFSs for all of the architecture's effects (N can be less than the total number of FFS in the architecture) that, when ordered in descending order of their effect's DAL, exhibit the following 'chain' property:

   Let FFS_i of size n be followed by FFS_{i+1}, FFS_{i+2} and so on. The chain property holds for these FFSs if:

   a) there exists a common FF that belongs to both FFS_i and FFS_{i+1}
   b) $|FFS_{i+1}| - |FFS_i| \leq 1$, i.e. the difference in the cardinality of two neighbouring FFS in the chain is maximum one

3) there exists a FFS amongst those N that satisfies the chain property with a single member

Note that the above heuristic can only apply in analysis of simultaneous allocation of more than one function DALs. The reason is that, in the case of allocation of a single DAL which is done on the basis of analysis of a single fault tree, the redundant FFS required to satisfy the chain property have already been eliminated during logical reduction of the sets. In cases of multiple allocation of DALs, the chain property may apply to subsets of the total set of FFS of the system and can be used in those cases to fix a subset of allocations in the system, thus reducing the overall search space required in subsequent optimisation.The pseudo code for this reduction stage follows:

1) sizeCounter = 1
2) sort all FFSs in descending order of DAL
3) changesMade = true
4) while changesMade is true
   a) changesMade = false
   b) foreach FFS k
      i) if sizeCounter = k.size then:
         (1) if there is just one Member in k unassigned, then:
            (a) assign it the lowest possible DAL
            (b) changesMade = true
         (2) end if
      ii) end if
   c) end foreach
   d) increment sizeCounter
5) end while

2.3 Tabu Search

There is a range of optimization algorithms that could be adapted to solve the DAL allocation problem. We chose Tabu Search (Glover, 1986) for this study as a good candidate as it has already shown good performance in earlier work in allocation of automotive requirements (Azevedo, et al., 2013). Tabu search is a meta-heuristic optimization technique, which owes its name to its memory structures, used to store recently evaluated candidate solutions. The candidates stored in these structures are not eligible for generation of further candidates and are thereby considered 'Tabu' by the algorithm. The memory artefacts allow for the technique to trade space for time and therefore accelerate the search for the optimal solution. We have implemented a basic

version of Tabu Search; each candidate is an allocation of DALs over all FFs and the best candidates are those with the lowest overall DAL cost. Candidates recently chosen and therefore Tabu are stored on the short-term memory structure, the 'Tenure'. An Aspiration Criterion is employed, allowing a candidate to be chosen despite being Tabu. The chosen criterion requires the candidate allocation to beat the Tenure's current best candidate in terms of overall DAL cost, thus being the best (i.e. cheapest) allocation in recent memory. The search method used to generate the next set of candidates produces a new candidate for each allocation of the current one that can be changed and not violate DAL decomposition rules. This means that allocations assigned by the reduction stage cannot be reduced in DAL under the level they were then assigned, only increased. The algorithm for Tabu Search follows:

1) Generate a random allocation
2) Set random allocation as the current choice
3) Add the current choice to Tabu Tenure
4) Repeat until iteration count or time limit are reached
   i) Produce random alternative allocations from the current choice
   ii) Sort the produced allocations by DAL cost, ascending
   iii) Select the lowest cost allocation as the next choice
   iv) Repeat until a next choice has been selected or all alternative allocations have been examined
      (1) If the next choice is not Tabu, select it to be the next choice
      (2) If it is Tabu but aspiring, select it to be the next choice
      (3) Otherwise, examine the next produced choice
   v) If none of the produced allocations is either non-Tabu or aspiring, set the lowest cost one as the next choice
5) The next choice becomes the current choice
6) Add the current choice to the Tabu Tenure
7) Sort the Tabu Tenure by DAL cost, ascending

The potential options per each FFS are placed in 'Allocation Packs'. Generating a random allocation in Step 1 involves selecting a random option from each Allocation Pack and then combining them with the non-optional allocations from the reduction stage, as mentioned earlier. Sorting the generated allocations for the next iteration means that after each iteration, the lowest cost — and ideally non-Tabu or aspiring — choice out of the produced candidates will have been made. To check if a candidate is Tabu, the fixed-size Tabu Tenure is parsed to see if there's an identical allocation on it. If so, the subject allocation is not allowed. An allocation is considered aspiring if its cost is lower than each member of the Tabu Tenure. Thanks to the sorting of the list in Step 4ii, a simple comparison with the first entry in the list is only needed.

To produce the next set of alternative allocations, the process described in the following algorithm is performed:

1) Add into list Optional all 'optional' partial allocations
2) Add into list IncreasableNonOptional all 'non-optional' partial allocations of DAL lower than 4
3) For every partial allocation in the IncreasableNonOptional list,
   a) Select a random Allocation Pack affecting that partial allocation
   b) Select a random partial allocation from that Pack which assigns a higher DAL than the current one
   c) If none exist, continue to the next partial allocation
   d) Else, use that selection to generate a new allocation and add it to the resulting list
   e) Repeat
4) For every partial allocation in the Optional list,
   a) Select a random Allocation Pack affecting that partial allocation
   b) Select a random partial allocation from that Pack which alters the current allocation
   c) Use that selection to generate a new allocation and add it to the resulting list
   d) Repeat
5) Return the resulting list

Note that steps 3 and 4 of the above algorithm are almost (except for sub-steps b and c) identical — simply applied to a different list. The purpose of the two lists is to find all partial allocations due to options taken from Allocation Packs which can be altered. Sub-step b) illustrates the difference between the two lists. In Optional, any partial allocation which alters the current allocation can be selected, whereas in the IncreasableNonOptional list we can only choose a partial allocation, if any exist, which increases the DAL of the resulting allocation.

3. CASE STUDY: AIRCRAFT WHEEL-BRAKING SYSTEM

Our case study is based on an example aircraft wheel braking system from ARP4761 (S-18, SAE, 1996), adapted by (Sharvia & Papadopoulos, 2011). The system is illustrated in Figure 1. The purpose of the system is to provide safe braking during aircraft takeoff and landing. It features two primary hydraulic pumps, GreenPump and BluePump. The Brake System Control Unit (BSCU) forwards input from the brake pedals to the brakes, monitors input systems and states for correctness and provides feedback to other systems. The SelectorValve receives a constant stream of pressure from both pumps, relaying the pressure from the appropriate one to the corresponding meter valve. The anti-skid meter valves (ASMeterValveG and ASMeterValveB) output the required amount of pressure based on BSCU's commands. The system features two modes of operation, Normal and Alternate. In Normal mode, GreenPump is used, sending pressure through the SelectorValve to ASMeterValveG. In Alternate mode, BluePump is used to send pressure through the SelectorValve to ASMeterValveB. Alternate mode is activated by BSCU when the pressure output falls under a certain threshold in Normal mode.

If braking fails during takeoff or landing, consequences could be catastrophic. The plane could fail to decelerate as expected on landing, or brake during take-off, potentially causing a severe accident. We found this reasoning sufficient to test the allocation by assigning the overall DAL of the system output (WBS) to be A. We should note that this assignment is

primarily used as an example and in practice the actual DAL assignment could be lower. However, this would not impact the allocation process described; a lower allocation can only potentially lead to a smaller number of potential allocations that need to be evaluated. Therefore, in this sense, we are demonstrating the worst case scenario with regards to the number of potential candidates that need to be evaluated. For the purposes of the case study, the costs in Table 5 were used to approximate the cost each DAL would have on its component.

Table 5. Item DAL Cost

DAL	A = 4	B = 3	C = 2	D = 1	E = 0
Cost	50	40	20	10	0

The resulting allocation when the parent FC (WBS) is assigned with DAL A can be seen in Fig. 1. (DALs from A to E numbered from 4 to 0 respectively).

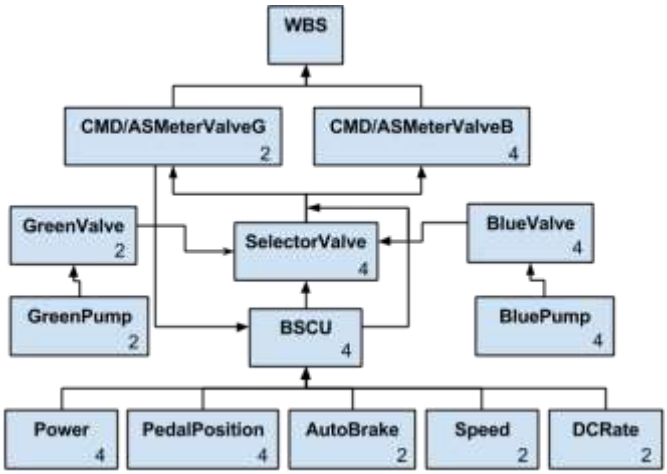

Fig. 1. Allocation of DALs on model.

The allocation algorithm was executed numerous times, each time producing a permutation of the allocation shown above or the one displayed with the same overall DAL cost. This allocation was found to be optimal after exhaustively enumerating all possible combinations of DAL allocations for this model. It should be noted that this was only possible due to the relative small scale of the search space (531,441 possible allocations). Larger-scale models might require excessively long periods of time to exhaustively search for their optimal solutions.

6. CONCLUSIONS

In the aerospace industry, dissemination of safety requirements across the system's architecture is a fundamental part of the safety development process described in ARP4754A. Applying the guidance of the standard is challenging as an increase to the number of components within a system results in a super-linear growth of the number of allocation options that need to be evaluated. Furthermore, finding a trivial allocation manually would not be ideal, as each allocation has a different impact on development costs. Therefore, there is a significant incentive in determining the optimal allocation automatically and in this paper we have described a method and tool to achieve this. We have demonstrated that it is possible to allocate DALs to a given system architecture automatically, optimally and efficiently by applying this method to an example system. This development suggests that the possibility of automation of safety development processes is common to multiple standards and we feel confident that further automation is possible in this direction. Reflecting on the implementation of the Tabu Search, we believe it could be improved by including midterm and long-term memory structures, which would allow models with larger FFSs to have their DALs allocated more effectively. Additionally, although Tabu Search has proven to be an effective meta-heuristic in solving the problem, other optimization techniques could be evaluated to compare their efficiency, as indicated in relevant work described in (Bieber, et al., 2011).

REFERENCES

Azevedo, L. S. et al., 2013. *Automatic Decomposition of Safety Integrity Levels: Optimization by Tabu Search.* Toulouse, France, s.n.

Azevedo, L. S. et al., 2014. Assisted Assignment of Automotive Safety Requirements. *IEEE Software,* 31(1), pp. 62-68.

Bieber, P., Delmas, R. & Seguin, C., 2011. *DALculus - Theory and Tool for Development Assurance Level Allocation.* Naples, Italy, Springer Berlin Heidelberg, pp. 43-56.

Glover, F., 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, pp. 533-549.

Nordhoff, S., n.d. *DO-178C / ED-12C - The new software standard for the avionic industry: goals, changes and challenges.* [Online]
Available at: www.sqs.com/uk/_download/DO-178C_ED-12C.pdf

Papadopoulos, Y. et al., 2011. Engineering failure analysis and design optimisation with HiP-HOPS. *Engineering Failure Analysis*, pp. 590-608.

S-18, SAE, 1996. *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment,* s.l.: SAE Int..

S-18, SAE, 2010. *ARP4754A Guidelines for Development of Civil Aircraft and Systems,* s.l.: SAE Int..

SC 65-A, 2010. *IEC61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems,* s.l.: International Electrotechnical Commission.

Sharif, B., Khan, S. A. & Bhatti, M. W., 2012. Measuring the Impact of Changing Requirements on Software Project Cost: An Empirical Investigation. *International Journal of Computer Science Issues,* 9(3), pp. 170-174.

Sharvia, S. & Papadopoulos, Y., 2011. *IACoB-SA: an Approach towards Integrated Safety Assessment.* Trieste, Italy, IEEE, pp. 220-225.

TC 22/SC3, 2011. *ISO 26262 - Road vehicles - Functional safety,* s.l.: International Organization for Standardization.