S. Götz, L. Linsbauer, I. Schaefer, A. Wortmann (Hrsg.): Software Engineering 2021 Satellite Events,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2021 1

A Framework for IMA-based Architecture Design with Unmanned Aerial System (UAS) as a Test Case

Muhammad Salman Akhtar[1], Muhammad Adnan[2]

Abstract: This paper presents a framework for designing, analyzing, and optimizing an Integrated Modular Avionics (IMA) compliant avionics architecture with Unmanned Aerial System (UAS) as a test case. A stepwise approach has been adopted by dividing the design process into smaller and easier to manage modules. The proposed framework covers two major aspects of IMA that are Real-Time Operating System (RTOS) and Communication Protocol. Design steps that include Top Level Design and Detail Design have been further divided into subparts to handle design aspects of IMA complaint RTOS and AFDX communication protocol separately. The proposed framework also includes the analysis part that helps validate and optimize the subject design. The Net2Plan-AFDX, an open-source network analysis tool has been modified and extended for calculating and analyzing End-to-End delays, jitter, goodput, and throughput.

Keywords: IMA, UAS, RTOS, AFDX, Latency, Jitter, Goodput, Throughput, Net2Plan-AFDX.

1 Introduction

IMA or IMA like architectures have been successfully implemented for larger aircraft such as Airbus A380, Boeing 787 Dreamliner, Lockheed Martin F-22, and Lockheed Martin F-35 [Ch94]. IMA promises to provide a safe and secure environment to application software through logical partitioning of the operating system and hardware resources. Moreover, IMA also optimizes the Size, Weight, and Power (SWaP) through computational and communication resource sharing. Since smaller aerial platforms or UASs have more rigid SWaP requirements, therefore, it would be logical to extend the IMA concept for these platforms as well. [EAF05] proposed a distributed modular architecture for small UAS composed of a set of computing modules communicating through CAN bus. The proposed design is targeted for mini or micro UAS. The design is kept modular but it uses a microcontroller as a processing unit and therefore no RTOS was employed. [ERG05] investigated a partial IMA architecture for the Queensland University of Technology (QUT) research UAV, based on a collection of dedicated processors communicating through the CAN bus, RS232, and Ethernet. The proposed design involves Linux and QNX as Operating Systems (OS). Both these OSs are not IMA compliant and lacks partitioning. [Lo07] further investigated the flexible and low-cost solution for UAS based on middleware (MAREA) which provides an easy to use interface for a network programmer in a Publish-Subscribe scenario using Data

[1] Air University, IAA, E-9, Islamabad, 44000, salman.866@gmail.com,
https://orcid.org/0000-0003-2439-5871

[2] Air University, IAA, E-9, Islamabad, 44000, madnan@mail.au.edu.pk

Distribution Service (DDS). The underlying protocol is based on commercial Ethernet therefore, the proposed design does not meet the real-time constraints and lacks details about processing nodes. [Jo12] implemented control and mission software partitioned under Linux-based ARINC 653. However, their work does not encompass communication architecture since everything was implemented on a single processor.

Existing research work covers the IMA implementation to some extent but no research work undertook the IMA study for UAS comprehensively. The proposed UAS are of small sizes mostly between 10 to 20 kilograms, classified as mini or micro UAS, having a small range and loiter times with small avionics package requirement [KI13]. Moreover, they use network protocols like CAN bus which has limited data rates of the order of 1 Mbps [La08], or Ethernet-based middleware which lacks determinism.

Research studies have also been conducted for modeling and simulating the AFDX network. [JWR11] simulated and analyzed the real-time performance of the AFDX network using OPNET. They have modeled the End System comprising of three modules: data generation, data reception, and protocol module. They calculated and analyzed the end-to-end delays but their study lacked jitter calculation. Similar work was also undertaken by [SZA15] in which they investigated the effects of switching delays of varying frame size on network performance. They also used OPNET for modeling the AFDX network and added integrity and redundancy checking in the receiving part of the End System. They used the simulation model on a case study presented in [La12] for calculating end-to-end delays. However, their research work also did not include the jitter experienced by the data packets in the network. [Do10] used Network Simulator (NS 2) as a platform to simulate the AFDX network. They modeled the End System, Switch, and protocol stack and calculated latency and jitter using the simulation. However, the modeled End System only supports one Virtual Link. Another attempt on the subject was undertaken by [FBP16] in which they extended the open-source networking tool Net2Plan for the AFDX network which is also an open-source work. They calculated the worst-case delay through the network calculus and trajectory approach algorithm. The tool provides an excellent interface for rapidly modeling, simulating, and analyzing an AFDX network. However, the tool tends to give a little pessimistic result as compared with the result suggested by [So17].

This paper proposes an elaborate framework for conceptualizing avionics architecture that is based on IMA using UAS as a test case. The framework encompasses both the Operating System and communication layer aspects of the IMA architecture. Moreover, we also extended the work on Net2Plan-AFDX by Fernandez et al by identifying the source of pessimism and later on rectified it through algorithm modification. Additionally, parameters like the Goodput and Throughput calculation were also included in the software tool.

2 A Framework for the IMA based architecture

A stepwise approach has been adopted that follows a top-down model. The modules at

the higher layer are agnostic of the details of lower-layer modules. As we go further and deeper into the framework, the details related to the modules start to expand.

2.1 Top-level design

In top-level design, the whole avionics system is taken as a single system and then divided into smaller subsystems. The subsystems‘ functionality is briefly elaborated and their mutual interactions are described in this step. Moreover, avionics architecture and configuration are finalized in this step.

Identification of avionics subsystems

The required avionics package is determined based on the role and task of the aircraft and specifications demanded by the user. The avionics package is selected to achieve the desired mission objectives. As a test case for the proposed UAS following avionics systems have been identified:-

1. Communication System
2. Navigation System
3. Identification System
4. Vehicle Management System
5. Propulsion Management System
6. Flight Control System
7. Mission Management System

Identification of interface and interaction between subsystems

Modern avionics systems do not work in isolation; they interact with neighboring avionics systems by exchanging data. It is this interaction and data exchange that results in achieving far more functionalities than if the systems would have worked in isolation. Here the type of data along with source and destination is determined to map the interaction. Table 1 shows the data flow for some of the subsystems of UAS.

Source	Destination	Description
Comm System	Flight Control System	Control Commands
Navigation System	Comm System	Latitude, Longitude, Altitude, etc.
Vehicle Mgt System	Mission Mgt System	Status messages and Operating Parameters (Hydraulic pressures, electrical load, fuel status, etc.)
Mission Mgt System	Comm System	Payload Status, Targeting

Tab. 1: Interaction between subsystems of UAS

Architecture and configuration of the avionics system

Integrated Modular Avionics (IMA) has been adopted as avionics architecture for the avionics suite of Unmanned Aerial System. For this purpose, systems have been divided into three computing units and some distributed sensor units. Table 2 shows the

distribution of systems into IMA partitions among these computing units.

Computing Unit	Partitions
Nav Computer	Sensor Fusion, IFF, TCAS
Vehicle Mgt Computer	FCS, Autopilot, Fuel, Electrical, Hydraulic, Propulsion, ECS, VHMS
Mission Mgt Computer	Targeting Pod, Data Link, Data Loader, Payload Management, Data Storage
Distributed Sensors	Air Data Sensors, TACAN, INS, Radio Altimeter, GPS, AHRS

Tab. 2: Avionics Systems Distribution among Computing Units for the UAS

2.2 Detail design

After going through requirement gathering, identification of specifications, and top-level design, basic design architecture is available. This basic architecture is further elaborated through the Detail design. Following are the major steps of detail design.

Identification of partition attributes

Partition attributes are dependent upon the processes within that partition. The individual process periods and time capacities within a particular partition decide the Partition Period and Partition Duration. Partition Duration is the time required by a partition for the execution of all its processes which comes out to be the sum of time capacities of processes. The Partition Period is the time after which that partition is scheduled again which is the Greatest Common Divisor (GCD) of the periods of processes. Process Period can be taken as the minimum time after which the process interacts with another process of the same partition or sometimes with a process of another partition. The Time Capacity of a process is determined by equation 1.

$$TimeCapacity = \frac{(NumberofInstructions)x(CPI)}{ProcessorClockSpeed} \quad (1)$$

The number of Instructions is determined by the complexity of the process whereas processor clock speed is determined by the selected processor. Clocks per Instruction (CPI) are dependent upon the processor architecture, compiler, and the type of instructions used by the application programmer [PH13]. Average CPI can be estimated in the case of unavailability of the required data. An automated tool has been developed in Microsoft Excel spreadsheet which requires inputs such as Process period, the number of instructions required by the process, average CPI, and processor clock speed whereas the outputs from this tool are Partition Period and Partition Execution time. Figure 1 shows the interface of the IMA Partition Automation Tool in which the required partition comprises four processes.

IMA Partition Automation Tool

Name	Period (ms)	Number of Instructions	Average Cycles Per Instruction (CPI)	Processor Clock (MHz)	Time Capacity (ms)	Partition Period (ms)	Partition Execution Time (ms)
Process 1	8	90000	1.45	250	0.522	8	2.001
Process 2	16	110000	1.45		0.638		
Process 3	128	75000	1.45		0.435		
Process 4	32	70000	1.45		0.406		
Process 5	0	0	0		0		
Process 6	0	0	0		0		
Process 7	0	0	0		0		
Process 8	0	0	0		0		
Process 9	0	0	0		0		
Process 10	0	0	0		0		

Note: Enter 0 as time period and 0 as execution time for Processes not used

Fig. 1: IMA Partition Automation Tool

Using this Partitioning tool and the interaction between partitions from table 1, all the required attributes of the UAS partitions from table 2 are calculated.

Communication architecture

Earlier avionics data networks were peer to peer like Tornado serial bus, then came single source to multiple sinks like ARINC 429, then came multiple sources to multiple sinks like ARINC 629 and MIL-STD 1553 (master-slave). A more recent advancement to avionics data networks is Avionics Full Duplex Switched Ethernet (AFDX) based on the ARINC 664 Part 7 [AF05]. Figure 2 shows the communication architecture for the UAS.

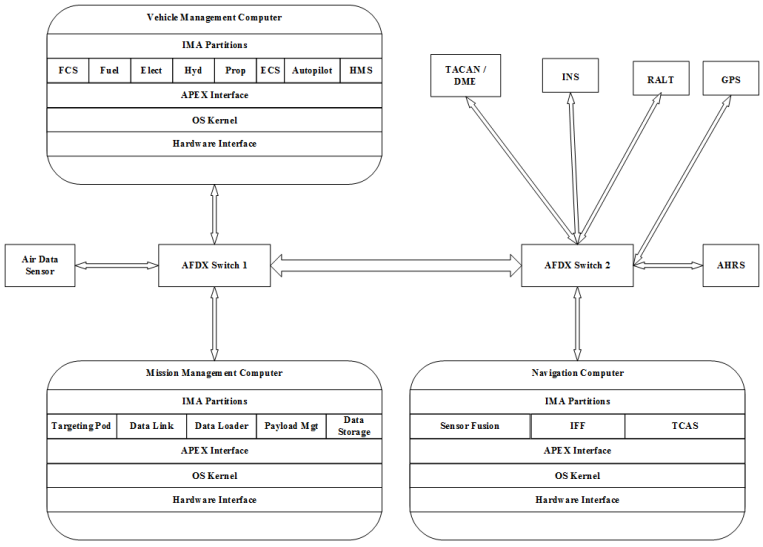

Fig. 2: Communication architecture of the UAS

Inter-partition interaction

Regardless of the communication scheme used, a few common parameters have to be determined before designing any data network for the aircraft avionics system. These basic parameters give an initial estimate of the requirements and suitability of the intended scheme. These parameters include the size of the individual data packet and its period which are then used to calculate the required bandwidth. This approach is used for calculating the data rate requirement for all the partitions of UAS. Table 3 shows inter-partition communication (IPC) of the UAS through the AFDX network for some partitions.

Source	Dest	Freq (Hz)	BAG	Lmax	Lmax + overhead	Data Rate (Kbps)	VL ID
Sensor Fusion	FCS	62.5	0.016	200	267	130.37	1
TCAS	Autopilot	125	0.008	50	117	114.26	4
Air Data	Sensor Fusion	62.5	0.016	150	217	105.96	5
FCS	Data Storage	31.25	0.032	400	467	114.01	12
Data Loader	FCS	31.25	0.032	50	117	28.56	28
INS	Sensor Fusion	125	0.008	50	117	114.26	31
Air Data Sensor	Propulsion	62.5	0.016	150	217	105.96	40
HMS	Air Data Sensor	15.625	0.064	25	92	11.23	41

Tab. 3: UAS IPC through AFDX Network

2.3 Analysis

IMA compliant RTOS provides an execution environment for the application software of avionics subsystems whereas communication protocol provides a mechanism for data sharing between these subsystems. Once the architectural design of the avionics system has been finalized, it would be logical to analyze this design.

Real-Time Operating System attributes analysis

The Partition Periods and Partition Execution Times determine Minor Frame Period and Major Frame Period which are then used for checking the validity of the intended scheme. Minor Frame Period is the minimum Partition Period among all Partitions while the Major Frame Period is the Least Common Multiple (LCM) of the Partition Periods. For a valid Partition scheduling, the Execution Time of Partition should be less than the

Minor Frame Period. Similarly, for complete module validity, the sum of all Partition Execution Times should be less than the Minor Frame Period.

A Microsoft Excel-based "IMA Partition Analysis Tool" has been designed for calculating Major and Minor Frame Periods. Additionally, Partitions are checked for individual validity as well as combined validity. Partition Period and Partition Execution Time are the input while the Minor Frame Period, Major Frame Period, and Percentage of the processor used is output. Moreover, the tool also outputs the validity statuses of the individual partitions and the whole module. Figure 3 shows the IMA Partition Analysis Tool results of Mission Management Computer while Figure 4 shows the scheduling details for the Mission Management Computer.

IMA Partition Analysis Tool
(Mission Management Computer)

Partition	Period (ms)	Execution Time (ms)	Minor Frame Period (ms)	Partition Schedulable Status	Maximum Required Execution Time (ms)	Major Frame Period (ms)	Module Schedulable Status	Max % of Processor Usage in Minor Frame	% of Processor Usage in Major Frame
Tgt Pod	16	2	8	Valid	8	32	Valid	100	53.125
Data Link	8	2		Valid					
Data Loader	32	1		Valid					
Payload Mgt	32	1		Valid					
Data Storage	16	1		Valid					
Health Monitor	32	1		Valid					

Note: Enter 1 as time period and 0 as execution time for Partition Not Used
Maximum Required Execution Time should be less than Minor Frame Period

Fig. 3: IMA Partition Analysis Tool (Mission Management Computer)

Mission Management Computer
(Partition Scheduling)

Major Frame (32 ms)

Data link	Tgt Pod	Data Storage	Data Loader	Payload Mgt	HM	Data link	Idle	Data link	Tgt Pod	Data Storage	Idle	Data link	Idle
2	2	1	1	1	1	2	6	2	2	1	3	2	6
1st Minor Frame (8 ms)						2nd Minor Frame (8 ms)		3rd Minor Frame (8ms)				4th Minor Frame (8ms)	

Fig. 4: Partition Scheduling (Mission Management Computer)

Communication protocol analysis

In our proposed scheme, we have used AFDX as a data network which is further analyzed based on performance parameters such as latency, jitter, Throughput, and Goodput. Latency is one of the requirements of an avionics network that maximum end to end delay remains within the bound. Network delay is dependent upon factors like processing, queuing, transmission, and propagation delays. Jitter is another key parameter that is associated with network performance. Jitter accounts for the variation in the latency of data packets. For deterministic networks, the jitter must also be bounded like latency. AFDX specifies the upper limit for jitter to be

$$maxjitter \leq 40\ \mu s + \frac{\sum_{j\in(set\ of\ VLs)}(20 + Lmax_j)\ X\ 8}{Nbw} \tag{2}$$

$$maxjitter \leq 500\ \mu s \tag{3}$$

Here N_{bw} is the Network Bandwidth which is 100 Mbps for AFDX and L_{max} is the size of the maximum data packet of a particular Virtual Link (VL). Maximum jitter for AFDX network must be less than the lower value of equations 2 and 3.

The Net2Plan-AFDX tool has been modified and extended for measuring and analyzing these network performance parameters. The tool tends to give pessimistic results; therefore it was modified to remove the source of pessimism. We have implemented improved algorithms for the calculation of end to end delays and jitter using Network Calculus and Trajectory Approach algorithms as well as incorporated goodput and throughput calculations.

Modeling AFDX Network using Network Calculus and Trajectory Approach

In AFDX, Virtual Links (VLs) are used for transferring data packets from a single source to single or multiple destinations. AFDX constraints the VLs with maximum packet size L_{max} and minimum transmission gap between data packets known as Bandwidth Allocation Gap (BAG). These constraints enable AFDX VLs to be modeled as a leaky bucket arrival curve $\alpha(\frac{L_{max}}{BAG}, L_{max})$. A switch routing port can be modeled as rate latency service curve $\beta(r, T)$. Here r is the bandwidth of the link and T is the delay that the switch induces while routing data packets.

While using the Trajectory approach, every intermediate switch is considered as a network node. An inherent switching latency due to technological constraints is fixed at 16 µs [So17]. VL path of AFDX network is considered as a flow for Trajectory approach. The processing delay of a single data packet is given by equation 4.

$$Packet\ Processing\ Time = \frac{Size\ of\ Data\ Packet}{Link\ Bandwidth} \tag{4}$$

Net2Plan-AFDX

The Net2Plan-AFDX is an open-source tool developed for modeling and analyzing the AFDX data network [FBP16]. It is developed by extending the Net2Plan which is another open-source Java-based network analysis tool [PZ15]. Currently implemented algorithms in Net2Plan-AFDX give pessimistic results as compared to other published works [So17][Ch06]. Through a detailed analysis, it was observed that the algorithm always add processing delay for an additional data packet for the second hop. The algorithms for both Network Calculus and Trajectory Approach have been modified to cater for requisite change. The modified algorithms also incorporated Pay Burst Only Once (PBOO) which means that if one VL is competing for resources with another VL in a network node then it will affect the latency only once. Figure 5 shows the pseudo-code for Network Calculus and Trajectory Approach algorithms respectively. The Net2Plan-AFDX-Extended version is used for analyzing and validating the Inter-Partition Communication of UAS.

```
Foreach (NumberOfHops)
{
    Foreach (NumberOfVLCrossing)
    {
        If (VL has not already crossed)
        {
            NumberOfBytes =+ Lmax_i
        }
        Else
        {
            Continue
        }
        ListOfAlreadyCrossedVLs = VL_i
    }
    NumberOfBytes =+ DeltaPacketBytesOfArrivalCurve
    NodeLatency = (NumberOfBytes / LinkCapacity) x 8
    DeltaPacketBytesOfArrivalCurve = (NodeLatency x Lmax_i) x 8 / VL_BAG
    Latency =+ NodeLatency
}
```

```
Foreach (NumberOfHops)
{
    Foreach (NumberOfVLCrossing)
    {
        If (VL has not already crossed)
        {
            NumberOfBytes =+ Lmax_i
        }
        Else
        {
            Continue
        }
        ListOfAlreadyCrossedVLs = VL_i
    }
    Latency =+ (NumberOfBytes/LinkCapacity) x 8
}
```

Fig. 5: Pseudo Code {Network Calculus (Left)} {Trajectory Approach (Right)}

Table 4 depicts a few of the resultant Link utilization in the UAS AFDX network. It can be seen that the link between Switch_1 and Vehicle Management Computer has a maximum Link utilization of 1.04% and a maximum number of VLs pass through this link. No link utilization exceeds the maximum link capacity which is 100 Mbps.

Origin Node	Destination Node	Crossing VLs	Capacity (Mbps)	Occupancy (Kbps)	Percent of Occupancy
Switch_2	Switch_1	6	100	423.5	0.42
AirDataSensor	Switch_1	3	100	349	0.35
Switch_1	Vehicle Mgt Comp	14	100	1039.5	1.04
Mission Comp	Switch_1	10	100	636.75	0.64
Switch_1	Mission Comp	14	100	1005.5	1.01
Navigation Comp	Switch_2	11	100	493.38	0.49
Switch_2	Navigation Comp	7	100	507.88	0.51

Tab. 4: Link Utilization of UAS AFDX network

Figure 6 shows that the links which are handling more number of VLs have a higher percentage of the Link utilization which is intuitive since more number of VLs means more data traffic passing through that Link. However, there is no certain relationship since different VLs can have different L_{max} and BAG values which decide the data rate of a VL. This effect can be seen in Link 8 and Link 9. Although, Link 8 has 11 VLs as compared to Link 9 which has 7 VLs still Link 9 has higher Link utilization of 0.51% as compared to Link 8 that has 0.49%.

Figure 7 and 8 depicts the comparison of latency and jitter respectively. The difference of results for the case of Network Calculus and Trajectory Approach are so small that they seem to overlap.

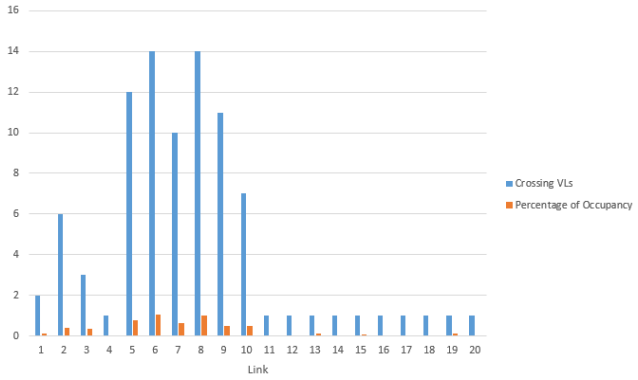

Fig. 6: Effect of Number of VLs Crossings on Link Occupancy

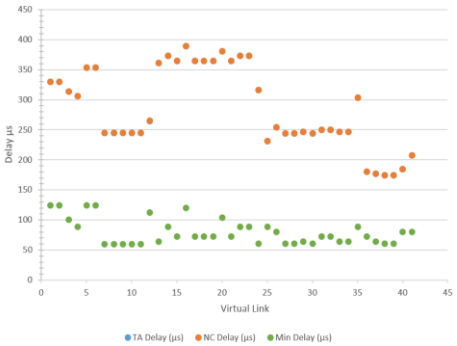

Fig. 7: End to End Delays of UAS

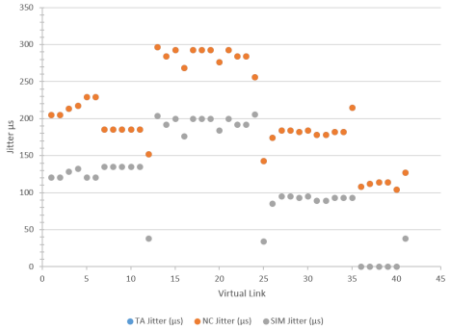

Fig. 8: Jitter in UAS data network

Figure 9 shows the Throughput and Goodput comparison. The analysis reveals that efficiency is greater for VLs with higher values of L_{max} which is intuitive since protocol overhead due to headers and Inter Frame Gap (IFG) remain constant for all VLs irrespective of L_{max}. Thus VLs with lower values of L_{max} are affected more in terms of Goodput.

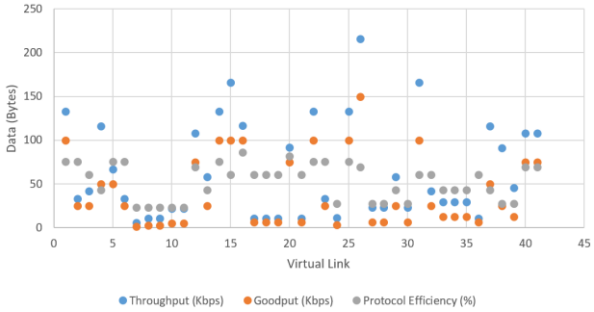

Fig. 9: Throughput and Goodput Comparison of UAS

3 Conclusion

This study presents a framework that would help implement and analyze an IMA complaint avionics architecture. The Avionics package for a UAS is taken as a test case for demonstrating the applicability of the proposed framework. We explored both RTOS as well as communication layer aspects of the IMA architecture. This research work elaborates on each step of the design process with the help of appropriate examples where necessary. The main steps, Top Level Design and Detail Design are further divided into sub-steps for segregating different aspects of the design process. The division of the design process into sub-steps makes the flow more logical and manageable. Figure 10 shows the complete design flow of the proposed IMA framework.

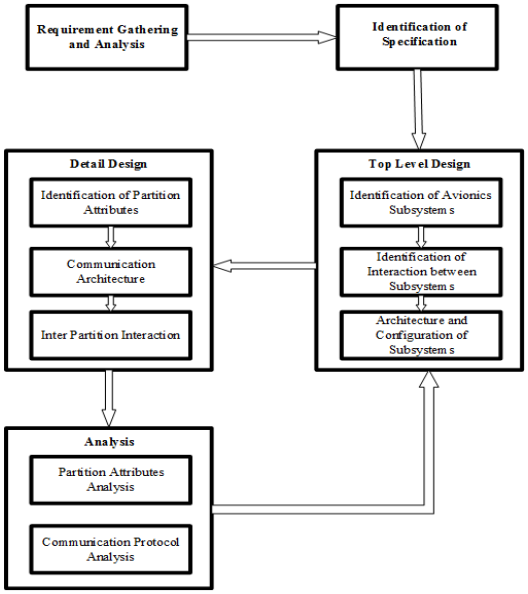

Fig. 10: Proposed IMA Framework

Bibliography

[Ch94] Lt Col Chuck Pinney, Joint Advanced Strike Technology Program, Avionics Architecture Definition, Version 1.0, 1994.

[EAF05] Elston; Argrow; Frew: A Distributed Avionics Package for Small UAVs. AIAA Conference, Arlington, Virginia. 2005.

[ERG05] Ellen; Roberts; Greer: An investigation into the Next Generation Avionics Architecture for the QUT UAV Project. Goh, Roland & Ward, Nick (Eds.) Smart Systems 2005 Postgraduate Research Conference, Brisbane 15 December 2005.

[Lo07] López, J et.al. A Middleware Architecture for Unmanned Aircraft Avionics. Middleware'07, Newport Beach, California, November 26-30 2007.

[Sa07] Samuel, A et.al. Subsystem Design and Integration of a Robust Modular Avionics Suite for UAV Systems Using the Time-Triggered Protocol (TTP). SAE Aerospace Technology Conference, 2007.

[Jo12] Jo, H et.al. Implementing control and mission software of UAV by exploiting open-source software-based ARINC 653. 2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC), Williamsburg, VA, 2012.

[KI13] Korchenko, A.G; Illyash, O.S: The Generalized Classification of Unmanned Air, IEEE 2nd International Conference "Actual Problems of Unmanned Air Vehicles Developments" Proceedings, 2013.

[La08] Lawrence M. Thompson, Industrial Data Communications, 4th Edition, (2008). The Instrumentation, Systems, and Automation Society, ISBN: 13: 978-1-937560-88-1.

[JWR11] Jiqiang; Weimin; Ronggang: Study on Real-time Performance of AFDX Using OPNET, 2011 IEEE, 978-1-4577-0860-2/11.

[SZA15] SafWat; Zekry; Abouelatta: Avionics Full-duplex switched Ethernet (AFDX): Modeling and Simulation, 32nd National Radio Science Conference (NRSC 2015), March 24-26, 2015.

[La12] Lauer, M. (2012). Dne methode glob ale pour la verification d'exigences temps reel: application a l'AvioniqueModulaire Integree (Doctoral dissertation, Institute National Poly technique de Toulouse-INPT).

[Do10] Dong, S et.al. The Design and Implementation of the AFDX Network Simulation System, 2010 IEEE, 978-1-4244-7874-3/10.

[FBP16] Fernandez; Burrull; Pavon, Net2Plan-AFDX: An open-source tool for optimization and performance evaluation of AFDX networks, 2016 IEEE, 978-1-5090-2523-7.

[So17] Soni, A et.al. Work In Progress Paper: Pessimism analysis of Network Calculus approach on AFDX networks, 2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES).

[PH13] Patterson, David; Hennessy, John: Computer Organization and Design (The Hardware / Software Interface), 2013, Fifth Edition, ISBN: 978-0-12-407726-3.

[AF05] AFDX / ARINC 664 Tutorial (1500-049), May 2005, Condor Engineering, Inc, Document Version: 3.0.

[PZ15] Pavon; Zaragoza: Net2Plan an open-source network planning tool for bridging the gap between academia and industry, IEEE Networks Magazine, Vol. 29, Sept-Oct. 2015.

[Ch06] Charara, H et.al. Methods for bounding end-to-end delays on an AFDX network. In Proceedings of the 18th ECRTS, Dresde, Germany, July 2006.