A SYSTEMS ARCHITECTING FRAMEWORK FOR DISTRIBUTED INTEGRATED MODULAR AVIONICS

B. Annighöfer, F. Thielecke,
Hamburg University of Technology / SYSTAR Innovation,
Nesspriel 5, 21129 Hamburg, Germany

Abstract

This work presents a novel holistic framework for Distributed Integrated Modular Avionics (DIMA) architecture design and optimization. IMA is a standardization of avionics components. IMA is beneficial in weight and costs if the complexity of sizing, function allocation, and topology selection is mastered. A holistic framework enables model and algorithm-aided design of avionics architectures. Domain specific modeling of systems software, hardware, and aircraft anatomy enables automated verification and early evaluation of architectures. Moreover, the model is the foundation for a flexible kit of eight optimization routines. For design issues in which humans likely lose the overview optimization routines are proposed. Automation ranges from function mapping over routing to a complete architecture generation. Routines for platform selection, network, and topology optimization are unique and unrivaled today. All optimization problems are solved globally optimal and a multi-objective solving algorithm calculates the best trade-off architectures for contradicting objectives, the Pareto optimum. All optimization routines are extensively tested by designing the optimal DIMA architecture for aircraft system functions in an A320-like scenario. Results show significant optimization potential of generated architectures compared to a manually designed one. The resulting architectures are analyzed and compared in performance and structure in detail.

1. INTRODUCTION

Integrated Modular Avionics (IMA) are state-of-the art for avionics systems of recent aircraft. An IMA system consists of standardized hardware for computing and I/O, as well as a common high-bandwidth network. The resources of the hardware are shared in a safe manner between hosted aircraft system functions, i.e. IMA is a single avionics system for multiple avionics functions. Its purpose is defined by function allocation and configuration [1]. The shared utilization of fewer devices and fewer device types make IMA systems superior in weight and costs compared to traditional avionics systems [2]. The second generations of IMA platforms, so called Distributed IMA (DIMA), increase the saving potential by a separation of computing and I/O and by spatially distributed IMA device installations [3].

The **main challenge** in developing DIMA avionics systems, called DIMA architectures, is the complexity resulting from shared resources and spatial distribution [4, 5]. Currently CPUs, memory, and I/O are shared by approximately 1000 individual functions and peripheral components such as sensors and actuators. Current architectures comprise round about 50 DIMA modules, and up to 1000 installation locations exist for modules and peripherals. Bringing systems, hardware, and anatomy together, as visualized in FIG 1, is the main design challenge in planning DIMA systems. Moreover, since the number of hosted functions and capabilities of electronic devices rises continuously [6], the number of possible architecture variants explodes. Moreover, systems have safety and performance requirements and the importance of complex contradicting economic design objectives rises. Today DIMA architectures have reached a number of objects, relations, requirements, and objectives that make engineers struggle with finding a valid and, especially, with finding the optimal architecture. A straight and target oriented systems engineering, alias systems architecting [7], is hardly possible by hand. The manual design process is an iterative trial and error procedure. At the end of the design process the correctness of the architecture is assured, but the optimality is unknown. This situation could be improved by computer-aided design methods.

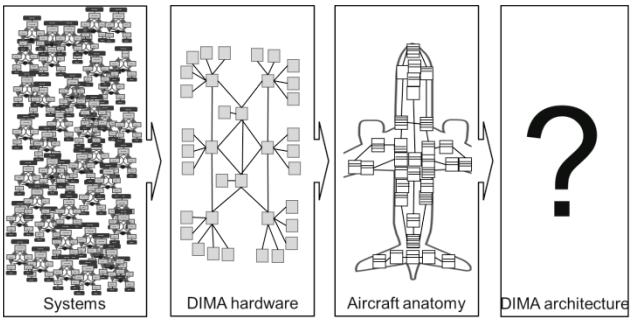

FIG 1: The main design challenge of modern avionics

Computer-aided design of IMA architectures is an **active research area**. Research is mainly divided in modeling, function allocation, and signal routing. Approaches for modeling IMA architectures with the goal of simulation and verification can be found in [8], [9], [10], [11], and [12]. Most often these approaches are based on AADL [13] or SysML [14]. Both seem not rigid and complete enough for holistic algorithmic aid. The optimal distribution of functions is a traditional challenge in computer science. Approaches for IMA can be found in [15], [16], [17], [18], and [19]. These approaches range from function distribution to redundancy allocation. The relations of different automations are commonly not considered. Moreover, the size for which methods are demonstrated is typically only around 20 functions. The last major topic is the assignment of signals to a common bus system. This is most commonly done for AFDX as for instance in [20], [21], [22], [23], and

[24]. The relation between function allocation and network allocation is not considered, and the demonstration scopes are again small.

The framework presented supports the DIMA design process with **model-based engineering and mathematical optimization**. Therefore, it presents a novel domain specific model especially for planning avionics architectures. In addition, methods for architecture optimization from the different domains are stream-lined, and based on a mathematical foundation that allows global optimal and multi-objective optimization. Moreover, completely novel approaches for device sizing, topology generation, and complete architecture generation are presented. Most important, model and optimization are integrated in a seamless flexible framework.

This paper is structured as follows.

Chapter 2 presents a domain specific model for DIMA architectures and requirements that allows a formal and rigid representation of architecture data. The model covers systems, hardware, anatomy, and mapping.

In **Chapter 3** a holistic set of optimization routines for DIMA design issues is presented. A general description of the routines, as well as the fundamental mathematical solution approach is given.

Chapter 4 demonstrates modeling and optimization. Four aircraft systems from the air-domain are modeled and optimization routines are extensively applied. Resulting improvements and architectures are analyzed.

The paper ends with a discussion and conclusion in **chapter 5** and **chapter 6.**

2. AVIONICS ARCHITECTURE MODEL

The baseline of the avionics architecting framework is a domain specific model especially designed for planning avionics architectures. Planning DIMA architectures is a highly concurrent process [1]. The actual architecture is developed by the avionics design engineer. The planning, however, strongly depends on the functions to be hosted, the peripherals to be connected, the available hardware, and the aircraft anatomy. Functions, DIMA hardware, and aircraft anatomy are developed in parallel. It is, therefore, of major importance to decouple the system, hardware, and structure domain as good as possible. Nevertheless, one of the main challenges is to bring software, hardware, and anatomy together optimally. Moreover, in the early design stages requirements can quickly be changed and several architecture variants must be compared. Besides avionics architectures, therefore, the system requirements that are driving for an architecture must be represented. The modeling of those requirements must be rigid enough to enable automatic verifications. Moreover, the attributes shall be modeled that allow early evaluations of architectures and variants in respect to mass or costs.

The model developed is a static model of systems, hardware, and installations. Static means a time invariant representation of functions, signals, and resource sharing. The top level structure of the model is depicted in FIG 2. The main three layers are **systems**, **hardware**, and **installation**. Those are almost independent. However, all three layers are built upon the same components from a definitions layer, e.g. resource or device types. The operational avionics architecture is composed in a **mapping** layer. It integrates elements from systems, hardware, and software models. Several different mappings might exist. Therewith high reuse of model elements and an easy comparison of architecture variants are enabled. A **scenario** layer includes parameters for evaluation.

The concept and content of the systems, hardware, installation, and mapping layer, is described in detail in the following.

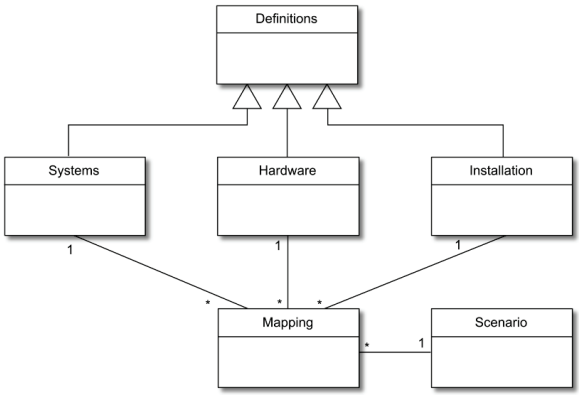

FIG 2: Class diagram of the main layers of the avionics architectures model

2.1. Systems

The main driver for an avionics architecture are the aircraft systems to be hosted. When considering DIMA, the main components of systems are software functions, sensors, and actuators. In addition, functions and peripherals exchange data. The system functions, I/Os to connect peripherals, and the communication backbone must be provided by DIMA hardware. Therefore, systems are modeled as a set of tasks that exchange signals. An example is given in FIG 3.

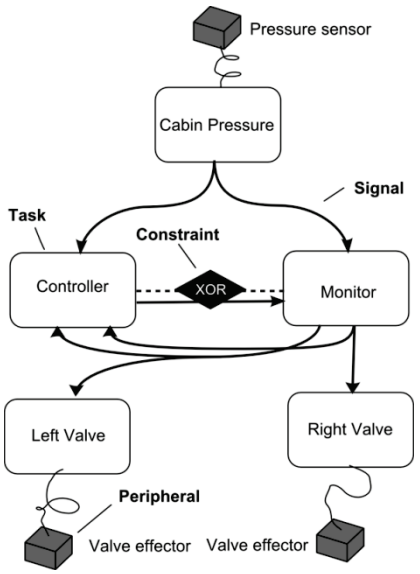

FIG 3: Model example for a system

Both tasks and signals specify required resources. A resource is a countable quantity that has to be provided by the hosting hardware. The task or signal specifies how much of a certain resource type is consumed upon hosting. Examples for resources are CPU power in MIPS, memory in MB, bandwidth in Mbps, the number of analog interfaces, or the number of CAN busses. This simple

resource model allows a major verification. After systems are assigned to hardware, the resources on none of the devices must be exceeded. However, resources are not sufficient to express operational, safety, and performance needs of systems. Therefore, eight general additional constraints have been identified sufficient to characterize all function needs.

- **Peripheral constraints** connect functions to peripherals. During mapping it must be ensured that the function requiring the peripheral is physically connected to the peripheral. This requires an I/O interface, as well as wiring. The latter induces weight and cost.
- **Device constraints** limit the possible mapping of a function to a set of devices. Alternatively, a set of devices can be excluded for the function, although resources might be available.
- **Installation locations constraints**. Prohibit or force certain locations for functions, i.e. the device hosting the function must or must not be in one of the specified locations. For instance the rotor burst area can be excluded for a function.
- **Power supply constraints** specify the power supply that a device must, or must not have if a function is mapped on the device. For instance safety critical functions may only be mapped on devices connected to the emergency power bus.
- **Segregation constraints** prohibit two or more functions on the same hardware. This is a very common constraint for redundant lanes of a system.
- **Location segregation** constraints prohibit not only the same device, but also all devices in the same location for the mapping of two functions. This incorporates spatial safety considerations.
- **Dissimilarity constraints** are applied to functions that for redundancy reasons must be mapped to different device types.
- **Latency constraints** can be applied to a series of functions and signals that form a time-critical path. It is assumed that this chain is executed periodically, and that each execution must be faster than the specified maximum execution time when mapped to hardware. Therefore, the execution and transmission times for signals and tasks on different hardware types must be known in terms of Worst-Case-Execution-Time (WCET).

2.2. Hardware

The execution platform for system functions is the DIMA hardware. This is modeled in the hardware layer. Hardware is basically expressed by general **devices** and **links**. It is independent of a certain technology. Each device is an instance of a device type. Device types are stored in the definitions layer. The device type specifies physical characteristics such as weight, dimensions, or reliability. Moreover, the device type specifies the resources available for task hosting. Basically, the number and type of available resources is specified. Since resource provision can be ambiguous the resource model is extended by **capabilities**. For instance for I/Os with lightning protection or current levels a minimum requirement exist, but all I/Os with higher levels might be used as alternatives. A capability defines for a device type, which task can be mapped under which precise resource consumption. There can be multiple capabilities for the same type with different resource consumptions on the same device type. Links are point-to-point communications between devices. Links provide resources and capabilities like devices. If the communication is a switched network or a bus, the switch or bus is represented by an additional device. An example of Core Processing Modules (CPM), Remote Data Concentrators (RDC), and an AFDX network is given in FIG 4. It shows the resources available on the device and link types, as well as the resources consumed if mapping a *Controller*, *Monitor*, *Pressure* or *Valve* task; or a *Float* or *Boolean* signals.

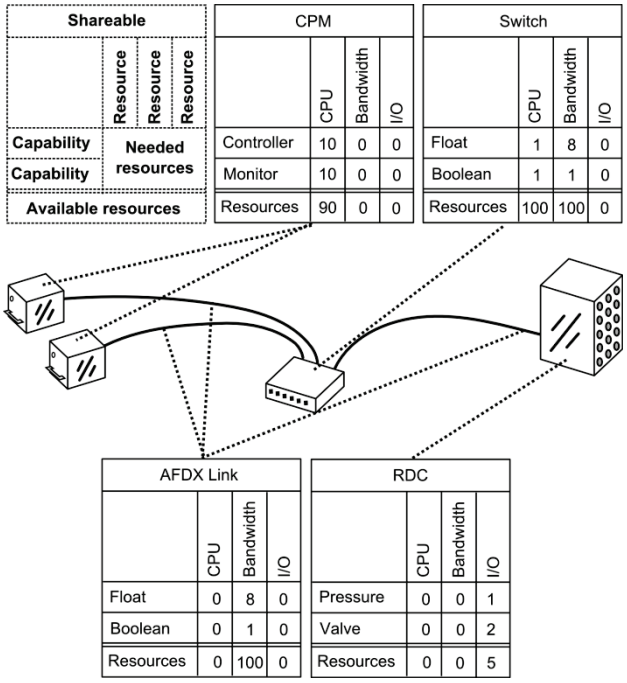

FIG 4: Resource sharing and capabilities model

2.3. Installation

The aircraft structure or anatomy is the container for avionics. Installation space and infrastructure like fixation, cooling, and power are required. The available installation structure is modeled in the installation layer. Rooms for devices are **installation locations.** Locations provide infrastructure resources like A600 slots, volume or cooling capacity. Device types define the numbers of resources required. In a valid architecture the resources of locations shall not be exceeded by the infrastructure resources required by the installed devices. Links and peripheral wires can be inside a location. Connections between installation locations are modeled with cable routes and joints. A **cable route** is a point-to-point connection with a fixed length. Cable routes may be split and merged at cable **route joints**. Routes and joints form a topology of the installation anatomy. The lengths of cable routes can be correlated to 3D coordinates, but this is not necessary. An example of three installation locations, their infrastructure resources, and connections is depicted in FIG 5.

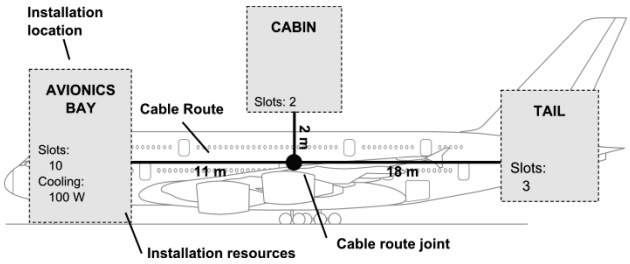

FIG 5: Modeling elements of the installation layer

2.4. Mapping

The avionics architecture is only operational if tasks are mapped to devices, signals to links, devices to locations, and links to cable routes. This is modeled in the mapping layer. One mapping references one systems layer, one hardware layer, and one installation layer. For each element of the referenced layers a **mapping object** in the mapping layer exists. For instance devices are represented by **device assignments** and tasks by **task assignments.** These assignment objects are arranged hierarchical, i.e. a task assignment is a child of a device assignment which is a child of a **location assignment**. Signal and wiring assignments are split in several segments along their route. Each segment is assigned to a link, device, or cable route. With this hierarchy the mapping directly presents the realization of a certain architecture. All verifications and evaluations are carried out on a mapping. Moreover, the same system, hardware, and installation elements can be referenced in multiple other mappings to express alternatives. In addition, changes in the systems, hardware, and installation layers will directly propagate to all referencing mappings.

2.5. Avionics Architect

The model described above is implemented in the **Avionics Architect** (AA) of SYSTAR Innovation[1] . The Avionics Architect (s. FIG 6) is a planning environment for avionics architectures. It supports modeling, verification, evaluation, and comparison as described above. It is an Eclipse-based implementation, using the formal data modeling language ECORE for the architecture domain model. The model is extended with a flexible verification and evaluation framework. Both verification and evaluation can be extended by arbitrary custom rules and objectives. Rules and objectives are evaluated on-the-fly during modeling.

FIG 6: Avionics Architect of SYSTAR Innovation

[1] TuTech SYSTAR Innovation GmbH is a Spin-off of the Hamburg University of Technology

Modeling is carried out in a hierarchical editor or in special 2D views for different aspects. 2D editors exist for definitions, systems, hardware, installation, and mapping. Editors and views provide seamless interface to optimization routines, which are described in the following.

3. OPTIMIZATION ROUTINES

During the design of avionics architectures several design issues appear, where the human can likely lose the overview. This are all assignment issues where the number of elements to assign or the number of targets, or the number of cross relations is large, e.g. the assignment of all tasks to DIMA modules. For this the resources and constraints of approximately 1000 tasks must be considered. A mapping can be derived and verified manually. If this mapping is, however, optimal especially in respect to complex objectives as maintenance costs is often unknown. Moreover, design engineers face contradicting design objectives, where the best trade-off must be found. To speeding up assignment issues and to get improved and verified architectures is envisioned by optimization, design automation, and architecture auto-completion.

Eight optimization routines as depicted in FIG 7 were developed. The routines are organized in three levels of automation. Routines of the first level are single assignments as known from distributed systems research, e.g. task assignment or signal routing. They depend on each other. For instance task assignment depends on device assignment and vice versa. Level two routines are combined device type or network optimizations, which leverage some level one dependency. The full potential for optimization has level three with combined device and network optimization. All routines can be applied to a single objective or calculate the multi-objective optimum in terms of the Pareto optimum. In practice the routines which match the process and available data best are selected. In addition, the scope can be chosen from single system optimization to full or even multiple aircraft optimizations. Each single routine is explained in the following. This is, however, only a top-level summery. For details on mathematics see [25], [26], [27], [28], and [29].

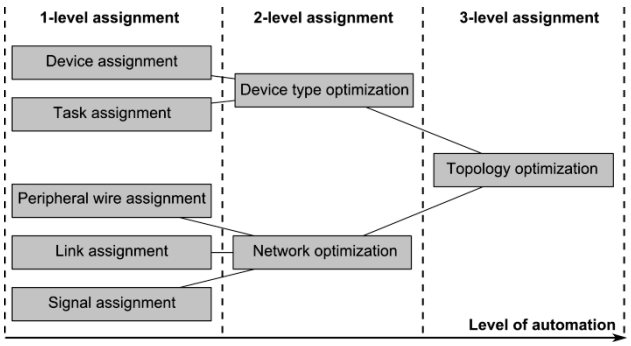

FIG 7: Comprehensive set of eight architecture optimization routines

- **Device assignment** finds the optimal installation locations for a set of devices in a given anatomy. The devices are already assigned with functions. Optimized is, for instance, the minimum mass for peripheral wirings.
- **Task assignment** maps a set of tasks to devices installed in an anatomy. The mapping considers all resource and segregation constraints. Task

assignment can, for instance, minimize the device mass by using less devices or the peripheral wiring mass.

- **Peripheral wire assignment** assumes an architecture with tasks and peripheral already assigned to locations. It finds the optimal, e.g. shortest, routes for peripheral wires. If the related tasks need to be segregated routes are also segregated.
- **Link assignment** finds the cable routes for links from the hardware layer if devices are already assigned. If signals are assigned to links also spatial segregations are respected.
- **Signal assignment** calculates the optimal routes for signals if tasks, devices, and links are already assigned. While respecting bandwidth limitations and segregations, signal assignment, can, for instance, minimize the number of necessary links and switches.
- **Device type optimization** derives the optimal number of devices, the allocation of tasks and the device sizings in parallel. Its inputs are a set of device types, system tasks, and the anatomy. The resources per device types are not specified, but the possible types and an upper limit is given. According the infrastructure resources the algorithm decides how many instances of which device type are used in which location and how tasks are distributed. The objectives can be device and wiring mass, but also costs.
- **Network optimization** finds the optimal number of links and switches for a given set of signals and the locations of devices and tasks. An additional input is the number of ports per switch. The results are switch and link instances placed in installation locations. Moreover, all signals are routed while resource and segregation constraints are hold.
- **Topology optimization** combines device type and network optimization. Since the only inputs are tasks, signals, device types, and the anatomy, it is almost complete architecture generation. It finds the optimal number of devices and network topology, while considering the trade-off between device and network weight or costs.

3.1. Algorithms and Solving

The eight optimization routines presented above are solved with the same mathematical foundation. All optimization routines are combinatorial optimization problems. A well-known mathematical representation and the most advanced global optimal solving algorithms are used. All problems are expressed as binary programs (BP) and are solved with the Branch-and-Cut approach. The difficulties are finding a suitable BP formulation for each DIMA design problem and making the right simplifications such that also architectures above 100 elements can be optimized.

The general form of a **binary program** (BP) is to find a binary solution vector x such that

$$f^T x$$

is minimized subject to

$$Ax \leq b$$

$$A^{eq} x = b^{eq}$$

$$x \in \{0,1\}^n.$$

x encodes - depending on the optimization routine - either task assignment possibilities, signal routes, or topologies and so on. See the mentioned publications for details. The cost vector f allows quasi-linear objectives. “Quasi” means that by introducing auxiliary variables also certain non-liniearities can be considered. The objectives section in the next chapter gives examples of what kind of objectives can be expressed. Linear inequalities A and equalities A^{eq} express consistency, resource, segregation, or uniqueness constraints. The most advanced global optimal approach to solve BPs are Mixed-Integer-Linear-Program (MILP) solvers based on Branch-and-Cut. Alternatively, Boolean Satisfiability (SAT) solvers might be used.

If considering multiple at least partially contradicting objectives a **multi-objective** extension must be made. Instead of calculating a single optimum, it is proposed to calculate the set of Pareto-optimal solutions. As depicted in FIG 8 Pareto optimal solutions are the best trade-off solutions for the given objectives, i.e. those candidates from which the design engineer would chose his final favorite. Formally speaking, Pareto optimal or efficient solutions are not dominated. Domination means there is no architecture more optimal in all objectives.

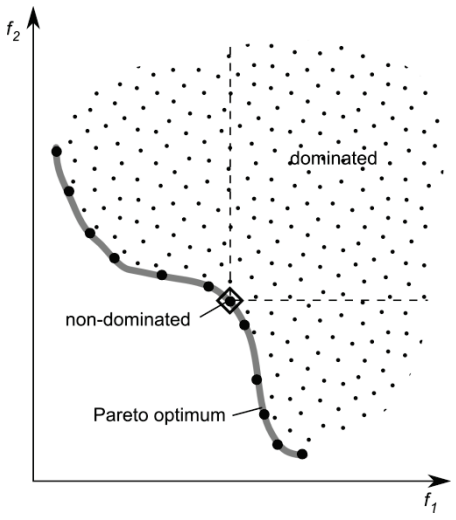

FIG 8: Pareto optimum for two minimum objectives

The presented combinatorial optimization approach is extended for an arbitrary number of linear cost functions by surrounding it with an adapted version of the **Pareto-Front-Sampling** (PFS) algorithm from [30]. As depicted in FIG 9 PFS is an iterative algorithm. In each iteration a single objective BP is solved. The solution is one point of the Pareto optimum. Between iterations variable artificial bounds on the objectives are applied, such that the boundary of the solution space is sample point by point. Although multi-objective optimization with PFS needs as much iterations as Pareto optima exists, it enables to stay with the most efficient global combinatorial solvers at the lowest level. Solving can be speed up by artificially increase the minimum stepping between two solutions.

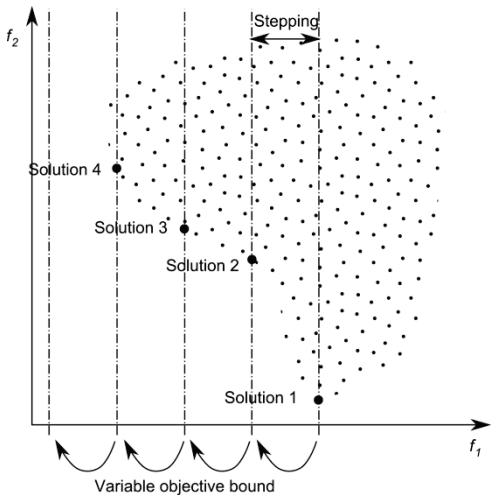

FIG 9: Pareto-Front-Sampling example

3.2. Avionics Architecting Toolbox

The eight optimization routines are efficiently implemented in the **Avionics Architecting Toolbox** (AAT), an add-on for the Avionics Architect. The AAT is integrated seamlessly into the AA. By right-clicking on model objects optimizations for the selection can be started. AAT is based on MATLAB. Information for optimization is retrieved by reading the domain specific model of the AA. From the model all relevant information for the selected optimization is retrieved and converted into the matrix representation of the BP. In addition, pre-calculations are carried out. Matrices are provided either to off-the-shelf MILP solvers or a PFS implementation. Multi-objective optimization is optionally supported by a live feedback as shown in FIG 10. The interface to MILP solvers is generic. Implemented are, for instance, CPLEX or GUROBI connectors. After optimization the results are automatically converted to model elements and stored in the architecture model. Within the AA optimized architectures can be viewed, evaluated and edited. Moreover, differences to the previous architecture are automatically visualized.

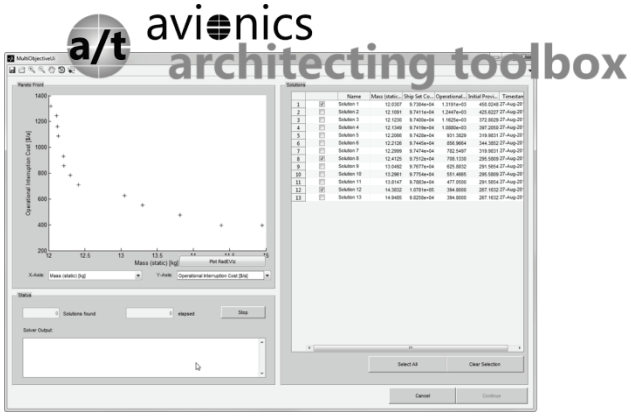

FIG 10: Multi-objective UI of the Avionics Architecting Toolbox

4. OPTIMIZATION STUDIE

Modeling and optimization is demonstrated using an A320-like scenario. The task is to find optimal DIMA architectures for four aircraft systems from the air domain. Each of the optimization routines is applied and the results are compared with a manual design.

4.1. Objectives

The objectives considered are mass, ship-set-costs, operational interruption costs and initial provisioning costs.

Mass is the cumulative mass of all DIMA devices, peripheral wires, and the network.

Ship-set-cost (SSC) are the costs for all devices and wires. SSC can also include credit, installation, and fixation costs. It is similar to mass. The major difference is that the gap between device costs and wiring costs is bigger that for mass.

Operational interruption costs (OIC) are costs spend by the airline per anno for flight delays or cancelation caused by the avionics system. It depends on the mean-time-between-failure (MTBF) of devices, the MEL-level of the failed functions, as well as the accessibility of the location. The MEL level is defined roughly as

- NOGO for no costs,
- GOIF for low countermeasure costs, and
- NOGO for high repair costs.

Moreover, airline parameters as spare part availability, fleet size, usage, and prices are considered.

Initial provisioning costs (IPC) must be spent for spare parts before an airline can operate a new aircraft type. IPC depend on the MTBF, the MEL-level, and the individual device costs.

4.2. Scenario

For demonstration the tasks of the four aircraft **systems,** a ventilation control (VCS), a bleed-air (BAS), a pneumatic (PS), and an overheat detection system (OHDS) are modeled as shown in FIG 11. Each system is composed of two redundant controller task (C) and peripheral tasks (1-9). The tasks of two redundancies must be segregated and OHDS and PS must be dissimilar. In total 58 task and 54 signals needs to be mapped. The MEL-level of the VCS and OHDS is NOGO. The BAS has GOIF and the PS is a GO system. MEL-level and segregation is inherited to signals.

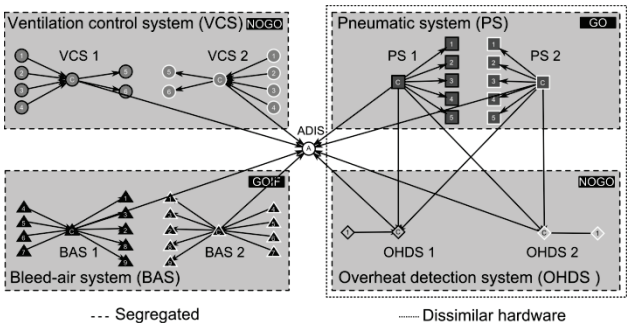

FIG 11: Systems and signals to be mapped

The **anatomy** and the position of system peripherals are given in FIG 12. The dimensions are similar to an A320-200. For DIMA devices on actively cooled avionics bay and six remote locations in the nose, middle, and tail exist. Cable routes connect locations and peripheral positions. For each installation location the access time for repair is known.

The avionics **hardware** shall be selected from a platform similar to the A350. Core Computing Modules (CPM) can host up to three controller tasks and need active cooling.

Remote Data Concentrators (RDC) connect peripherals to the AFDX network and can be installed in any installation location. Owing to the dissimilarity constraint two types of RDC and CPM are needed. Each device is assigned with a mass, cost, MTBF, and resource limit reasonable for the selected aircraft systems. For hosting the systems ten I/O types are needed. For each I/O type the space requirements in the device, as well as potential bundle sizes are known. The network shall be composed of AFDX switches with five ports and AFDX links.

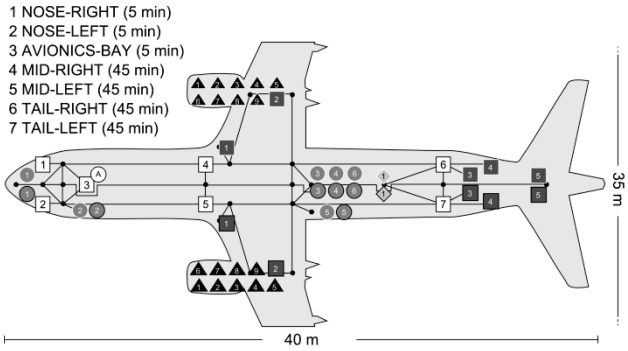

FIG 12: Installation locations and cable routes for the optimization studies

As a reference a task mapping (s. FIG 13) and network topology (s. FIG 14) was derived manually in all conscience. The architecture is a left-right symmetric utilization of four CPMs and ten RDCs. The design drivers were short peripheral wires to RDCs and segregation. Wire mass and SSC should be minimal. OIC and IPC could not be optimized manually. The network topology requires eight switches and four redundant routes. In summary the manual architecture evaluates to 35.6 kg in mass, 203 k$ SSC, 2451 $/a OIC and 768 $/aircraft IPC.

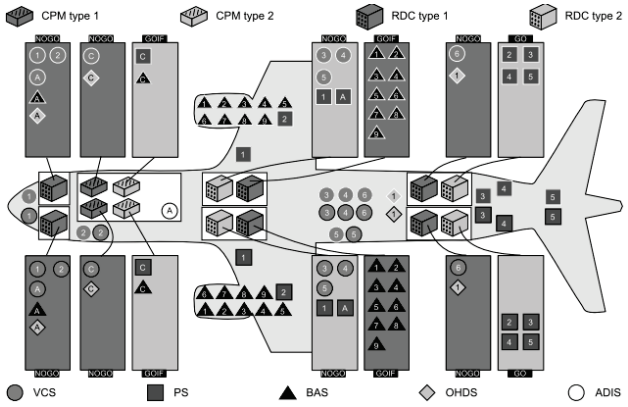

FIG 13: Manual task mapping

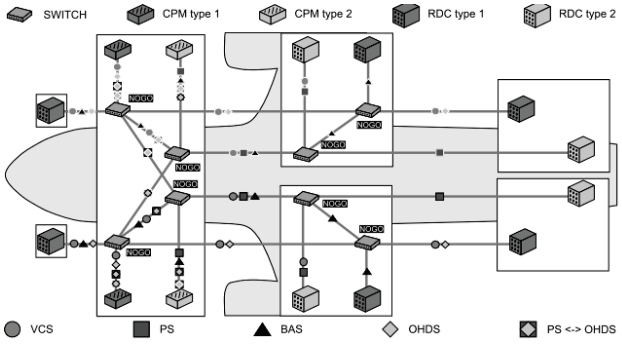

FIG 14: Manually derived network topology

4.3. Results

In total six optimizations were carried out. The resulting six Pareto frontiers include 32 valid architectures. For all architectures the four objectives were calculated. FIG 15 compares the optimization results and the manual design (M) in four quadrants. It is visible that 80% of the solutions **dominate the manual design**. It can be stated that the higher the automation level, the higher the improvement. The maximum decrease is 11.3 kg (30%) in mass, 73 k$ (36%) in SSC, 1850 $/a (75%) in OIC, and 415 $/aircraft (54%) in IPC. All minima are achieved for topology optimization. The minima cannot be achieved in the same solution. Mass and OIC are clearly contradicting, while Mass and SSC, as well as OIC and IPC share some minima. Moreover, the trade-off space is not linear, but shows a big decrease of OIC and IPC for small mass/SSC increase with some knee-points in all device optimizations. Optimizations including the network show a more equal distribution between mass/SSC and OIC/IPC. The runtimes for optimization ranged from the below one minute for signal and device assignment to 10 and 45 minutes for task and device type optimization up to 4 and 48 hours for network and topology optimization on a 3 GHz desktop computer. For the latter only four points of the Pareto optimum could be determined. Runtimes, therefore, strongly correlate with the achieved improvements.

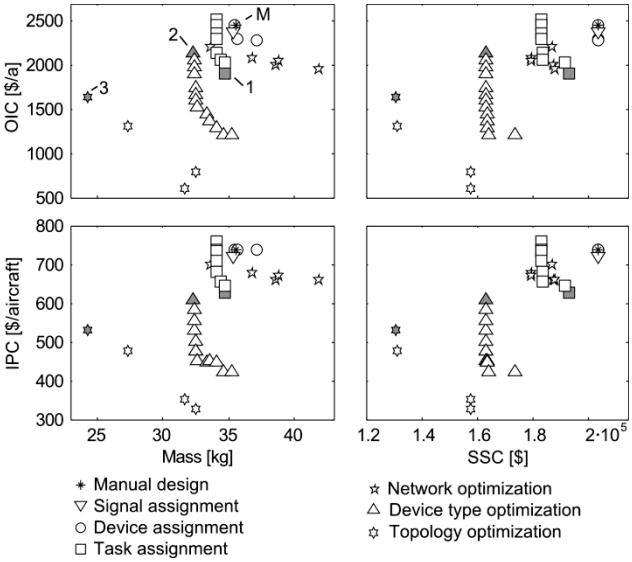

FIG 15: Pareto optimal architectures for all optimization routines

For a better understanding three extreme solutions (1-3) are analyzed in detail. In the following the architectures for the OIC-optimal task assignment, the lightest device type optimization and the mass-optimal topology optimization are given.

The architecture with the lowest OIC in the Pareto optimum of **task assignment** is depicted in FIG 16. Compared to the manual solution one RDC has been removed lowering the OIC. In addition, tasks with high MEL-levels are grouped on two nose and two middle RDC. This creates two additional GO RDC with no OIC. Because the removal of one device and the smart shifting of tasks also mass and SSC are lower than in the manual mapping, although the resources per device type are the same than in the manual design.

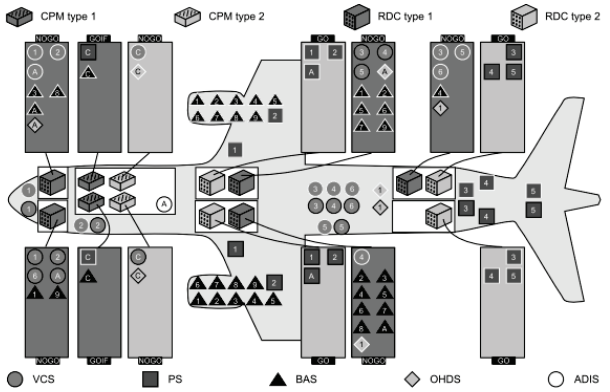

FIG 16: Solution 1 – Task assignment with minimum OIC

A big decrease in device weight is made by **device type optimization**. FIG 17 shows the Pareto optimal solution with the lowest mass and SSC. Obviously the decrease is made by having only six RDCs compared to the ten in the manual design. By resizing the number and types of resources per device type it is possible to increase the usage rate of RDCs from 75% to 95%. The increase in cable weight is more than compensated. Interestingly no CPMs can be removed because segregations are the dominating driver. By removing four RDCs also OIC and IPC are decreased.

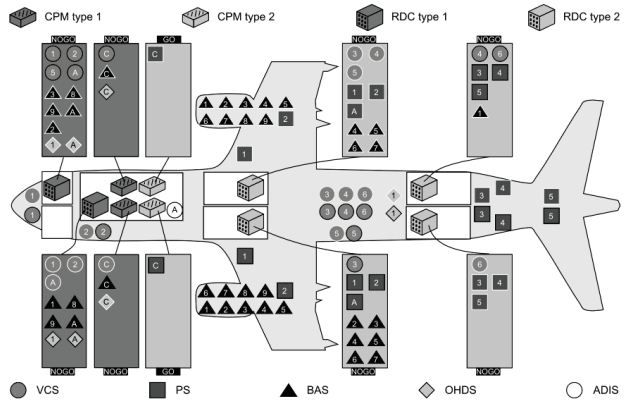

FIG 17: Solution 2 – Device Type optimization minimum mass

From **topology optimization** only the four extreme solutions could be determined because of extremely high runtime. FIG 18 and FIG 19 show the mapping and network topology of the solutions with the overall lowest mass. The decrease of additional 12 kg in mass shows the major influence of network mass and the correlation of task placement and network links. Looking at the mapping in FIG 18 it shows that the same number of device is used as in device type optimization. The device positions and the task allocation is, however, slightly different. Most important, one RDC is moved from the tail to the middle. This saves long links and makes it lucrative to only have four switches in the avionics bay instead of eight in the manual design. Although the mapping and network is asymmetric, the assignment of systems is left-right symmetric. This was not the case for device type optimization and task assignment. Left-right symmetry seems beneficial when targeting smaller networks, which is reasonable since it eases signal segregation.

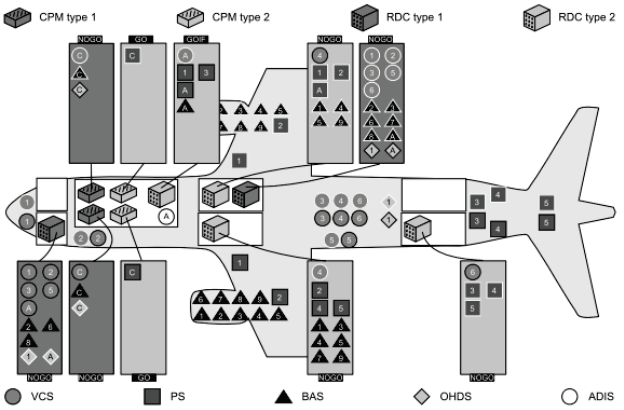

FIG 18: Solution 3 – Topology optimization minimum mass

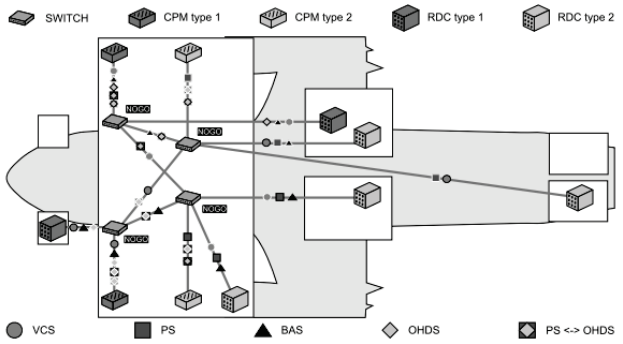

FIG 19: Solution 3 – Topology optimization minimum mass

5. DISCUSSION

Analyzing the results from the optimization studies, it can be stated that the scenario bears high optimization potentials in all objectives, which were not visible manually. It shows that these improvements are especially high if complex and non-traditional objectives as OIC and IPC are considered. Moreover, architectures looking asymmetric or odd on the first glimpse might be optimal, but would never be chosen manually. The assumption that DIMA design issues are dependent and that higher automation level increases the room for improvements is proven. The runtime was acceptable for level one and level two routines. For topology optimization this small excerpt of a complete aircraft hits already the computational feasibility limit. The latter can be weakened by restricting the solutions space. See [29] for a success-full optimization of an A380-like network. However, it is visible that for bigger architectures such complete and unbounded optimization studies are maybe not feasible. Therefore, the framework especially optimization cannot be a replacement for the design engineer, but a valuable helper and sparring partner. In this tandem the proposed method works also on aircraft level sized architectures. Not only can it help to discover improved architectures, but it shows up new architecture variants and insights into objective relations. Moreover, the global optimal approach enables formal justification of architectures even if found manually. The flexibility in scope, objectives, and level of automation allows an adaptation of optimization to architecture sizes and specific designer's needs. This requires, however, an understanding of the designer of optimization routines. This, in addition, increases the trust in solutions.

6. CONCLUSION

In avionics systems based on the DIMA concept standardized avionics modules and network are shared by safety critical system functions. Current architectures host approximately 1000 functions, which cause non-optimal manual selections, sizings, and allocations of DIMA architectures. To support design engineers a **model and algorithm-aided systems architecting framework** for avionics architectures is proposed. The framework comprises a domain specific model and optimization routines, both seamlessly integrated. An architecture model especially for planning enables the independent modeling of systems, hardware, and anatomy, which can be combined to multiple architecture variants. Information is rigid enough for early verification, evaluation and optimization. A set of eight flexible optimization routines is presented, that automates design tasks ranging from function allocation, module selection, and network definition to complete topology generation. The latter are unique in the IMA scope. Routines can be chosen from three automation levels, and are free in their input scope and objectives. The stable and efficient foundations for solving the optimization problems are Binary programs and best-effort MILP-solvers. Moreover, a multi-objective solver extension is provided, that retrieves Pareto optimal architecture sets. Model and optimization are implemented in the **Avionics Architect and its Toolbox**. The applications of six of the optimization routines to an A320-like example of four aircraft systems reveals optimization potentials up to 75% compared to manual design. It showed up the best possible tradeoffs for mass, SSC, OIC, and IPC. Moreover, the resulting architectures showed how mass or cost improvements affect the architecture. Overall, the optimization potential increases with the automation level, but solving time follows. However, if the solution space is manually bounded and objectives are wisely chosen, optimizations can be applied on full-scaled avionics architectures, leading to results, insights, and design justifications, hardly achieved manually.

7. REFERENCES

[1] Martin Halle and Frank Thielecke. Konfigurationsmanagement für Integrierte Modulare Avionik. *Deutscher Luft- und Raumfahrtkongress, Hamburg*, September 2010.

[2] P.J. Prisaznuk. Integrated Modular Avionics. *Aerospace and Electronics Conference*, pages 39–45 vol.1, May 1992.

[3] R. Fuchsen. Preparing the Next Generation of IMA: A new Technology for the SCARLETT Program. *Digital Avionics Systems Conference*, pages 7.B.5–1 –7.B.5–8, October 2009.

[4] C.B. Watkins. Integrated Modular Avionics: Managing the Allocation of Shared Intersystem Resources. *25th Digital Avionics Systems Conference*, pages 1–12, October 2006.

[5] Henning Butz. Open Integrated Modular Avionic (IMA): State of the Art and Future Development Road Map at Airbus Deutschland. *Proceedings of the 1st International Workshop on Aircraft System Technologies*, pages 211–222, March 2007.

[6] Jean-Bernard Itier. IMA1G - Genesis and Results. SCARLETT MOSCOW - 1st Forum, September 2009.

[7] Mark Warner Maier and Eberhardt Rechtin. *The Art of Systems Architecting*. CRC press, 2000.

[8] Christian Fraboul and Frank Martin. Modeling Advanced Modular Avionics Architectures for early Real-time Performance Analysis. *Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing*, pages 181–188, 1999.

[9] S. Forster, M. Fischer, A. Windisch, B. Balser, and D. Monjau. A new Specification Methodology for Embedded Systems Based on the π-calculus Process Algebra. *Rapid Systems Prototyping, 200,* pages 26 – 32, June 2003.

[10] A. Gamatie, C. Brunette, R. Delamare, T. Gautier, and J.-P. Talpin. A Modeling Paradigm for Integrated Modular Avionics Design. *32nd EUROMICRO Conference on Software Engineering and Advanced Applications,* pages 134–143, September 2006.

[11] Julien Delange, Laurent Pautet, Alain Plantec, Mickael Kerboeuf, Frank Singhoff, and Fabrice Kordon. Validate, Simulate, and Implement Arinc653 Systems Using the AADL. *Proceedings of the ACM SIGAda annual international conference on Ada and related technologies*, pages 31–44, 2009.

[12] Michael Lafaye, Marc Gatti, David Faura, and Laurent Pautet. Model Driven Early Exploration of IMA Execution Platform. *Digital Avionics Systems Conference*, pages 7A2–1 –7A2–11, October 2011.

[13] AADL. http://www.aadl.info, 2009.

[14] Object Modeling Group. Omg Systems Modeling Language (OMG SysML). http://www.omg.org/spec/-SysML/1.3/, June 2012.

[15] Laurent Sagaspe, Gerard Bel, Pierre Bieber, Frederic Boniol, and Charles Castel. Safe Allocation of Avionics Shared Resources. *IEEE International Symposium on High-Assurance Systems Engineering*, pages 25–33, 2005.

[16] Laurent Sagaspe and Pierre Bieber. Constraint-based Design and Allocation of Shared Avionics Resources. *26th AIAA-IEEE Digital Avionics Systems Conference*, Dallas, 2007.

[17] P. Bieber, J.P Bodeveix, C. Castel, D. Doose, M.Filali, F. Minot, and C. Pralet. Constraint-based Design of Avionics Platform - Preliminary Design Exploration. *4th European Congress ERTS Embedded Real Time Software*, 2008.

[18] Ahmad Al Sheikh, Olivier Brun, and Pierre-Emmanuel Hladik. Decision Support for Task Mapping on IMA Architecture. *Junior Researcher Workshop on Real-Time Computing (JR-WRTC2009)*, pages 31–34, October 2009.

[19] Uwe Salomon. *Automatic Design of IMA-based Systems*. PhD thesis, Faculty of Aerospace Engineering and Geodesy of the University of Stuttgart.

[20] Shu Zhang. *Communication Infrastructure Supporting Real-Time Applications*. PhD thesis, Technische Universität Hamburg-Harburg, 2008.

[21] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul. Methods for Bounding End-to-end Delays on an AFDX Network. *18th Euromicro Conference on Real-Time Systems,* 2006.

[22] Thanikesavan Sivanthi, Shu Zhang, and Ulrich Killat. A Holistic Framework for Optimal Avionics System Resource Planning. *AST 2007 Workshop on Aicraft System Technologies*, pages 257–268, 2007.

[23] D.C. Carta, J.M.P. de Oliveira, and R.R. Starr. Allocation of Avionics Communication Using Boolean Satisfiability. *IEEE/AIAA 31st Digital Avionics Systems Conference (DASC),* pages 6C1–1 –6C1–12, Octobert 2012.

[24] Ahmad Al Sheikh, Olivier Brun, Maxime Chéramy, and Pierre-Emmanuel Hladik. Optimal Design of Virtual Links in AFDX Networks. *Real-Time Systems*, pages 1–29, 2012.

[25] Bjoern Annighöfer, Ernst Kleemann, and Frank Thielecke. Model-based Development of Integrated Modular Avionics Architectures on Aircraft-level. In *Deutscher Luft- und Raumfahrtkongress, Bremen, September* 2011.

[26] Björn Annighöfer and Frank Thielecke. Supporting the Design of Distributed Integrated Modular Avionics Systems with Binary Programming. D*eutscher Luft- und Raumfahrtkongress, Berlin,* September 2012.

[27] Björn Annighöfer and Frank Thielecke. Multi-objective Mapping Optimization for Distributed Modular Integrated Avionics. *31st Digital Avionics System Conference*, Williamsburg, VA, USA, October 2012.

[28] Bjoern Annighöfer, Ernst Kleemann, and Frank Thielecke. Automated Selection, Sizing, and Mapping of Integrated Modular Avionics Modules. *32st Digital Avionics System Conference*, Syracuse, NY, USA, October 2013.

[29] Bjoern Annighöfer, Caroline Reif, and Frank Thielecke. Network Topology Optimization for Distributed Integrated Modular Avionics. *33rd Digital Avionics System Conference*, Colorado Springs, CO, USA, October 2014 (to appear).

[30] Melih Ozlen and Benjamin A. Burton. Multi-objective Integer Programming: An Improved Recursive Algorithm. *arXiv*, (arXiv:1104.5324v1), 2011.