Umut Durak · Jürgen Becker
Sven Hartmann · Nikolaos S. Voros
Editors

Advances in Aeronautical Informatics

Technologies Towards Flight 4.0

Springer

Advances in Aeronautical Informatics

Umut Durak · Jürgen Becker
Sven Hartmann · Nikolaos S. Voros
Editors

Advances in Aeronautical Informatics

Technologies Towards Flight 4.0

Springer

Editors
Umut Durak
Institute of Flight Systems
German Aerospace Center (DLR)
Braunschweig
Germany

Sven Hartmann
Department of Informatics
Clausthal University of Technology
Clausthal-Zellerfeld
Germany

Jürgen Becker
Institute of Information Processing
Karlsruhe Institute of Technology (KIT)
Karlsruhe
Germany

Nikolaos S. Voros
Computer and Informatics
Engineering Department
Technological Educational Institute
of Western Greece
Nafpaktos
Greece

ISBN 978-3-319-75057-6 ISBN 978-3-319-75058-3 (eBook)
https://doi.org/10.1007/978-3-319-75058-3

Library of Congress Control Number: 2018937708

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To the pioneers of flight,
who made a dream of mankind come true.

To the pioneers in computer science,
who opened us new frontiers of innovation.

Foreword

Since the very beginning of manned flight more than hundred years ago, aeronautics was one of the most advanced fields for generating new technologies. Requirements against safety, reliability, lowest weight, and pilot integration put the highest demand on the engineers to find solutions in these conflicting areas.

The introduction of new navigation and air data sensors and power amplification by pneumatic and hydraulic systems opened the arena for large aircraft and first automated flight segments. After WWII, onboard computing opened the scope for new support functions, auto flight systems, and many more capabilities of modern aircraft. System integration became one of the key elements during design and development of aircraft and step-by-step software design became one of the most important areas in defining the functional structures of new aircraft. The art of designing these aircraft underwent an evolutionary change toward software integrated systems of systems. Today, we are at the threshold to highly automated and unmanned autonomous systems that will pose even more emphasis on information technology and pushing the limits to the highest criticality levels possible.

The Institute of Flight Systems at the German Aerospace Center (DLR) is a leading research institution of flight sciences and airborne systems technologies with strong links to industry and worldwide research. Our involvement in the most advanced aircraft designs and our role in the development of future autonomous systems and aeronautics regulations brought us early to investigate the impact of new computing architectures like multi-core platforms, high-speed reliable networking, data sciences, and semantic infrastructures such as ontologies on airborne systems. Safety critical systems will have to host real-time capable decision-making software that is suited for highly automated verification and qualification.

Information science has developed in the past years with a stunning speed and it is now high time that the links between this science and modern aeronautics are investigated and discussed.

This book is providing a profound compilation of chapters from experts of information communication technologies and aeronautics who share their view about the advances in aeronautical informatics. I would like to thank the authors to present this book which closes an important gap in the literature.

Braunschweig, Germany
February 2018

Stefan Levedag
Director, Institute of Flight Systems
German Aerospace Center (DLR)

Preface

Aeronautical informatics is a cross-disciplinary field that involves aeronautics and computer science. We are witnessing the evolution of Information and Communication Technologies (ICT) through various disruptive innovations that create new paradigms and change our lives. The impact of this evolution is evident on many technical systems. Industry 4.0 refers to this evolution and designates the new era with the keywords “smart” and “connected”.

The impact of ICT on how we design and fly aircraft is observable with respect to the progress in aeronautical informatics. In this book, we tried to have a closer look into the advances in this area. The book is organized in *Introduction*, *Information and Communication Technologies Supporting Flight 4.0*, and *The Challenges* sections.

The *Introduction* encompasses Chap. 1 from Umut Durak where he tries to establish a base for the book by elaborating the evolution of aeronautics parallel with the other technical domains. Thereby, in relation to advances in ICT, he introduces the fourth revolution in aeronautics as Flight 4.0.

In *Information and Communication Technologies Supporting Flight 4.0*, there are six chapters that address six fields of advancement. Falco K. Bapp and Jürgen Becker present advances in avionic platforms with the breakthrough in multi-core systems in Chap. 2. Emerging trends in avionics networking are addressed in Chap. 3 by Andreas Reinhardt and Aysegul Aglargoz. In Chap. 4, Christos P. Antonopoulos, Konstantinos Antonopoulos, and Nikolaos S. Voros discuss Internet of Things and Service-Oriented Architecture as the infrastructures for Flight 4.0. Gerrit Burmester, Hui Ma, Dietrich Steinmetz, and Sven Hartmannn present big data and data analytics concepts applied to aeronautics in Chap. 5. In Chap. 6, Carlos Insaurralde and Erik Blasch address utilization of ontologies in aeronautics. The last contribution of this section is Chap. 7 from Shafagh Jafer, Umut Durak, Hakan Aydemir, Richard Ruff, and Thorsten Pawletta. They review the advances in software engineering and their reflections in aeronautics.

The *Challenges* section is composed of three chapters. In Chap. 8, Christoph Torens and Johann C. Dauer and Florian Adolf discuss autonomy and corresponding safety issues particularly in unmanned aircraft domain. Reinhard

Wilhelm, Jan Reineke, and Simon Wegener extend the section with Chap. 9 that elaborates challenges in tackling the real-time requirements as we move toward multi-core avionic platforms. In the last chapter, Ella M. Atkins proposes an expansion to aerospace engineering curricular in order to incorporate aeronautical informatics.

We believe that the notable contribution of this book is highlighting aeronautical informatics as a field of research by providing a comprehensive array of chapters that render various recent advancements in information and communication technologies and their effect on aeronautics. It emphasizes the change in technology landscape of aeronautics as revolutionary. The upcoming era of "smart" and "connected" flight is named as Flight 4.0.

We invite the reader to this unique collection from eminent contributors who elaborate the advancement in their respective fields and explore their applications in aeronautics. We further encourage the reader to contribute for the development of this flourishing multidisciplinary field, aeronautical informatics.

Braunschweig, Germany	Umut Durak
Karlsruhe, Germany	Jürgen Becker
Clausthal-Zellerfeld, Germany	Sven Hartmann
Antirio, Greece	Nikolaos S. Voros

Acknowledgements

This book effort is initiated and partially supported in the context of the ARGO Project (http://www.argo-project.eu/) that is funded by the European Commission under Horizon 2020 Research and Innovation Action, Grant Agreement Number 688131.

Contents

Contributors

Florian Adolf German Aerospace Center (DLR), Braunschweig, Germany

Aysegul Aglargoz German Aerospace Center (DLR), Braunschweig, Germany

Christos P. Antonopoulos Technological Educational Institute of Western Greece, Antirio, Greece

Konstantinos Antonopoulos Technological Educational Institute of Western Greece, Antirio, Greece

Ella M. Atkins University of Michigan, Ann Arbor, MI, USA

Hakan Aydemir Turkish Aerospace Industries (TAI), Ankara, Turkey

Falco K. Bapp Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Jürgen Becker Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Erik Blasch US Air Force Research Lab, Rome, NY, USA

Gerrit Burmester Clausthal University of Technology, Clausthal-Zellerfeld, Germany

Johann C. Dauer German Aerospace Center (DLR), Braunschweig, Germany

Umut Durak German Aerospace Center (DLR), Braunschweig, Germany

Sven Hartmannn Clausthal University of Technology, Clausthal-Zellerfeld, Germany

Carlos C. Insaurralde Teesside University, Middlesbrough, UK

Shafagh Jafer Embry Riddle Aeronautical University, Daytona Beach, FL, USA

Hui Ma Victoria University of Wellington, Wellington, New Zealand

Thorsten Pawletta Wismar University of Applied Sciences, Wismar, Germany

Jan Reineke Saarland University, Saarbruecken, Germany

Andreas Reinhardt Clausthal University of Technology, Clausthal-Zellerfeld, Germany

Richard Ruff The MathWorks, Dallas, TX, USA

Dietrich Steinmetz Clausthal University of Technology, Clausthal-Zellerfeld, Germany

Christoph Torens German Aerospace Center (DLR), Braunschweig, Germany

Nikolaos S. Voros Technological Educational Institute of Western Greece, Antirio, Greece

Simon Wegener AbsInt Angewandte Informatik GmbH, Saarbruecken, Germany

Reinhard Wilhelm AbsInt Angewandte Informatik GmbH, Saarbruecken, Germany; Saarland University, Saarbruecken, Germany

Part I
Introduction

Chapter 1
Flight 4.0: The Changing Technology Landscape of Aeronautics

Umut Durak

Abstract This chapter draws the readers into a comprehensive discussion about the advances in Information and Communication Technologies (ICT) and their influence on the technology landscape of aeronautics. It gives a rough overview of the advances in technical systems from the industrial revolution up until Industry 4.0 and elaborates the reflection of these advancements in aeronautics from the pioneers era toward Flight 4.0. It briefly describes various recent fields of research in ICT such as Cyber-Physical Systems (CPS), Internet of Things (IoT), wireless networks, multi-core architectures, Service-Oriented Architecture (SOA), cloud computing, big data, and modern software engineering methodologies as the parts of future aeronautical engineering body of knowledge. Thereafter, it describes aeronautical informatics as an establishing interdisciplinary field of study of applied informatics and aeronautics.

1.1 Aeronautics: The Study of Flight

Aeronautics is defined as the study or the practice of all aspects of flight through the air [1]. It also refers to design, construction, and operation of aircraft [2]. Aeronautical engineering is the corresponding engineering discipline. It applies the scientific principles of flight and engineering in design and development of aircraft and its operation. Aerospace engineering extends the limits of aeronautical engineering with including space flight and astronautics into its scope.

Encyclopedia of Aerospace Engineering from Wiley documents the aspiration of the largest professional organizations of aeronautics, namely Royal Aeronautical Society (RAeS) and the American Institute of Aeronautics and Astronautics (AIAA) in seeking the body of aerospace knowledge [3]. This large-scale reference that covers entire range of scientific and engineering principles of aeronautics and astronautics is organized in eight volumes: fluid dynamics and aerothermodynamics, propulsion and power, structural technology, materials technology, dynamics and control,

U. Durak (✉)
German Aerospace Center (DLR), Braunschweig, Germany
e-mail: umut.durak@dlr.de

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_1

environmental impact, manufacturing and operations, vehicle design and systems engineering. This classification provides a comprehensive list for the fields of study in aeronautics in the classical sense.

1.2 The Evolution of Aeronautics

The term technical systems refers to all man-made artifacts, objects, products, tools, and technical works that are a result of a manufacturing activity [4]. Hubka describes evolution of technical systems starting from the early times of machines where each machine is perceived individually as a whole. The studies about the common elements of machines started with the establishment of polytechnical schools in late eighteenth century. Nineteenth century brought the systematic studies of machine elements and mechanisms. The underlying commonalities and patterns across various types of machines ranging from weapons to mining machines and steam engines to aircraft were studied. Engineering is established as the study of machines.

Engineering enabled the age of machinery, the transition from manpower to machine power in various areas. Its effect in production of goods is accepted as a revolution. The transition from hand production methods to machine production is named as industry revolution. Engineering started the pioneers era in aeronautics. Otto Lilienthal [5] as one of the most famous pioneers of this era applied the engineering principles to unpowered airplanes and made the fist successful flight with his glider. The next remarkable step was the success of Wright brothers [6] with the powered aircraft. Among others, these pioneers paved the way to the establishment of aircraft industry at the beginning of twentieth century.

Rapid introduction of machines in wide range of application fields created an enormously increasing demand on automating them. Utilization of the electric, pneumatic, and hydraulic power provided the necessary means. The machines became to be named as systems which are assemblies of numerous elements with the aim to fulfill dedicated function or provide capabilities. The term "technical system" emerged as the recognition of machines as systems. In production, this leap is named as the second industrial revolution and symbolized by mass production. This effected also the production techniques of aircraft. The period between the world wars is named as the golden age of aviation [7]. This was the time where the progress from slow production of wood and fabric aircraft to streamlined production of metal aircraft occurred. Flying stepped toward means of transportation from experimental activity. Aircraft such as Douglas DC-3 marked a turning point air transport in the 1930s and 1940s. It was the time of expansion for commercial airlines companies. Besides revolutionary advances in aerodynamics, the innovation in aeronautical technology rendered various components and aspects of aircraft. Aligned with the advances in technical systems in general, the automation requirements hit the aircraft. The first automatic flight control systems were developed during this period. MacRuer and Graham [8] report that by the late 40s the technology level for all electric control from sensors to servos was already reached. The advances in hydraulics on the other

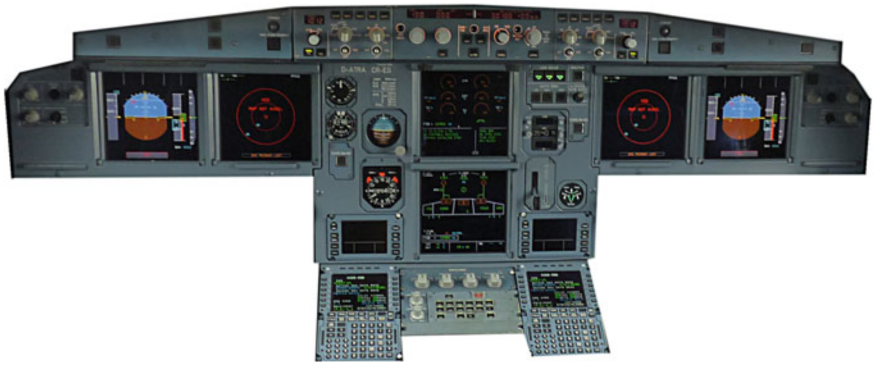


Fig. 1.1 Airbus A320 glass cockpit

hand lead to successful implementations such as in De Havilland Comet 1 [9]. It was the first commercial airliner powered by jet engines.

The next revolutionary step in the history of technical systems came with the rise of computers. The introduction of computers or computing elements in technical systems for enabling further automation characterized the third revolution. Computer Numerical Control (CNC) machine tools, Supervisory Control and Data Acquisition (SCADA), and Computer Integrated Manufacturing (CIM) became the fields of study in production domain.

In aeronautics, it was the rise of avionics, or in other words aviation electronics. Computers became the core elements of avionic suites where they are utilized for functions such as flight control and monitoring, navigation or terrain avoidance. Fly-by-wire flight control systems were introduced to convert control inputs from the pilots to actuator commands to move the control surfaces via electronic means. Computers are used not only for augmenting the pilot inputs for stabilization but also for further automation tasks such as autopilots. Concorde can be referred as the first civil aircraft with fly-by-wire system. In the late 80s, with the Airbus A320, the disruptive capabilities such as high-level control laws in normal operation are started to be provided by fly-by-wire systems [10]. The glass cockpits (Fig. 1.1) can also be introduced as part of the computer revolution in aeronautics. Already in 70s, the increasing level of complexity in aircraft resulted in hundreds of cockpit displays which are meant to aid pilots for more efficient flight, and provide them warnings and cautions. When computers provided the information processing capabilities required to fuse the data and generate condensed information for the displays, glass cockpits were introduced as simplified means of flight data representation using multifunction displays driven by onboard computers [11]. Further advances of the era include Flight Management Systems (FMS) that provide enhanced in-flight automation and pilot cockpit interactions [12] and warning systems such as Terrain Awareness and Warning Systems (TAWS) or Traffic Collision Avoidance Systems (TCAS) [13]. Among all, fly-by-wire systems have kept a special position regarding their high reliability requirements and redundancy approaches.

1.3 On the Cusp of the Fourth Revolution

Today, we are discussing the fourth revolution in the history of technical systems. It is characterized by the words "smart" and "connected". While all previous efforts were intending to automate individual technical systems, today the focus is on the integration of all technical systems within a value chain into digital ecosystems. The driving force of the change can be found in the disruptive innovations in Information and Communication Technologies (ICT).

In his book *The Innovators Dilemma*, Christensen defines two types of technological change [14]. The first one sustains the rate of improvement in product performance. It can be incremental or radical. The second one disrupts or redefines the performance trajectories by creating a new product that provides performance in other dimensions than the ones in main stream, and eventually lead to a new market.

The production domain named the fourth revolution as Industry 4.0. The final report of the Industrie 4.0 Working Group sponsored by the Federal Ministry of Education and Research in Germany lists the revolutions in ICT that brought radical transformations [15]. Smart devices, miniaturization, and smart networks (cloud computing) are introduced as the enablers of ubiquitous computing. It is argued that cloud computing and services permit applying smart algorithms on large quantities of diverse data (big data) collected from smart devices. Powerful microcomputers, wirelessly networked among each others, and the Internet are presented as facilitator for convergence of physical and cyberspace and Cyber-Physical Systems (CPS). The new Internet Protocol IPv6 is named as the enabler of creating Internet of Things (IoT) and services that is composed of network resources, information, objects, and people. The fourth revolution in production is explained as introducing the IoT and services to create networks of smart machines and facilities, namely cyber-physical productions systems that feature end-to-end ICT-based integration.

Aligned with Industry 4.0, Siemens introduced the fourth revolution of the infrastructure technology as follows [16]: The first one is described as brick and steel infrastructure, whereas the second was (semi-) automated infrastructures such as electric railways. The third revolution was named as intelligent infrastructure, e.g., fully automated buildings. They argue that the fourth revolution in infrastructure is a kind of fully integrated intelligent infrastructure which is now being discussed as smart cities. While there are various definitions for smart city, the common understanding is embracing pervasive and ubiquitous computing as well as embedding digitally instrumented devices in urban environments to monitor, manage, and regulate city flows and processes [17].

Aeronautics is also on the cusp of the fourth revolution (Fig. 1.2). After realizing far-reaching automation levels on aircraft, aeronautics domain is now looking at the "smart" and "connected" flight. In the global scale, the Next Generation Air Transportation System (NextGen) project of United States and Single European Sky ATM Research (SESAR) have been looking into transforming the radar-based airspace to a connected one which highly utilizes smart automation of connected entities [18–20]. Almost a decade ago, the NextGen Integrated Plan [21] states that the approach

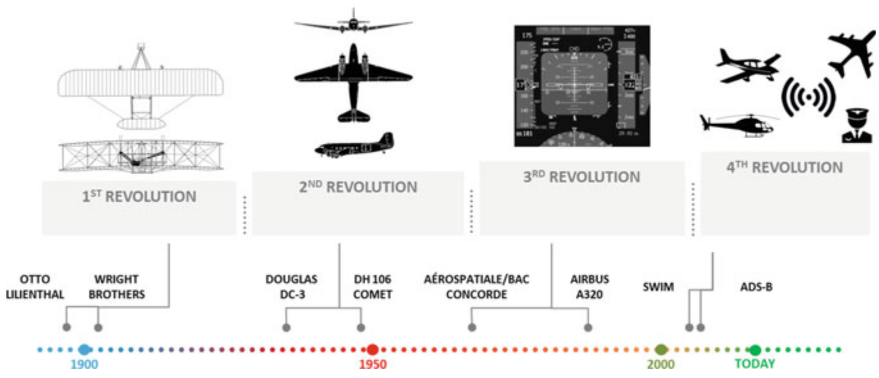


Fig. 1.2 The four eras of flight

where ground-based radars track flyways and pass information from control center to control center on the ground is becoming increasingly inefficient with increase of the density of air traffic. It introduces the upcoming era with its bold recommendation to utilize modern communication techniques, advanced computers, precision localization through Global Positioning System (GPS), and modern computer-based decision assistance programs. In the last decade, we encountered giant leaps in ICT, for example CPS, IoT, service orientation, cloud computing, big data, and wireless networks. They are now pushing the technologies related to flight further than this recommendation and motivate us to name this upcoming fourth revolution in aeronautics as Flight 4.0. It refers to changes in the way that we are doing things in aeronautics from design, construction, to operation of aircraft. Sampigethaya and Poovendran [22] claim that control of aircraft electrical, mechanical, structural, thermal, hydraulic systems and processes, entertainment of passengers, coordination between aircraft and ground stations, between pilots and air traffic depend on advances in ICT. This can be extended to flight training and simulation by extrapolating the evolution of flight simulators presented by Allerton [23].

1.4 ICT of the Fourth Revolution

CPS, IoT, wireless networks, services, cloud computing, and big data are some of the terms that we hear more and more frequently. They are effecting our daily lives; how we work, how we live, how we commute, or even how we sleep. Flight 4.0 is a proposition that says these technologies will be changing how we fly.

CPS is defined as the integration of computation and physical processes [24]. It stresses the challenges that emerge with the embedded systems and networks that monitor and control the physical processes. It includes wide range from medical devices to networked autonomous vehicles and networked building control systems to critical infrastructure control. CPS is claimed as vital for future of the flight [22, 25]. The behavior of CPS is defined by both physical and cyber parts of the system [26]. The physical part of CPS is named as physical plant which may include mechanical parts, chemical processes, or human operators. Cyber part is composed of computational platforms which consists of sensors, actuators and controllers, and network fabric. The physical world of flight includes mechanical aircraft parts, pilot, natural airspace, terrain, and all other man-made systems associated with aircraft whereas cyber world encompasses avionic systems, sensors, and actuators as well as the networking elements such as data buses. Aircraft is an extremely complex CPS with many component interactions to control flight and provide rich functionality. A single flight of an aircraft from a departure to an arrival gate, for example, involves the utilization of actuators to move the control surfaces of the aircraft, cyber domain interactions with ground, air, and space infrastructure, airborne sensing of natural processes such as weather, wind, and wildlife, and human-in-the-loop interactions such as between flight and ground crew [27].

Sampigethaya and Poovendran [22] provide a comprehensive categorization of CPS in aeronautics. We would like to adapt this categorization in the following paragraphs. They start with growth of cyber layer in aircraft where they stressed the paradigm shift from federated and distributed onboard systems architectures to Integrated Modular Avionics (IMA) which employs higher throughput multi-core, multiprocessor systems. Chapter 2 (*Advances in Avionic Platforms: Multi-core Systems*) from Bapp and Becker will be addressing this topic. The enhanced use of wireless networking and data links enabled the integration of onboard, off-board, space and ground elements. High, very high, and ultrahigh frequency radio links, satellite links, and commercial wireless protocols, such as IEEE 802.11, are mentioned in this scope. The topic is further elaborated in Chap. 3 (*Emerging Trends in Avionics Networking*) by Reinhardt and Aglargoz.

Lee and Seshia introduce quadcopters as motivating examples for employing new CPS based approaches [26] which employ systems modeling methodologies, formal verification, simulation techniques, certification methods, and software engineering processes to tackle the emerging challenges in systems design and development [24]. Jafer and her colleagues broaden the discussion about software engineering aspect in Chap. 7 (*Advances in Software Engineering and Aeronautics*). Execution time verification was introduced as one of the key challenges of CPS design and development by Lee [24]. Wilhelm, Reineke, and Wegener in Chap. 9 (*Keeping up with Real Time*) address the timing issues in future avionics systems.

Cyber-physical integration in flight deck is another category. It stresses that the data integration of airspace will lead to jointly performed flight deck operations such as flight control and management or collision avoidance. On one hand, it will enable further automation and autonomy that optimizes flight management and control under various physical constraints such as fuel or noise, on the other hand, it will transform the pilot to an operator of complex network of software-based systems. Further the inclusion of unmanned systems brings us to another level of complexity. In Chap. 6 (*Ontologies in Aeronautics*), Insaurralde and Blasch present the use of ontologies in aeronautics for data integration and supporting decision-making.

Automatic Dependent Surveillance Broadcast (ADS-B) is defined as function on an aircraft or a surface vehicle that periodically broadcasts its state vector (horizontal and vertical position and velocity) along with some other information such as heading or capability codes [28]. It is a part of the aforementioned NextGen and SESAR projects. ADS-B consists of ADS-B Out which refers to aircraft broadcasting and ADS-B In which refers to aircraft receiving another aircraft's ADS-B Out information as well as the ADS-B In services provided by ground systems such as Traffic Information Service Broadcast (TIS-B) and Flight Information Service Broadcast (FIS-B) [29]. While TIS-B includes traffic information within the vicinity, FIS - B includes various weather and flight information such as Aviation Routine Weather Reports (METARs) or Notice to Airmen (NOTAM). Sampigethaya and Poovendran introduce an ADS-B based cyber-physical integration. In ADS-B based air-to-ground networks, effective sharing of weather and traffic information provides means for Trajectory-Based Operations (TBO) [30] which means managing and optimizing individual aircraft precisely in three spatial dimensions and time. With ADS-B based air-to-air networks, it is argued that the ground-independent traffic control tasks such as inter-aircraft spacing may be accomplished by onboard means, which may eventually lead to formation flight [31] type visionary concepts.

Following the ideas from computer augmented environments [32], ubiquitous computing became popular starting from late 90s with a vision having transparent availability of computing resources throughout the physical environment [33]. Ubiquitous sensing supported this concept with providing a transparent sensing abilities with network of sensors [34]. Radio-Frequency Identification (RFID) enabled the smart identification, monitoring, and tracking of objects [35]. Advances in wireless networking provided means for having transparent communication and information sharing [36, 37]. Integrating the devices with the ability to measure, infer, and understand environmental indicators in communicating-actuating networks is now creating the Internet of Things, IoT [38]. Sensors and actuators as well as information and communication system are invisibly embedded in the environment around us. Cloud computing is envisioned as the virtual infrastructure for monitoring devices and storage, analytics, visualization, and presentation with end-to-end services [39]. The National Institute of Standard and Technologies (NIST) defines cloud computing as a model that enables convenient, ubiquitous, on-demand access to share computing resources which include not only networks, servers, and storage but also applications and services [40]. The software architecture of IoT follows the Service-Oriented Architecture (SOA) [41]. SOA promotes integration through a set of services which are programming language and platform independent software components with well-defined interfaces [42]. Voros and his colleagues are elaborating IoT and SOA further in Chap. 4 (*IoT and Service-Oriented Architecture for Flight 4.0*).

Gartner defines big data as high volume, high velocity, and/or high variety information assets that require new processing approaches for supporting discovery, optimization and decision making [43]. IoT promotes the sharp growth of data with sensors all over the world collecting and transmitting data to be stored and processed [44]. It is a heterogeneous large-scale data with strong time and space correlation. Analyzing this data to extract useful information using complex procedures is named as analytics [45]. Analysis procedures include clustering, filtering, regression, and coloration. Hartmann and his colleagues will be presenting a discussion about this topic in Chap. 5 (*Big Data and Data Analytics in Aviation*).

Wind River, one of the market leaders in embedded software, is arguing that IoT concepts have already been applied in aeronautics domain through programs such as NextGen and SESAR [46]. The air space is operated as an open architecture IoT system. The connectivity and information flow through the airspace is enabled. Further cloud-based service infrastructure System Wide Information Management (SWIM) is facilitating the data exchange about all aspects of aircraft operation from flight paths to weather information between producer and consumer systems [47]. IoT concepts have also started to get utilized in the aeronautics domain for individual devices. The best examples are sensor packages that monitor engines [48]. Li et al. are proposing an aircraft as a big data platform [49]. They provide a number of application cases that utilize aircraft data from anti-icing to health monitoring. Tyagi and Nanda also report a bunch of projects that utilize big data and analytics, particularly in air traffic domain [50].

The initial adaptations and implementations provide an insight about the upcoming era where we wait for the wide spread of IoT, wireless networking, cloud computing, SoA, and big data concepts in aeronautics. The fourth revolution promotes smart automation of connected devices and pushing technology from automatic system toward autonomous and intelligent systems. Unmanned Aircraft Systems (UAS) are early adapters of high levels of autonomy and intelligence. In aeronautics, autonomy topic always comes with safety concerns. Torens and his colleagues will be addressing both topics together in Chap. 8 (*Toward Autonomy and Safety for Unmanned Aircraft Systems*).

1.5 A Gentle Introduction of Aeronautical Informatics

Two decades ago, one of the National Aeronautics and Space Administration (NASA) reports, namely *Aeronautics Technology Possibilities for 2000*, defines the fields of study in aeronautics as aerodynamics, propulsion, structures, materials, guidance, navigation and control, computer and information technology, human factors and systems integration [51]. It addresses the key elements of the aeronautics technology landscape of the last 20 years.

Traditional pillars of aeronautics are flight mechanics, aerodynamics, and structures. The organization of the degree program in Massachusetts Institute of Technology for aerospace engineering [52] provides us a valuable insight about today's pillars of the domain. It is organized around three areas: aerospace systems engineering, aerospace vehicles engineering, and aerospace information engineering. Aerospace information engineering focuses on real-time, safety-critical aerospace systems. Its subareas include autonomy, software, communications, networks, controls, and human–machine and human–software interaction. Aerospace systems engineering focuses on the creation, implementation, and operation of complex aerospace systems. Its subareas include system architecture and engineering, simulation and modeling, safety and risk management, policy, economics, and organizational behavior. And finally, aerospace vehicles engineering takes the engineering of air and space vehicles with all their subsystems into its central focus. Fluid and solid mechanics, thermodynamics, acoustics, combustion, controls, computation, design, and simulation are its subareas.

Aforementioned classification from traditional to contemporary designates the change in the technology landscape of aeronautics. ICT which is not there traditionally becomes a part of the classification in NASA report 20 years ago. Today, it is one of the core areas according to MIT when looking at how they structured their aerospace engineering program. Atkins says that aerospace engineering curricula are at crossroads [53]. In Chap. 10 (*Aerospace Engineering Curricular Expansion in Information Systems*), Atkins stresses the recognition about the role of ICT in today's and tomorrow's aerospace education, and further emphasizes the lack of structured means to address basic ICT skills in aerospace curricula.

The term informatics comes from German word "Informatik" which is first introduced by Karl Steinbuch in 1957 [54]. While the theoretical fields of informatics such as formal methods, complexity theory, or computability create the scientific foundations of the discipline, the applied fields such as medical informatics or geoinformatics have been established as the interdisciplinary fields that apply scientific foundations of informatics in particular fields of application. Aeronautical informatics can be defined as the application of informatics in the aeronautics domain. It is the intersection of informatics and aeronautics. This multidisciplinary field is involved in information processing and engineering of information systems in relation to the science or practice of building or flying aircraft. It is concerned with the use of information systems for developing and flying aircraft. Aircraft systems such as flight control, engine control, and navigation systems and all aspects of simulation for systems development and crew training are in its scope.

With this book, the authors would like to highlight aeronautical informatics, the ever-growing applied field of informatics in order to achieve next generation of the flight with understanding, applying, and enhancing the disruptive advancement of ICT in aeronautics.

References

1. Collins English Dictionary - Complete & Unabridged 10th Edition (2017), www.dictionary.com/browse/aeronautics
2. The American Heritage Science Dictionary (2017), www.dictionary.com/browse/aeronautics
3. R. Blockley, W. Shyy, *Encyclopedia of Aerospace Engineering* (2010)
4. V. Hubka, W.E. Eder, *Theory of Technical Systems: A Total Concept Theory for Engineering Design* (Springer, Berlin, 2012)
5. B. Lukasch, O. Lilienthal, *Der Vogelflug als Grundlage der Fliegekunst* (Springer, Berlin, 2014)
6. P.L. Jakab, *Visions of a Flying Machine: The Wright Brothers and the Process of Invention*(Smithsonian Institution, 2014)

7. R.G. Grant, Flight-100 years of aviation. Aircr. Eng. Aerosp. Technol. **75**(2) (2003)
8. D. McRuer, D. Graham, Flight control century: triumphs of the systems approach. J. Guid. Control Dyn. **27**(2), 161–173 (2004)
9. A. Garg, R.I. Linda, T. Chowdhury, Evolution of aircraft flight control system and fly-by-light flight control system. Int. J. Emerg. Technol. Adv. Eng. **3**(12), 60–64 (2013)
10. P. Traverse, I. Lacaze, J. Souyris, Airbus fly-by-wire: a process toward total dependability, in 25th Intenational Congress of the Aeronautical Sciences (2006)
11. E.L. Wiener, Human factors of advanced technology (glass cockpit) transport aircraft (1989)
12. N.B. Sarter, D.D. Woods, Pilot interaction with cockpit automation: operational experiences with the flight management system. Int. J. of Aviat. Psychol. **2**(4), 303–321 (1992)
13. C.R. Spitzer, *Avionics: Elements, Software and Functions* (CRC Press, 2016)
14. C.M. Christensen, *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail* (Harvard Business Review Press, 2013)
15. H. Kagermann, J. Helbig, A. Hellinger, W. Wahlster, Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group (Forschungsunion, 2013)
16. Siemens, Our future depends on intelligent infrastructures (2014), www.siemens.com/digitalization/public/pdf/siemensintelligent-infrastructure.pdf (visited on 06/25/2017)
17. R. Kitchin, The real-time city? Big data and smart urbanism. GeoJournal **79**(1), 1–14 (2014)
18. T. Prevot, NextGen technologies for mid-term and far-term air traffic control operations, in IEEE/AIAA 28th Digital Avionics Systems Conference, 2009. DASC'09 (IEEE, 2009), 2-A
19. P. Ky, B. Miaillier, SESAR: towards the new generation of air traffic management systems in Europe. J. Air Traffic Control **48**(1) (2006)
20. A. Sipe, J. Moore, Air traffic functions in the NextGen and SESAR airspace, in 2009 IEEE/AIAA 28th Digital Avionics Systems Conference (2009). https://doi.org/10.1109/DASC.2009.5347554
21. FAA JPDO, Integrated Work Plan for the Next Generation Air Transportation System (2008)
22. K. Sampigethaya, R. Poovendran, Aviation cyber-physical systems: foundations for future aircraft and air transport. Proc. IEEE **101**(8), 1834–1855 (2013)
23. D. Allerton, *Principles of Flight Simulation* (Wiley, 2009)
24. E.A. Lee, Cyber physical systems: design challenges, in 2008 11th IEEE International Symposium on Object Oriented real-time Distributed Computing (ISORC) (IEEE, 2008), pp. 363–369
25. K. Sampigethaya, R. Poovendran, Cyber-physical integration in future aviation information systems, in 2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC) (IEEE, 2012), pp. 7C2–1
26. E.A. Lee, S.A. Seshia, *Introduction to Embedded Systems: A Cyberphysical Systems Approach* (MIT Press, 2016)
27. S. Jafer, U. Durak, Tackling the complexity of simulation scenario development in aviation, in Proceedings of the Symposium on Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems. MSCIAAS '17 (Society for Computer Simulation International, Virginia Beach, Virginia, 2017), pp. 4:1–4:10. ISBN: 978-1-5108-4030-0, http://dl.acm.org/citation.cfm?id=3108414.3108418
28. R.T.C.A. Minimum Aviation System Performance Standards for Automatic Dependent Surveillance Broadcast (ADSB) (RTCA, Incorporated, 2002)
29. FAA, *Automatic Dependent Surveillance: Broadcast (ADS- B) Out Performance Requirements to Support Air Traffic Control (ATC) Service* Final Rule, Standard (2010)
30. S. Ramasamy, R. Sabatini, A. Gardi, T. Kistan, Next generation flight management system for real-time trajectory based operations. Appl. Mech. Mater. **629**, 344–349 (2014)
31. F. Giulietti, L. Pollini, M. Innocenti, Autonomous formation flight. IEEE Control Syst. **20**(6), 34–44 (2000)
32. P. Wellner, W. Mackay, R. Gold, Computer-augmented environments: back to the real world. Commun. ACM **36**(7), 24–27 (1993)
33. M. Weiser, Hot topics-ubiquitous computing. Computer **26**(10), 71–72 (1993)

34. I.A. Essa, Ubiquitous sensing for smart and aware environments. IEEE Pers. Commun. **7**(5), 47–49 (2000)
35. K. Domdouzis, B. Kumar, C. Anumba, Radio-frequency identification (RFID) applications: a brief introduction. Adv. Eng. Inf. **21**(4), 350–355 (2007)
36. F.L. Lewis et al., Smart environments: technologies, protocols, and applications, *Wireless Sensor Networks* (2004), pp. 11–46
37. V. Coskun, B. Ozdenizci, K. Ok, A survey on near field communication (NFC) technology. Wirel. Pers. Commun. **71**(3), 2259–2294 (2013)
38. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
39. A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of cloud computing and internet of things: a survey. Future Gener. Comput. Syst. **56**, 684–700 (2016)
40. P. Mell, T. Grance et al., The NIST definition of cloud computing (2011)
41. L. Atzori, A. Iera, G. Morabito, The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
42. T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design* (Pearson Education India, 2005)
43. M.A. Beyer, D. Laney, The importance of 'big data': a definition (Gartner, Stamford, CT, 2012), pp. 2014–2018
44. M. Chen, S. Mao, Y. Liu, Big data: a survey. Mob. Netw. Appl. **19**(2), 171 (2014)
45. A. Cuzzocrea, I.-Y. Song, K. C. Davis, Analytics over large-scale multidimensional data: the big data revolution! in Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP (ACM, 2011), pp. 101–104
46. Wind River, Internet of Things in Commercial Aviation (2015), http://events.windriver.com/wrcd01/wrcm/2016/08/WP-IoTthe-internet-of-things-for-commercial-aviation.pdf. Accessed 25 June 2017
47. J.S. Meserole, J.W. Moore, What is system wide information management (SWIM)? IEEE Aerosp. Electron. Syst. Mag. **22**(5), 13–19 (2007)
48. B. Marr, How Big Data Drives Success At Rolls-Royce (2015), www.forbes.com/sites/bernardmarr/2015/06/01/how-big-data-drives-success-at-rolls-royce. Accessed 25 June 2017
49. S. Li, Y. Yang, L. Yang, H. Su, G. Zhang, J. Wang, Civil aircraft big data platform, in 2017 IEEE 11th International Conference on Semantic Computing (ICSC) (IEEE, 2017), pp. 328–333
50. A. Tyagi, J. Nanda, ATLAS: big data storage and analytics tool for ATM researchers. AIAA Infotech Aerosp. 0577(2016)
51. N.R.C. Aeronautics Technology Possibilities for 2000: Report of a Workshop (National Academies, 1984)
52. MIT, Aeronautics and Astronautics (2017), https://ocw.mit.edu/courses/aeronautics-and-astronautics/. Accessed 07 July 2017
53. E.M. Atkins, Education in the crosscutting sciences of aerospace and computing. J. Aerosp. Inf. Syst. (2014)
54. K. Steinbuch, Informatik: Automatische Informationsverarbeitung, in SEGNachrichten (Technische Mitteilungen der Standard Elektrik Gruppe)-Firmenzeitschrift 4 (1957), p. 171

Part II
Information and Communication Technologies Supporting Flight 4.0

Chapter 2
Advances in Avionic Platforms: Multi-core Systems

Falco K. Bapp and Jürgen Becker

Abstract Embedded systems play an ever increasing role in almost any field of daily life, including the mobility domain taking massive benefits from using software in their products. The intense use of software leads to a situation, where processing platforms have to be introduced in many different fields of applications. However, well-known platforms will not be able to satisfy the ever increasing requirements on processing performance. Thus, for new functionality, higher performant systems have to be implemented using alternative and emerging architectures. Multi-core technology, being state of the art in standard ICT for a couple of years now, seems to be the most promising way and will also find its way into avionics systems. However, the characteristics of the target platforms—as will be outlined in Sect. 2.2—changed over the years. Coming from more simple and more easy to use single-core processors to distributed multiprocessor systems toward multi-core processors, the development shows huge differences as discussed in Sect. 2.3. Especially the use of multi-core based systems in the mobility domains introduces challenges that are by far more complex than primarily expected. These challenges, resulting from the basic architecture of the processors, are identified and will be presented in Sect. 2.4. Resulting failure modes and their sources are identified in Sect. 2.5. Finally, the trends and conclusions regarding the emerging multi-core technology are discussed in Sect. 2.6.

2.1 Introduction

Aviation electronics (avionics) were mainly based on the so-called "Federated Architecture", in which (sub-)system function, respectively the corresponding software component, was executed on one dedicated computer. This trend was present in the 70s and was followed by a first optimization in the early 90s to reduce the total

F. K. Bapp (✉) · J. Becker
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
e-mail: bapp@kit.edu

J. Becker
e-mail: becker@kit.edu

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_2

amount of computers within an airplane to be better compliant to the space, weight, and power (SWAP) requirements and to achieve a better maintainability [1]. Based on these requirements, the first concepts arose, which tended to centralized computers [2]. This trend has later been standardized as the Integrated Modular Avionics (IMA) architecture in 2005 [3, 4]. With advanced hardware and software technologies, the same level of safety had to be ensured in comparison to the "Federated Architecture", which has been de facto free of interferences. Based on these assumptions, the characteristics of today's avionics computers changed and will change even more, when considering future and upcoming processing architectures.

2.2 Characteristics of Processing Target Platforms

Being more restricted than other domains, e.g., automotive, avionics are embossed by the certification authorities and the respective guidelines [5–8]. Following these, the use of latest hardware architectures is very ambiguous and hence, most avionics systems still rely on single-core computers. A general block schematic of such a single-core architecture is depicted in Fig. 2.1. These processors consist of one central processing core, an interconnect, and several peripherals. This structure directly makes clear that the processing core is the only entity, which is able to access the other components like peripherals. Furthermore, there are just little dependencies which make it possible to derive properties like, e.g., the calculation of worst case execution times (WCET). However, even though these architectures are better controllable and more deterministic, they are working more and more close to their limit regarding their workload. Adding more functionality might not be possible due to various limitations.

When having a deeper look into single-core processors, several aspects become obvious why these architectures are not able to be tuned to higher performance anymore. In the last years, the development of processors followed Moore's Law,

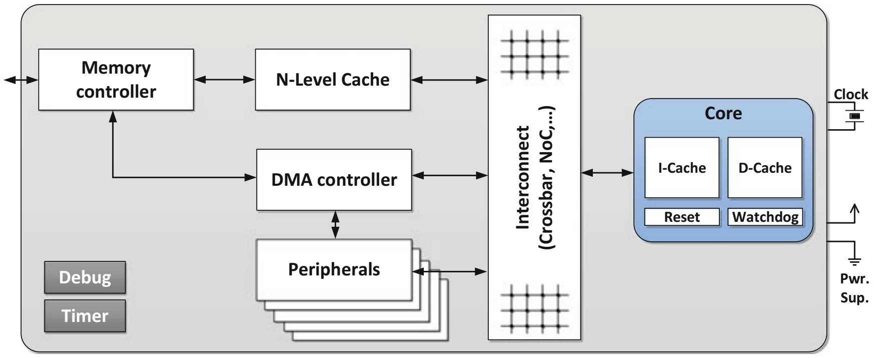


Fig. 2.1 Overview of a generic single-core architecture

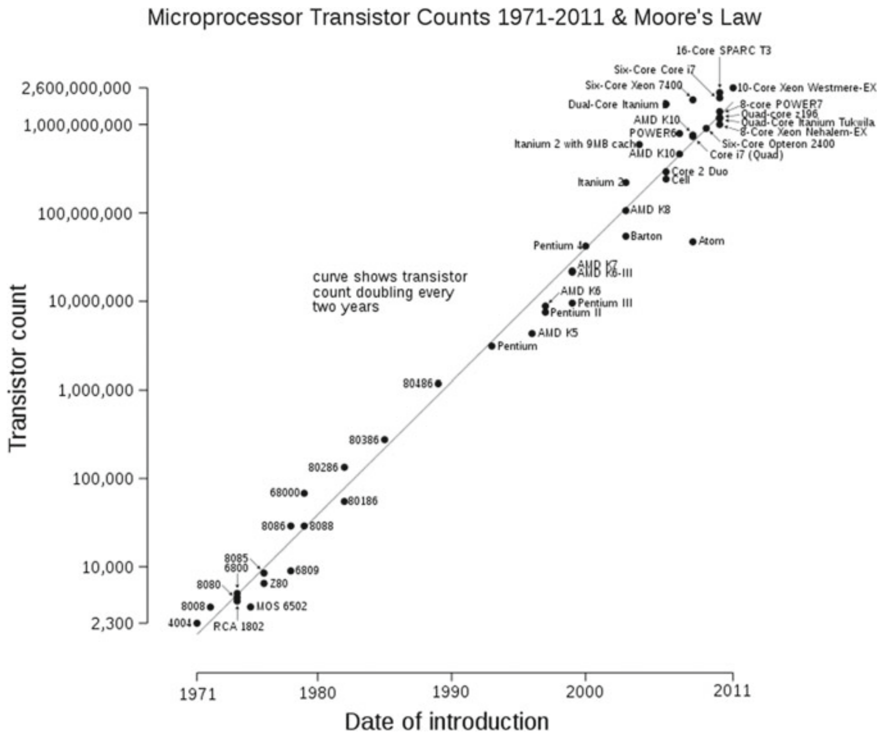


Fig. 2.2 Transistor count in different processor architectures from [9]

which states that the amount of components within a chip will double every year, later every 2 years, depicted in Fig. 2.2. However, the integration density of transistors as well as the realizable clock frequencies reached more and more the physical and thermal limits.

This kind of processors will therefor become rare and expensive. However, they are still the architecture used in avionics systems—as well as in many other domains.

One option to increase the processing performance is the use of increased clock frequencies (e.g., >1 GHz), which directly leads to the need for active heat dissipation. A second option would be the decrease of transistor sizes to have the chance to extend existing functionality. However, both options will lead to an increased risk regarding the sensitivity due to radiations and hence an increased probability for faults due to single event effects (SEE). To overcome these issues, redundancy could be used, but will directly be controversial to the targeted reduction of processing units so save space, weight and power. The usage of redundancy would not be the only option, also specialized controllers for avionics safety systems might be a possibility, which, however, is not feasible due to low amounts of needed devices.

Trends in other domains, e.g., automotive, show that specialized controllers for safety applications become more and more available. These processors include

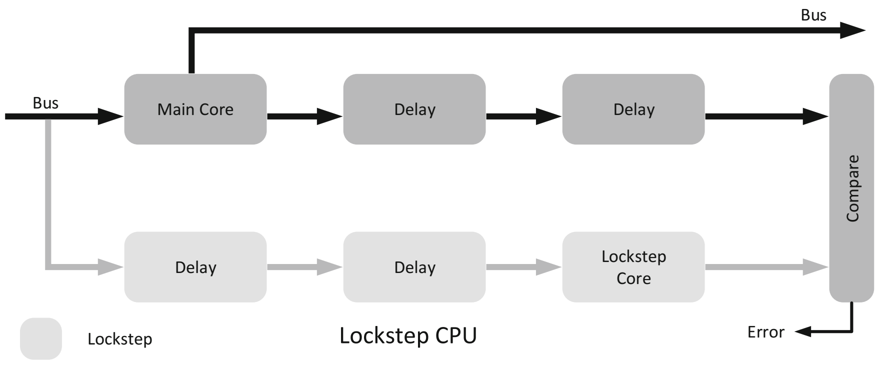


Fig. 2.3 Overview of lockstep CPU architecture according to [10]

several mechanisms that can be used to recognize misbehavior and perform the transition into a safe state. For example, using lockstep architectures (comp. Fig. 2.3) provides means to recognize SEE efficiently. This kind of mechanism uses two tightly coupled processing cores and a comparison logic. However, there is no huge improvement in processing performance since both processing cores are used as one logical core executing the same function and the same software code.

Adding more computing power in this sense means the addition of more computers—centralized or distributed in planes.

2.3 From Distributed Multiprocessors Systems to Multi-core Systems

In recent years, the trend of electronics in different domains changed from a decentralized to a more centralized architecture. This trend was mainly driven by two aspects, first the compliance to the SWAP requirements and second the more and more performant processing architectures made available by semiconductor manufacturers.

2.3.1 Distributed Multiprocessor Systems

For the integration of more processing performance, more and more computers have been integrated in planes, cars, and many other systems. These architectures have been put together using a communication infrastructure in order to exchange data. Each of the computers was responsible for a certain application in a certain place. However, due to increasing functionality, the communication increased as well to

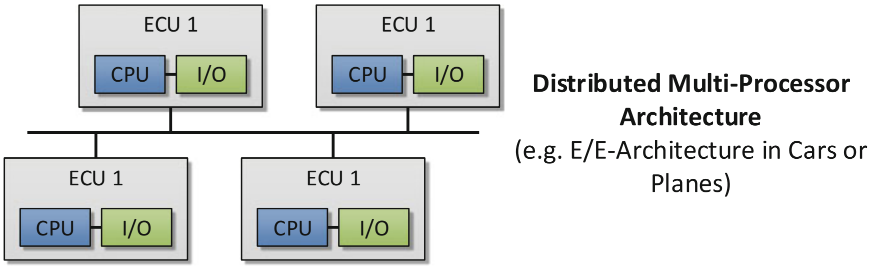

ECU 1
CPU
I/O
ECU 1
CPU
I/O
ECU 1
CPU
I/O
ECU 1
CPU
I/O
Distributed Multi-Processor Architecture
(e.g. E/E-Architecture in Cars or Planes)

Fig. 2.4 Overview of of a decentralized multiprocessor architecture

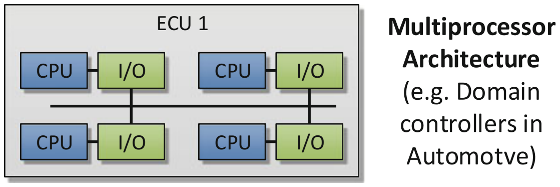

ECU 1
CPU
I/O
CPU
I/O
CPU
I/O
CPU
I/O
Multiprocessor Architecture
(e.g. Domain controllers in Automotve)

Fig. 2.5 Overview of a centralized multiprocessor architecture

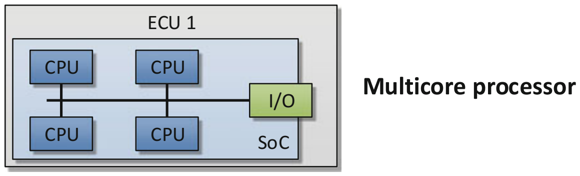


Fig. 2.6 Overview of a multi-core processor

exchange more and more information and data. This distributed multiprocessor architecture is depicted in Fig. 2.4. One very important characteristic of such an architecture is the fact that each of the controllers have, e.g., their own memory and I/O interfaces but a shared interconnection with comparably high latencies.

This kind of architecture is still valid and in use, however, the target is to decrease the number of computers and interconnections as well as communication overhead. Based on this idea, the next step was performed to more integrated architectures, like, e.g., domain controllers in the automotive domain, that can host several processors within one computer and hence reduce the "far" communication overhead and latencies. This results in an architecture as depicted in Fig. 2.5.

Having new possibilities due to latest technology advances, a further step of integration is feasible. A transition from centralized multi processor systems to multi-core systems helps reducing further overhead and controllers (comp. Fig. 2.6). However, also this transition has its drawbacks, in terms of challenges that arise as described in Sect. 2.4, as well as opportunities, which are presented in detail in Sect. 2.3.2.

2.3.2 *Emerging Technology: Multi-core Processors*

Having a more detailed look into latest processing architectures, namely multi-core controllers, will make the difference to their predecessors, namely single cores, obvious. In comparison to a single-core processor, these architectures have integrated several independent processing cores.

However, this is not the only difference. The main differences can be characterized as follows:

- Increased number of processing cores
- Shared memory infrastructure, starting from typically Level 2 Cache
- Shared on-chip interconnect infrastructure
- Shared peripherals, coprocessors, accelerators, and services
- Common voltage supply and clock for several cores

In such architecture, the processing cores are tightly connected and typically share the memory infrastructure. In various available embedded multi-core processors, only one single memory controller is available. Even the cache architecture is typically shared among different processors, starting typically from level 2 cache. To mention two architectures as possible representatives, the P4080 [11] architecture and the i.MX6 Quad [12] platform from NXP are chosen, even though, there are many more architectures, e.g., from Infineon, TI, and others.

When abstracting these architectures to a generic multi-core architecture, several components can be identified that they have in common. Such a generic multi-core architecture is depicted in Fig. 2.7.

Very beneficial of such architectures is the close interconnection and hence very low latencies for inter-core communications and data exchange. Furthermore, the

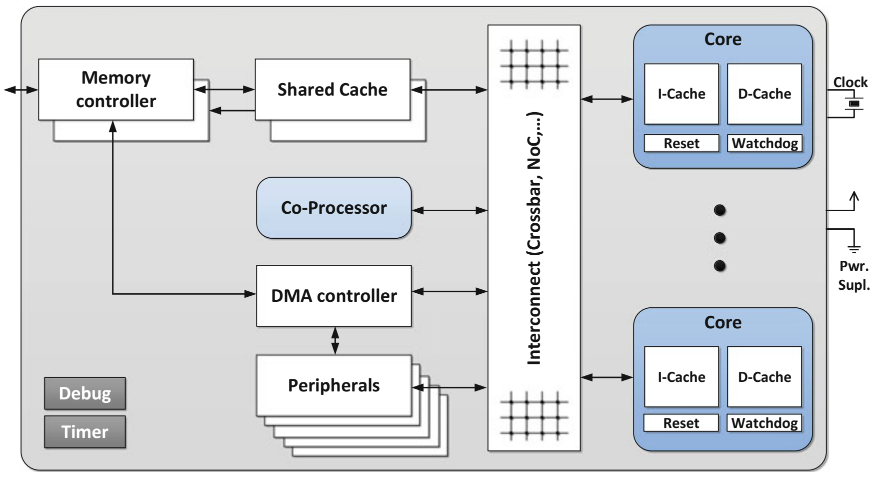


Fig. 2.7 Generic multi-core architecture

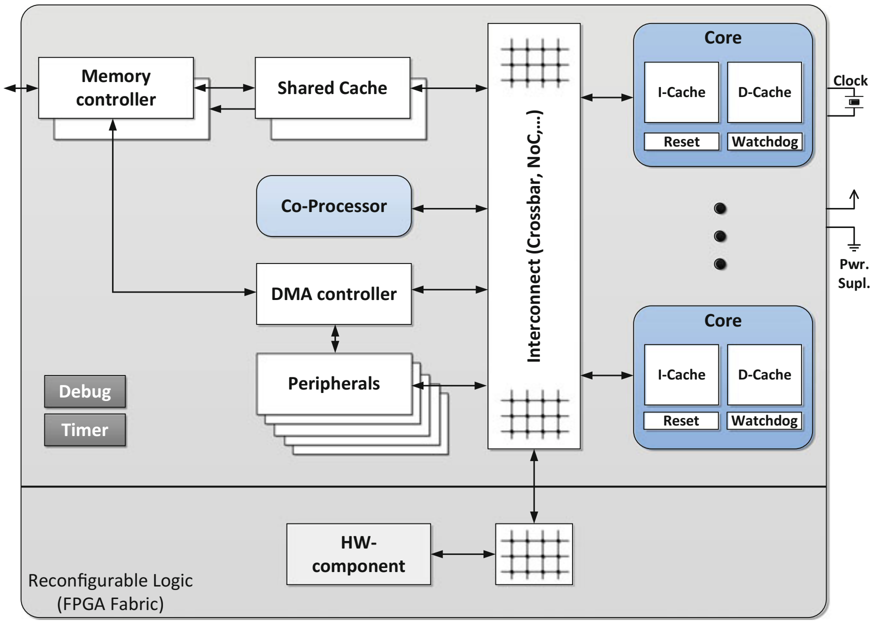


Fig. 2.8 Generic multi-core architecture with integrated reconfigurable logic

access to shared memory provides a good possibility to share data, but also introduces new challenges that will be discussed in Sect. 2.4.

Even though, multi-core processors can provide strongly increased processing power in comparison to single cores, several applications exist, where a hardware acceleration can be beneficial. These hardware accelerations can be realized using reconfigurable hardware such as FPGAs. Latest trends in processing architectures are the integration of a multi-core processor as well as a reconfigurable fabric. However, the reconfigurable logic can not only be used to realize accelerators but also means to increase safety. A generic overview of such an architecture including the reconfigurable logic is depicted in Fig. 2.8.

Possible hardware extensions using reconfigurable logic could be means for online monitoring [13, 14] or hardware virtualization [15]. Such reconfigurable logic can additionally be used in a very flexible and dynamic manner also at runtime of the system. However, this would be a major step for future avionics systems, since dynamic features are not considered in today's certification processes.

Following this description of the characteristics of multi-core processors, the challenges are derived in the next section.

2.4 Challenges Resulting from the Multi-core Architecture

Based on the integration of more and more functions as well as components within a Multi-Processor-System-on-Chip (MPSoC), the complexity for the development is ever increasing. Having a look at the documentation, it is obvious that this can be used as a proof for the increasing complexity. Even though, manuals have several thousand pages—and multiple manuals are needed for one architecture—not all components are clearly described and mentioned in them. Several components are used just for debugging and development purposes by the manufacturer and are hence not documented and usable for a productive use case. Also an influence of these components to the productive system cannot be precluded in general.

However, the main challenges for the use of multi-core systems in safety-critical applications can be summarized as follows:

- **Segregation in time and Space**:
  New methods are needed, which can realize a safe sharing of resources while respecting timing limitations.
- **Synchronization and distribution of Application Software**:
  Software components/functions realizing an application have to be distributed efficiently while respecting a cost function consisting of synchronization and communication efforts.
- **Efficient distribution of platform Software**:
  Analogously, the distribution of platform software has to be performed like, e.g., operating systems, in order to reduce communication and synchronization efforts.
- **Analysis of multi-core architectures**:
  In order to derive guarantees and worst case approximations like, e.g., worst case execution times (WCET), the hardware and software architecture need to be analyzed.
- **Managing the complexity**:
  Managing the configuration space of a multi-core architecture is necessary to achieve a safe and reliable configuration of the platform.

For new systems with a safety-related focus, these challenges need to be addressed. Especially predictability and determinism are of utmost importance in order to guarantee the systems behavior. However, for the described challenges there are not always solutions at design time, what makes it necessary to introduce also some mechanisms like online monitoring or advanced tracing and so forth.

Basically these challenges can result in various failure modes from an application point of view.

2.5 Possible Failures from an Application Point of View

Typically faults resulting from the multi-core challenges can result in a failure of the application. When having a look from an application perspective, three main failure modes can be identified—high execution latency, erroneous calculations/data/accesses, missing execution. These failure modes can result from different sources as depicted in Fig. 2.9.

These failure modes can also be present in single-core systems, but the resulting effects are by far extensive in a multi-core context. Additionally, the sources for the failure modes are manifold.

For example the calculation or estimation of worst case execution times (WCET) becomes more and more challenging, when having several accesses to resources in parallel. This can result from shared caches where data was displaced by another core and hence need to be fetched from main memory which was not intended to be. Based on the usage of highly efficient cache displacement strategies, which are typically not deterministic, timing can be delayed and hence deadline missed. A similar behavior can be observed when having highly increased communication traffic on a shared interconnect, where accesses of any type can be delayed.

Erroneous data on the other hand can result from usage of outdated information resulting from dependencies of one core to another, so called Race-Conditions (time

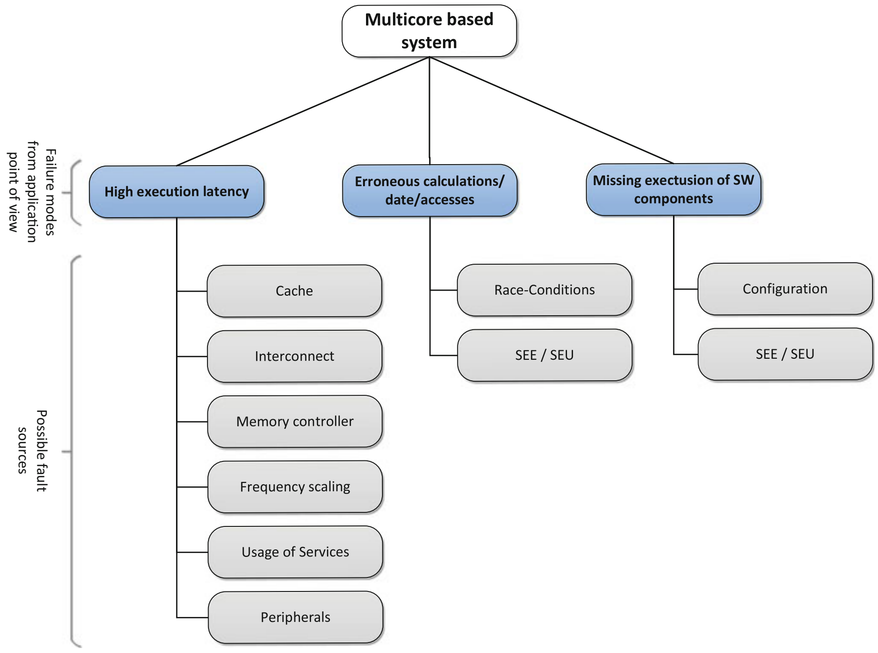


Fig. 2.9 Possible failure modes in multi-core based systems from an application point of view

of use vs. time of check). In such a case, new data is already in calculation but not available in memory while another core accesses the available data in the memory. Further reasons for failure modes of the application can result from single event effects (SEE)/single event upsets (SEU). Especially in a multi-core architecture, the integration density of transistors is higher than in a single-core and hence the probability for radiation effect becomes higher. Furthermore, more transistors can be influenced and hence a broader range of effects is possible. Additionally, missing execution of software components can result from a malicious configuration of the interrupt signals as one example. In such a case, a wrong core would react on an interrupt intended for another core. Also, these malicious interrupt configurations can result from SEE/SEU in the configuration registers.

To sum this up, there are many possible fault sources that can result in a failure mode of the application running on a multi-core based system. Even though, the failure modes can also arise in a single-core architecture, the effects and influences are much bigger in a multi-core scenario.

2.6 Conclusions

Having a look into avionics systems clearly shows that embedded systems still rely on single-core computers. However, this has to change in near future in order to fulfill all requirements regarding processing performance. On the other side, more and more highly performant embedded architectures become available, which could fulfill these needs. However, these architectures are typically not intended for the use in avionics and hence need some further mechanisms to be integrated.

This is exactly what is more and more also on the roadmap of the well-known semiconductor manufacturers, even though there is no avionics specialized controller. There are many possibilities and chances to integrate architectures from other domains, e.g., from the automotive industry. These synergies to other domains can be discussed as in [16].

As emerging technology multi-core processor have to be more and more in the focus of avionics engineers. Further detailed investigations have to be performed of course to gain more and more knowledge and arguments for a successful implementation. Online monitoring and detailed analysis become even more important than today in combination with low overhead and taking the opportunities of multi-core architectures.

References

1. R. John, Partitioning in avionics architectures: requirements, mechanisms, and assurance, in *NASA Langley Technical Report Server* (1999)

2. H. Butz, Open integrated modular avionic (IMA): state of the art and future development road map at airbus Deutschland. Signal **10**, 1000 (2010)
3. R.L. Eveleens, Integrated modular avionics development guidance and certification considerations. Mission Syst. Eng. **2**, 1120–1132 (2006)
4. J.W. Ramsey, Integrated modular avionics: less is more-fresh approaches to integrated modular avionic architectures will save weight, improve reliability of A380 and B787 systems. Avion. Mag. **31**(2), 24 (2007)
5. RTCA. DO-254 design assurance guidance for airborne electronic hardware (2000)
6. Certification Authorities Software Team (CAST). Position Paper CAST-32A - Multi-core Processors. Rev. 0 (2016)
7. European Aviation Safety Agency (EASA). Certification memorandum - SWCEH-001 development assurance of airborne electronic hardware. Rev. 01 (2011)
8. European Aviation Safety Agency (EASA). Certification memorandum - SWCEH-002 software aspects of certification. Rev. 01 (2012)
9. Wgsimon. Moore's Law, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=15193542
10. Infineon. Product brief: highly integrated and performance optimized, 32-bit microcontrollers for automotive and industrial applications (2014)
11. NXP. P4080 QorIQ integrated multicore communication processor family reference manual (2012)
12. NXP. i.MX 6Dual/6Quad applications processor reference manual (2013)
13. O. Sander, F. Bapp, T. Sandmann, V.V. Duy, S. Bähr, J. Becker, Architectural measures against radiation effects in multicore SoC for safety critical applications, in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)* (IEEE, 2014), pp. 663–666
14. F.K. Bapp, O. Sander, T. Sandmann, V.V. Duy, S. Baehr, J. Becker, Adapting commercial off-the-shelf multicore processors for safety-related automotive systems using online monitoring. Technical report, SAE Technical Paper (2015)
15. F.K. Bapp, O. Sander, T. Sandmann, H. Stoll, J. Becker, Programmable logic as device virtualization layer in heterogeneous multicore architectures, in *International Symposium on Applied Reconfigurable Computing* (Springer, Berlin, 2016), pp. 273–286
16. O. Sander, F.K. Bapp, L. Dieudonne, T. Sandmann, J. Becker, The promised future of multicore processors in avionics systems. CEAS Aeronaut. J. **8**(1), 143–155 (2017). https://doi.org/10.1007/s13272-016-0228-x. ISSN: 1869-5590

Chapter 3
Emerging Trends in Avionics Networking

Andreas Reinhardt and Aysegul Aglargoz

Abstract Embedded sensing systems are widely deployed aboard aircraft to capture flight parameters and cater to their processing, logging, and visualization. However, it is their interconnection to form *avionics networks* that facilitates the provision of a large range of additional functionalities. Most prevalently, the fusion of sensor data collected at different points within aircraft enables the collection of a holistic and comprehensive situational picture. Several key design decisions must be made to set up avionics networks in practice: Besides the identification of suitable hardware platforms, decisions must be made regarding the selection of communication technologies to use, the desired network topologies, and the choice of networking protocols. Across all these dimensions of the parameter space, application-specific requirements must also be adequately catered for, e.g., to meet latency, performance, or reliability constraints. In this chapter, we will discuss requirements to avionics networks as well as highlighting design options to meet them. At last, we present selected promising avenues for future research.

3.1 Introduction

Networked computer systems play a considerable role in almost all current means of transportation. Modern cars feature up to one hundred embedded systems to provide functionality for control and comfort [1]. Even more such devices are present in current-generation aircraft, catering to a broad diversity of services. A plethora of vital functionalities are provided by such *avionics systems*, such as flight control (*fly-by-wire*), navigation support, or the continuous monitoring of mechanical

A. Reinhardt (✉)
Clausthal University of Technology, Clausthal-Zellerfeld, Germany
e-mail: andreas.reinhardt@tu-clausthal.de

A. Aglargoz
German Aerospace Center (DLR), Braunschweig, Germany
e-mail: aysegul.aglargoz@dlr.de

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_3

components, environmental conditions, and the structural health of the aircraft. A commonality of virtually all deployed avionics components is their reliance on networking, i.e., the exchange of data between systems over communication links.

Parameters relevant to the aircraft's current operating conditions constitute the most prevalent and simultaneously the most important type of network traffic. Such data can originate from diverse systems and sensors, both from aboard the aircraft (e.g., engine condition monitoring) as well as external to it (e.g., navigation data received via satellite connections). The processing (i.e., fusion, consolidation, and interpretation) of incoming data enables control systems to infer the aircraft's operational state, exert control over actuators, and thereby ensure safe operating conditions during all flight phases. It is obvious that the data generated by this wide range of sources needs to be forwarded to processing systems or data concentrators in a reliable and timely manner, and that flight control commands impose similarly high dependability requirements on the communication links across which they are transmitted. Safety and dependability are consequently among the key requirements to be fulfilled during the design and implementation of avionics networks. In order to meet these requirements in practice, appropriate choices of the employed communication standards, network topologies, and communication protocols need to be made.

Today, *wired* communication networks are primarily used for the transmission of sensor readings and control commands aboard aircraft. Wired networks are preferably chosen because of the possibility to deterministically guarantee upper bounds on packet loss rates and delays. Thus, the dependable transmission of parameters critical to the safe operation of aircraft is ensured. Furthermore, wired networks can be designed to support the multiple-source multiple-sink traffic patterns prevalent in avionics networking, where collected sensor data need to be forwarded to the relevant systems for consolidation and processing. Besides designing the communication infrastructure, the choice of suitable communication protocols is of equal importance. Several such protocols for wired links in commercial aircraft have been proposed and published (mostly in the form of ARINC standards), and are widely used in practice. Their designs have been specifically adapted to accommodate the traffic patterns prevalent on aircraft, where the number of networked sensing, processing, and actuation devices is usually large. The symbiotic combination of aforementioned aspects in communication network planning enables such networks to provide the required quality of service (*QoS*) level. That is, they guarantee a deterministic bound for the minimally achievable throughput as well as upper bounds to delays and the likeliness of packet losses.

A number of limitations, however, impact the applicability of such, often proprietary, solutions to serve as the sole communication backbones in next-generation avionics. First, requirements to the communication links are strongly influenced by the nature of the data to transfer. Sensors that generate data of constant size at periodic intervals facilitate optimal traffic scheduling and thus allow to meet QoS requirements deterministically. An increasing scheduling complexity ensues, however, once data sources generate multimedia data streams with high-bandwidth requirements or transmit sensor data at irregular intervals and with varying packet sizes. More flexible networking solutions, capable of achieving traffic-dependent real-time characteristics, are necessary to meet these requirements. Second, several currently used communication protocols impose limits on the number of supported terminal systems that may be attached. In other words, their scalability is limited. To establish connectivity between all data sources and sinks, additional processing devices must be added to act as data concentrators. The resulting network topologies often no longer allow to reflect the logical interconnection of systems within the physical network, which in turn complicates their management. As a last important point, the cost of installation-specific wired connections as well as the corresponding weight of both cables and necessary conduits represent an obstacle in the design of fuel-efficient aircraft.

As demonstrated by the enormous success of the World Wide Web, the convergence to a basic set of widely accepted protocols and standards is a keystone for building future-proof networking solutions. In the avionics domain, a current trend toward unifying wired communication networks is the aircraft data network (ADN), specified in ARINC 664 and implemented, e.g., in the Avionics Full-Duplex Switched Ethernet (AFDX). Based on the well-established and widely used IEEE 802.3 (Ethernet) protocol, ADN has been designed to meet the requirements of avionics, i.e., accommodate traffic that has high reliability and real-time requirements. The development cost of switching and routing equipment for ADN is greatly lowered in comparison to the use of proprietary solutions (such as ARINC 429 and ARINC 629) due to its wide compatibility with commercial off-the-shelf Ethernet devices. Despite the higher bandwidth provided in ADN as compared to earlier avionics networking standards, a capacity limitation is still present due to the hardware limitations of IEEE 802.3.

Nonetheless, a constantly rising demand for networking capacities is expected. The reason lies within steady technological progress in the domain of microelectromechanical systems (MEMS) [2]. Being able to manufacture miniaturized and networked sensing devices for the collection of manifold physical parameters enables a more comprehensive situational monitoring in future-generation aircraft. This trend is well in line with the upcoming vision of the Internet of Things [3], in which a perpetually growing number of heterogeneous devices are expected to be interconnected on the same world-spanning network. As a consequence of this growth in the number of networked entities, network capacity limits will be reached eventually. Practical effects of the exhaustion of networking resources range from minor violations of QoS levels to total network failures. It is thus imperative to consider emerging trends in computer networking in order to create a sustainable and scalable infrastructure for the future deployment of sensing and processing devices on aircraft while meeting their QoS requirements at the same time.

One approach to temporarily accommodate the continuously growing number of devices is to establish separate networks for real-time control traffic and data transmissions without strict QoS guarantees. Interconnected at domain gateways or dedicated network bridges, the accomplished separation of concerns increases the possible extent of concurrency. The improvements to the management and control of such networks are, however, generally counterbalanced by the need to deploy separate network infrastructure. Another way to alleviate these effects is the use of *wireless* networking. Even today, the use of wireless connectivity is indispensable for *aviation communication*, i.e., the voice and data communication between aircraft as well as between aircraft and ground stations. By way of an example, each aircraft's automatic dependent surveillance broadcast (ADS-B) system periodically broadcasts position and velocity information to other stations [4]. In contrast, the potential of wireless networking used locally aboard an aircraft has only seen limited scientific coverage, despite strong ongoing research interest in the area of wireless networking.

3.2 The Potentials of Wireless Networking

Wireless communications are ubiquitous and find application in innumerable use cases. Satellites broadcast television channels or positioning information around the clock. Personal handheld devices are equipped with capabilities to wirelessly connect to cellular and/or local area networks (Wireless LANs). In aviation, all long-distance voice and data exchanges are handled via radio communications. Aligned with this trend, wireless channels have also been identified as promising candidates for the communication between embedded systems. The resulting intensive research activities in the areas of wireless sensor networks (WSN) [5] and cyber-physical systems (CPS) [6] have demonstrated the huge potential for the wireless monitoring of remote sites, the design of networked embedded control systems, and the realization of automation solutions that rely on the wireless exchange of data.

The enormous success of wireless networking between sensors, actuators, and processing systems can be primarily attributed to the flexibility of untethered operation. Wireless networks have measurably lower installation requirements than their wired counterparts. Instead of having to supply each connected terminal system with a dedicated wired data connection, required efforts reduce to the installation of bridging devices (i.e., wireless access points or border routers) and their interconnection within a local area network (LAN). Moreover, wireless embedded sensing systems can be deployed in close physical proximity to the phenomenon of interest. In fact, sensors can even be positioned on the fuselage of aircraft, as demonstrated in [7]. While the distributed deployment of devices creates new challenges for their power supply, low-power embedded systems can often be operated on battery power for extended periods of time, or employ techniques like energy harvesting [8] to sustain their operation. Lastly, the limited wireless range implicitly caters to locality-aware connectivity. Devices in physical proximity can communicate with each other directly, but interference between remote systems is extremely unlikely due to the path loss experienced by wireless data transmissions. While this necessitates the use of multi-hop routing protocols to interconnect devices beyond local range, the resulting smaller collision domains can cater to less channel contention and may even counterbalance the energy expenditure for multi-hop message relaying.

Wireless local area networks are already in practical use aboard many aircraft, particularly in the domain of in-flight Internet access and partially also for onboard entertainment. Virtually all such systems are based on the IEEE 802.11 standard and cater to on-demand multimedia (e.g., video and audio) streaming and/or satellite-based Internet access. Due to the non-real-time nature of transported traffic, however, QoS considerations take a subordinate role. It thus remains to be assessed whether IEEE 802.11 is a suitable substrate to exchange data collected from onboard CPS, or if other communication solutions should be given preference in order to meet the corresponding requirements better.

3.3 Wireless Networking in Avionics

Technological advancements in sensor technology and the ongoing miniaturization of embedded microsystems will result in a significant growth of the number of sensing and actuation systems aboard aircraft. Their rollout at large scale is, however, hampered by the limited scalability of wired network topologies, the weight of corresponding cables and conduits, and resulting constraints to deployment locations. Wireless connectivity between embedded devices that form cyber-physical systems is thus indispensable in the long term. Although current wireless LAN standards qualify as alternatives to wired network connections, they introduce new challenges. Most importantly, they often provide limited support for QoS and dependability, i.e., the resilience to disturbances and packet loss. This is in opposition to wireless avionics networks which generally require tight QoS guarantees to be in place due to safety and security concerns. Informed decisions on the choice of the used communication standard and protocol must thus be made, and possibly supplementary ways to increase reliability be added.

Even though wireless channels are inherently prone to external disruptions, several emerging technologies have the potential to mitigate the resulting problems. These disruptive innovations in wireless communications and networking can strongly be expected to pave the way toward the next generation of avionics networks and systems. We summarize selected current trends as follows.

3.3.1 Communication Standards and Spectrum Utilization

Diverse networking techniques and protocols have been proposed and standardized for wireless local and personal area networks. Their throughputs range from less than 1 MBit/s (IEEE 802.15.4) to data rates in excess of 1 GBit/s (IEEE 802.11ad). The range and power consumption of corresponding transceiver chipsets is, however, often correlated with the achievable transmission rate. Particularly as low-power embedded sensing devices may never fully saturate links that provide high data rates or do not require long-distance transmissions, the choice of the communication standard needs to be made carefully.

A key design decision to make is the selection of the frequency band to be used. Most existing wireless networking standards use frequencies in the ISM (industrial, scientific, medical) bands [9]. As a result, concentrations of traffic can be observed in very few select frequency ranges (such as 2.4 GHz, 5 GHz, or 868/915 MHz). While it may appear counterintuitive to rely on these highly utilized frequency bands in order to attain high reliability and low packet loss rates, reasons to do so exist nonetheless. First, ISM bands are globally exempt from licensing; thus, no additional cost is incurred for their usage. Second, due to the worldwide use of ISM frequencies for wireless data traffic, chipsets implementing many communication standards are readily available, thus lowering the development cost. Third, channel contention and the resulting risk of interference in the ISM bands may be high on the ground, but becomes less severe once an aircraft is airborne, particularly when measures for coordinated medium access (cf. Sect. 3.3.2) are adopted.

Performance and range considerations also take a key role in the choice of the wireless frequency band. While wireless communications operated at higher frequencies generally provide larger channel bandwidths and consequently allow for a greater throughput, their wireless range is often reduced due to the attenuation and reflection of signals by obstructions in their path. Higher transmission power settings can partially compensate for this limitation, yet they generally lead to a higher demand for electrical power to operate the wireless transceiver devices. Thus, they violate the low-power operation requirements of embedded sensing systems. Moreover, transmission power settings cannot be increased indefinitely due to regulatory restrictions. Another factor to consider is that transmissions at high power settings can cover larger distances, yet they simultaneously increase the size of the sending station's collision domain, i.e., the radius in which its signals can lead to interference. The advantages of multi-hop communications (see also Sect. 3.3.4), where data is iteratively being forwarded by devices with short communication ranges, are thus often likely to outweigh the use of long-range communication links because of their smaller collision domains.

It needs to be noted that the protocol selection need not result in a single choice for all wireless avionics communications. Rather, multiple protocols with different properties can be selected according to the application requirements and be operated in parallel, either as standalone networks or interconnected through bridging devices. In both cases, it is imperative that the coexistence between different networks devices is ensured. The simultaneous use of the wireless spectrum can be accomplished by dividing it into frequency bands (i.e., using a frequency division multiple access, FDMA) scheme or applying other dimensions of separation, such as time (TDMA), code (CDMA), or space (SDMA). Regardless of the planned approach to ensure coexistence of the resulting networks, the same selection criteria as outlined above apply when operating two or more communication standards in avionics networks.

3.3.2 *Cognitive Radios and Collision-Free Interoperability*

In order to facilitate the concurrent use of the wireless frequency spectrum by multiple services, it has been logically separated into *frequency bands*. The resulting assignments between these frequency ranges and corresponding service types [10] are performed by regulation authorities, such as the Federal Communications Commission (FCC). A key objective of using fixed mapping is the guaranteed availability of the spectrum allocation to a given service type and the consequent absence of interference. This separation is enforced in practice through penalties for unauthorized spectrum use. Regulations for the frequency use *within* defined bands, such as in the ISM ranges, are less strict, however. If limitations are applied at all, they primarily relate to the maximum transmission power and allowed duty cycle (i.e., the fraction of time a transmitter may utilize the wireless channel) of stations [11].

This lack of regulation within the ISM bands often leads to their unbalanced utilization [12] and a resulting network performance below the theoretical bounds. With interfering transmissions representing the main cause of delays and data losses, a better balancing of traffic is required for future-generation networks. On these grounds, the use of *cognitive radios* is a growing research area of high potential for avionics. Its objective is to enhance the use of the wireless radio spectrum through real-time assessments of its utilization. In essence, through constant monitoring of a network's utilization, stations can make informed decisions on their channel choices. Platforms fitted out with multiple radio transceivers can even switch to completely different frequency bands if their preferred band is saturated. Despite their operation without a central coordination, measurable throughput increments have been reported [13]. Besides the trend to use cognitive radios to improve the utilization of channels within the ISM bands, they bear even greater potential when applied to other frequency bands. More specifically, the reuse of licensed frequencies becomes possible [14] as long as the cognitive nature guarantees that transmissions will cease immediately when usage by its assigned spectrum owner is determined. This way, an even larger spectrum becomes available for onboard wireless traffic, on condition that the used transceiver hardware is ready to utilize these bands. Spectrum awareness is inevitable for wireless avionics networking to face the increasing penetration of aircraft cabins with user-operated wireless devices, such as Bluetooth-based headphones or cell phones. Moreover, through the identification of traffic patterns as well as currently available and unavailable channels in real-time, cognitive radios can even detect malicious network access attempts and thus increase the level of safety and security.

Instead of adding spectrum sensing functionality to all deployed wireless stations, the introduction of dedicated *spectrum agents* with complex sensing capabilities is proposed as an alternative approach. Even though originally conceived as a concept for 5th generation wireless systems (5G) [15], the use of spectrum agents in avionics networks is envisioned to carry high potential. Spectrum agents are devices with wide-band sensing capabilities and high computational performance to cope with the fast data processing tasks required to assess the current spectrum usage. Consequently, they can provide spectrum utilization information to the network within a short period of time while keeping the cost, complexity, and power consumption of embedded sensing devices unaffected by spectrum sensing tasks. With the introduction of spectrum agents, complex sensing schemes can be implemented, which provide more reliable information about the transmission medium and can, e.g., be used to optimize the channel allocations.

3.3.3 Software-Defined Networking

Cognitive radios and spectrum agents can be categorized as *reactive* approaches to improve the spectrum utilization. They rely on a passive monitoring of the spectrum and assist in the selection of networking resources when data shall be transmitted. The use of cognitive radios almost always caters to higher throughput as compared to the conventional approach of using fixed channel allocations. However, it needs to be noted that stations can only collect local observations of the spectrum utilization. Thus, network-wide requirements, e.g., QoS guarantees for multi-hop traffic flows, cannot be accommodated through cognitive radios by design. Even when the spectrum is probed proactively, i.e., regardless of whether an actual transmission is pending [16], cognitive radios can only take localized observations into consideration when making decisions on the channel to use.

The centralized coordination of networking resources represents an alternative solution to manage the use of available capacities. When complete information on data flows on network scale is known to a controller, an optimal assignment of routing paths and resource allocations can be centrally computed and equipment in the network be configured correspondingly. This approach is highly advantageous when QoS guarantees, e.g., the accommodation of traffic with real-time requirements, are needed beyond a station's one-hop neighborhood. The concept is known by the name of software-defined networking (SDN) [17, 18] and already widely exploited in corporate networks to date; gradually, it is also finding its way into broader use on the Internet [19]. Its operation is straightforward: Any station that needs to make decisions on the networking level forwards its requirements to an SDN controller node, which in turn computes a suitable network configuration and returns it to all concerned devices. Besides adjusting the channel allocations, decisions may even include the definition of parameters for the wireless communication protocol (e.g., channel setting or transmission power levels). The SDN concept thus caters for a flexible and scalable management of networks, which is vital for large-scale systems.

SDN relies on a separation of concerns: A communication channel, called the *control plane*, is used to let the SDN controller know about traffic flows and retrieve corresponding configuration information. The actual data traffic is carried on a logically (and often also physically) separated connection, the so-called *data plane*. In line with the possibility of equipping wireless devices on aircraft with multiple wireless interfaces (cf. Sect. 3.3.1), such a separation of channels can also be accomplished in avionics. An evaluation to what extent SDNs can be used to ensure QoS guarantees, improve dependability, and maximize network performance has to be considered an open research question, however, despite its promising prospect.

3.3.4 Dependable and Secure Communications

The aspects of safety and security are of paramount importance in avionics systems. Accomplishing these properties in software imposes strict requirements on development processes to ensure compliance to the DO-178B guideline (Software Considerations in Airborne Systems and Equipment Certification). Software execution on microprocessors is, however, much more reliable than a communication channel's physical properties encountered during wireless transmissions. Besides the impact of destructive interference due to channel contention, misconfigured systems (intentionally or by accident) can also strongly degrade the network performance. It is thus imperative to design communication systems accordingly to minimize the negative impact of interfering transmissions on a network. The mechanisms presented in Sects. 3.3.1–3.3.3 can help to meet these requirements to a certain extent.

To combat adverse wireless channel conditions at even higher efficacy, emerging trends from the domain of networked embedded systems can be considered for implementation in avionics networks as well. A milestone in research on dependable communications between embedded systems has been reached with the presentation of protocols that exploit the physical phenomenon of *constructive interference* [20]. By means of transmitting identical signal waveforms simultaneously by multiple stations, greater wireless transmission ranges can be covered at higher reliability. Moreover, the eventual involvement of all networked devices in the communication network generally allow protocols based on constructive interference to reach high packet success rates and the message's dissemination to all devices in the network.

Another approach to reduce to negative impact of high contention on select channels is the use of channel-hopping protocols. Examples include as IEEE 802.15.4e in TSCH mode or IEEE 802.15.1 (Bluetooth). In these protocols, stations periodically select their communication channel according to predefined sequences. As opposed to the reactive nature of cognitive radios, channel-hopping protocols proactively balance traffic between the channels available in a given frequency band. In principle, constructive interference and channel hopping can be used in conjunction, yet no such design has been presented in literature so far. As a third contribution toward more reliable communications to be listed here, multipath protocol designs have been proposed [21]. Transmitting a message across multiple routes in parallel will increase its chances of success, particularly in lossy networks, at an energetic overhead that is often smaller than when relying on constructive interference. The number of paths used is a key parameter in multipath protocols, as it increases both the reliability and energy demand of the network's operation. The dense networks anticipated for cyber-physical systems in avionics will enable fine-grained adjustments to this parameter, and allow for its adaptation to diverse QoS requirements.

Besides above considerations to ensure a dependable communications substrate, meeting application requirements is equally important for the success of wireless networking in avionics. Application traffic in computer networks is generally considered in the form of end-to-end flows, i.e., traffic from a source device to one or more destination nodes. In wireless and wired networks alike, the data transmission may require the involvement of intermediate routers or relay nodes that forward traffic toward its destination. In order to accommodate the requirements to QoS (particularly delay and throughput), all devices along a route must be configured accordingly. As compared to traditional approaches (e.g., DiffServ [22] and IntServ [23]), the aforementioned concept of software-defined networking again proves necessary for the global consideration of traffic flows and resource reservations.

The need for data integrity and security protection may also arise from the application requirements. Authentication, i.e., limiting network access to devices that are crucial for the functionality to be provided by the network, takes a key role therein. A key constraint for the widespread use of network access control in avionics cyber-physical systems, however, is their operability on resource-constrained embedded devices. A trade-off between the achieved degree of protection and the resulting resource footprint thus needs to be found. It is often also left to application developers to decide which additional network security objectives (e.g., integrity protection, confidentiality) are needed. A methodological approach to accomplish information security in avionics networks is thus urgently required. The same constraints as listed above apply in this case as well, i.e., corresponding mechanisms must be suitable for their use on low-power embedded systems.

3.4 Networking Between Aircraft and Other Stations

Besides the anticipated growth in the use of onboard wireless communications, voice and data links to ground stations (directly or via satellites) as well as to other aircraft also take a key role during flight operations. Given the breadth of sensor data anticipated to originate from avionics sensor networks, a torrent of real-time updates from aircraft health and usage monitoring systems (HUMS) can be expected to become available. They are often vital for corporate operations and thus need transmission to ground stations within tight temporal bounds. Distances between nodes in such networks, however, are significantly larger than in the case on onboard LANs described above; consequently, a different family of technologies qualify as candidates for their transmission. Existing VHF-based communication technologies, e.g., aircraft communications addressing and reporting system (ACARS) and VHF Datalink (VDL) systems, are only partially applicable due to their limited data rates (on the order of tens of kilobits per second) and their cost of use when comprehensive HUMS data traffic shall be forwarded. Instead, a promising technology is the use of K_u-band satellite connections in the frequency range between 12 and 18 GHz, whose large bandwidth allocations allow for high throughput rates. Different to the local area networks used onboard aircraft, wireless long-range signals are even more impacted by adverse weather conditions or other sources of interference. In-depth analyses of routing based on current meteorological conditions, frequency stability considerations due to the aircraft's velocity, or handover processes in transition regions between coverage sectors need to precede the widespread use of K_u-band links. Lastly, security recommendations for communication between aircraft, satellites, and ground stations need to be considered, e.g., such as presented in [24].

3.5 Insights and Conclusions

Within the development toward the next generation of flight, a strong increase in the number of sensing, processing, and actuation components can be expected aboard aircraft. Their interconnection to create health and usage monitoring systems allows airlines and operators to gain unprecedented in-depth insights into operation conditions and the potential for real-time predictive maintenance. As highlighted above, their integration into a wireless cyber-physical avionics network will be inevitable to realize such services at scale [25].

A set of prerequisites must be fulfilled for their practical realization. First, an informed choice of the wireless communication standards and protocols must be made. Second, quality-of-service guarantees take a crucial role in wireless avionics networking, and consequently means to deterministically provide latency bounds and throughput guarantees are needed. Third, ways to coordinate and control the wireless traffic, in particular with regard to a contention-free wireless medium access, must be carefully chosen in order to ensure dependable, extensible, and secure networks. Fourthly and lastly, ensuring that these mechanisms are ready to cope with the anticipated growth in the number of devices over time, i.e., the network's scalability, is of particular importance.

Research on computer networks dates back almost 50 years, when the ARPANET was first deployed. To date, the world-encompassing Internet connects billions of systems globally, using technologies that have matured over years of their practical use. Many of the resulting technological advancements can be applied in avionics data networks to collect, process, and wirelessly transmit information within aircraft. In this chapter, we have surveyed corresponding emerging trends in computer networks which may serve as the fundament to make wireless avionics networks a reality.

References

1. J. Yu, B.M. Wilamowski, Recent advances in in-vehicle embedded systems, in *Proceedings of the 37th Annual Conference of the IEEE Industrial Electronics Society (IECON)* (2011), pp. 4623–4625
2. M. Tanaka, An industrial and applied review of new MEMS devices features. Microelectron. Eng. **84**(5), 1341–1344 (2007)

3. H. Kopetz, Internet of things, *Real-Time Systems: Design Principles for Distributed Embedded Applications* (Springer, Berlin, 2011), pp. 307–323
4. R. T. C. for Aeronautics. Minimum aviation system performance standards for automatic dependent surveillance broadcast (ADS-S). RTCA, Incorporated (2002)
5. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks. IEEE Commun. Mag. **40**(8), 102–114 (2002)
6. S.K. Khaitan, J.D. McCalley, Design techniques and applications of cyberphysical systems: a survey. IEEE Syst. J. **9**(2), 350–365 (2015)
7. P. Durand-Estèbe, V. Boitier, S. Berhouet, B. Labrousse, M. Bafleur, J.-M. Dilhac, Energy harvesting for wireless in flight testing on A321 aircraft, in *More Electric Aircraft* (2015)
8. D. Lee, G. Dulai, V. Karanassios, Survey of energy harvesting and energy scavenging approaches for on-site powering of wireless sensor and microinstrument-networks, in *Proceedings of the SPIE*, vol. 8728 (2013)
9. International Telecommunication Union. "Radio Regulations". In: ITU, 2012. Chap. 1.15: Industrial, scientific and medical (ISM) applications (of radio frequency energy)
10. Federal Communications Commission, Office of engineering and technology, policy and rules division. FCC online table of frequency allocations (2017), https://transition.fcc.gov/oet/spectrum/table/fcctable.pdf
11. CEPT Electronic Communications Committee. ERC recommendation 70-03 relating to the use of short range devices (SRD) (2017), http://www.erodocdb.dk/Docs/doc98/official/pdf/REC7003e.pdf
12. B. Wang, K.J.R. Liu, Advances in cognitive radio networks: a survey. IEEE J. Sel. Top. Signal Process. **5**(1), 5–23 (2011)
13. F. Hou, X. Chen, H. Huang, X. Jing, Throughput performance improvement in cognitive radio networks based on spectrum prediction, in *Proceedings of the 16th International Symposium on Communications and Information Technologies (ISCIT)* (2016), pp. 655–658
14. H. Kim, K.G. Shin, Efficient discovery of spectrum opportunities with MAC-layer sensing in cognitive radio networks. IEEE Trans. Mob. Comput. **7**(5), 533–545 (2008)
15. Z. Zhang, W. Zhang, S. Zeadally, Y. Wang, Y. Liu, Cognitive radio spectrum sensing framework based on multi-agent architecture for 5G networks. IEEE Wirel. Commun. **22**(6), 34–39 (2015)
16. L.C. Wang, C.W. Wang, Spectrum handoff for cognitive radio networks: reactive-sensing or proactive-sensing, in *Proceedings of the IEEE International Performance, Computing and Communications Conference (IPCCC)* (2008), pp. 343–348
17. K. Kirkpatrick, Software-defined networking. Commun. ACM **56**(9), 16–19 (2013)
18. D. Kreutz, F.M.V. Ramos, P.E. Veríssimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
19. J. Stringer, D. Pemberton, Q. Fu, C. Lorier, R. Nelson, J. Bailey, C.N.A. Corrêa, C.E. Rothenberg, Cardigan: SDN distributed routing fabric going live at an internet exchange, in *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)* (2014), pp. 1–7
20. F. Ferrari, M. Zimmerling, L. Thiele, O. Saukh, Efficient network flooding and time synchronization with glossy, in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks* (2011), pp. 73–84
21. K. Sha, J. Gehlot, R. Greve, Multipath routing techniques in wireless sensor networks: a survey. Wirel. Pers. Commun. **70**(2), 807–829 (2013)
22. K. Nichols, S. Blake, F. Baker, D. Black, Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. RFC 2474 (Proposed standard). Updated by RFCs 3168, 3260. Internet engineering task force (1998), http://www.ietf.org/rfc/rfc2474.txt
23. J. Wroclawski, The use of RSVP with IETF integrated services. RFC 2210 (Proposed standard). Internet engineering task force (1997), http://www.ietf.org/rfc/rfc2210.txt
24. E. Fleischman, R.E. Smith, N. Multari, Networked local area networks in aircraft: safety, security, and certification issues, and initial acceptance criteria (Phases 1 and 2). DOT/FAA/AR-08/31. U.S. Department of Transportation, Federal Aviation Administration, Air Traffic Organization Operations Planning: Office of Aviation Research and Development (2008)
25. K. Sampigethaya, R. Poovendran, Aviation cyber-physical systems: foundations for future aircraft and air transport. Proc. IEEE **101**(8), 1834–1855 (2013)

Chapter 4
IoT and Service Oriented Infrastructures for Flight 4.0

Christos P. Antonopoulos, Konstantinos Antonopoulos and Nikolaos S. Voros

Abstract Flight 4.0 represents a rapidly expanding research domain that brings IoT (Internet of Things) technology in the aviation domain. Based on various engineering domains such as Wireless Sensor Networks (WSNs) and embedded systems, Flight 4.0 systems are characterized by high degree of heterogeneity regarding various perspectives, such as communication, hardware, and software solutions. Additionally, in order to be well accepted by the end users, it is of paramount importance to exhibit high degree of configurability and flexibility so as to be applicable in diverse application scenarios. Aiming to address such objectives, this chapter attempts to identify the main aspects and tendencies toward a holistic end-to-end communication infrastructure for Flight 4.0 systems. In this context, and serving as a roadmap, the respective architectures should offer a homogeneous support to a wide range of WSN communication technologies and protocols, while being able to support time-constrained monitor, control, and configuration of critical Flight 4.0 infrastructure. In addition, such architectures must emphasize on the use of distributed components that are able to offer enhanced fault tolerance performance, a critical aspect for most modern aviation systems.

4.1 Introduction

Flight 4.0 represents an emerging research domain, where Cyber-Physical Systems (CPS) are employed for delivering end-to-end services for aviation and aeronautics. The term CPS refers to the tight integration of cyber and physical objects, which in the case of Flight 4.0 are distributed components employed in aviation

C. P. Antonopoulos · K. Antonopoulos · N. S. Voros (✉)
Technological Educational Institute of Western Greece, Antirio, Greece
e-mail: voros@teiwest.gr

C. P. Antonopoulos
e-mail: cantonopoulos@teiwest.gr

K. Antonopoulos
e-mail: kostasantonop@yahoo.gr

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_4

monitoring and control systems. The term "cyber objects" refers to any computing hardware/software resources that can achieve computation, communication, and control functions in a discrete, logical, or switched environment [1, 2]. Also, physical objects refer to any natural or human-made systems that are governed by the laws of physics and operate in continuous time. Wireless Sensor Networks (WSNs) and embedded systems comprise the two main cornerstones of Flight 4.0 with respect to communication, processing, and control considering extremely resource limited systems. However, in order for a system to be really useful and practical, it is critical to offer adequate performance capabilities and high degree of flexibility with respect to remote monitoring and actuation systems. In that respect, messaging [3] is a technology that enables high-speed, asynchronous, highly reliable machine-to-machine or even program-to-program communication. Such communication is possible by exchanging data packets, called messages, to each other.

Respective solutions offer important advantages in terms of remote communication, platform language integration, and asynchronous communication, following a send-and-forget approach. Additionally, communication reliability is also a major concern, typically utilizing a store-and-forward approach for transmitting messages, while asynchronous communication enables the decoupling of the sender from the receiver. Finally, it is worth noting that a messaging system acts as a mediator between all programs that can send and receive messages (either residing in the same physical computer or being distributed amongst different ones). However, message passing technologies are characterized by a complex programming model since asynchronous messaging systems require developers to work following an event-driven programming approach [3].

Driven by the necessity for holistic solutions, in this chapter a system-wide, end-to-end architecture for Flight 4.0 systems is envisioned offering critical advantages, features, and services. The respective technologies and proposed designs emphasize on implementing a robust as well as efficient infrastructure based on IoT (Internet of Things) technology, which is able to support high-performance fault-tolerant systems. It is argued that, based on message passing communication technology and utilizing Commercial-Of-The-Shelf (COTS) technologies, the respective architectures can yield an abstraction layer that hides effectively all technology heterogeneity and/or peculiarity. In Fig. 4.1, a high-level end-to-end architecture of an IoT platform targeting Flight 4.0 scenarios is depicted. With respect to this vision, the rest of the chapter is segmented into two main sections corresponding to the two main segments of the architecture. On one hand, the left-hand side of the architectures focuses on the basic sensors, modalities, and actuators being aggregated to the IoT system gateway interconnecting the cyber with the physical domain and dealing with specialized technologies tackling specific requirements and needs. On the other hand, the right-hand side of the figure corresponds to the required backend infrastructure, effectively interacting with the gateway level, which is critical when aiming to offer advanced services related to data storage, handling, processing, and presentation to the end user. Based on such core services, user end points related to Human-Machine Interface (HMI), optimal performance warranty, etc. as depicted in Fig. 4.1 can be developed and deployed.

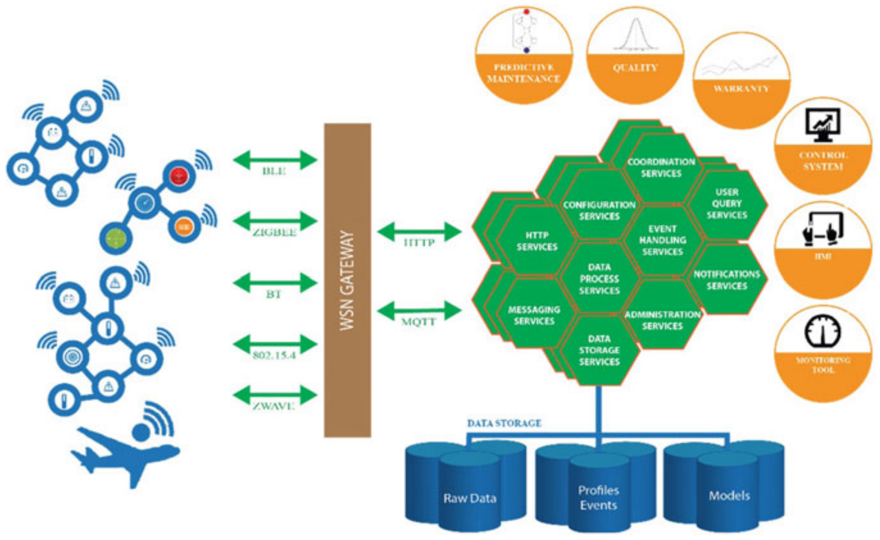


Fig. 4.1 Envisioned high-level IoT system for Flight 4.0 domain

4.2 Wireless Sensor Networks Paving the Way of IoT

One of the cornerstones of IoT are the wireless communication technologies that are able to tackle the challenges created by the highly volatile, dynamic, and unpredictable paradigm IoT has introduced. In this section, focusing on Flight 4.0 and respective demanding scenarios, an attempt is made to highlight the most important requirements and present a set of prominent mature communication protocols.

4.2.1 Technical Requirements

In this section, we identify the main requirements based on which adequate wireless communication technologies should be adopted, configured or even extended so as to meet the objectives of demanding IoT/Flight 4.0 application scenarios. The respective requirements stem both from the user requirements as well as the pure technical demands of a state-of-the-art wireless sensor network.

4.2.2 Low-Power Operation

The ability of a wireless communication to promote power consumption minimization comprises a characteristic of paramount importance. Although in aviation domain, scarce energy availability does not strike as critical problem, one should consider that IoT systems in Flight 4.0 effectively comprise by a group of highly distributed mobile embedded systems. Each component can have its own autonomous power supply; it can be mounted on a person, on a device or an aircraft. But what is critical to bear in mind is that all components must operate adequately at all times; otherwise, the whole system may be compromised. Meeting such objective is a multifaceted task involving many critical aspects of a wireless communication platform concerning both hardware and software aspects [4, 5].

4.2.2.1 Data Throughput Capabilities

Without a doubt, the data volume, per specific time period, a communication technology can handle comprises a critical metric in all network types. Even more, in scenarios involving highly complex and critical components such as in aviation domain, the respective data volumes are anticipated to be rather high. Furthermore, highly dynamic factors such as concurrent traffic flows and how they are accommodated, especially in the case of burst traffic, make the whole problem even more challenging. Throughput capabilities depend upon the mechanisms a wireless technology employs to assure robust communication. Through such mechanisms, each wireless communication approach aims to, on one hand, avoid packet collisions and, on the other hand, accurately and rapidly identify a packet collision and recover from it. Such mechanisms include efficient clear channel assessment mechanisms, CTS/RST control packet exchange, CRC (Cyclic Redundancy Check), and acknowledgment mechanisms support.

4.2.2.2 Delay–Jitter Requirements

Similarly to throughput capabilities, mean delay and the respective requirements (although posing relatively relaxed demands in typical IoT networks) pose partially difficult to tackle, problems in Flight 4.0 domain. This is because in aviation the respective demands are equivalent to industrial domain and there are hard real-time requirements. The former is related to the fact that the time windows between a request and the respective response or an event detection and the respective actuation are particularly small. The latter is related to the fact that if the posed deadlines are not met the whole system may be compromised with possibly catastrophic consequences in terms of cost as well as human lives.

4.2.2.3 Quality of Service Support

As WSN technologies are considered as a viable solution in demanding application scenarios such as Flight 4.0 and users tend to pose increasing demands upon the respective deployments, the need for advanced quality of service capabilities is analogously increasing and is expected to do so in the future. By quality of service, we refer to the differentiated handling among data traffic flows with respect to various (and dynamically changing) parameters as well as the differentiation among users. With respect to such differentiation, a communication technology can apply appropriate methodologies to meet the different requirements posed by each category in each particular time (to the degree that is possible). It is also valuable to allow the developer to apply custom algorithms to apply specific policies. In the MAC layer, such capabilities are typically dependent upon the access scheme utilized, typically CSMA, TDMA, or a combination of these two.

4.2.2.4 Security Support

Last but not least, security support comprises one of the most important and critical features of Flight 4.0 approach. This is because aviation represents a highly critical IoT application domains which directly relates to safekeeping of highly private personal data as well as the adequate management and distribution of them. Security support is comprised of three services: *data privacy*, *authentication*, and *authorization*. The first one relates to the assurance that data can be "understood" (i.e., be useful information) only by nodes that have the right to do so, while for the rest of the nodes effectively are useless bytes. This service typically relies on the capabilities of WSN nodes to execute sophisticated encryption algorithms offering high security level. Consequently, the inherent support of such cipher algorithms by the used communication technology is a significant added value. The second service allows nodes to prove they are who they claim to be, which effectively allows accountability and is a prerequisite for the third service which effectively determines what level of access can a specific user have to certain data or/and functionalities.

4.2.3 State-of-the-Art IoT Technologies for Flight 4.0

Nowadays, a plethora of short-range, ultralow-power wireless communication technologies are available, all aiming at meeting the requirements posed by IoT application scenarios. In the context of this chapter and aiming toward future Flight 4.0 adequate IoT systems, a set of prominent technologies is presented comprising mature solutions integrated into wide range of commercial platforms targeting diverse application domains. The goal of this effort is mainly to extract and highlight the respective characteristics (as well as differences) enabling the optimum selection of the appropriate technology with respect to specific application scenario requirements.

4.2.4 IEEE 802.15.4 Based Solutions

Solutions of this kind are prominent candidates as they are utilized in several experimental and commercial scenarios. Their communication capabilities are based on the IEEE 802.15.4 standard finalized by October 2003 [6]. Their popularity is based on significant advantages when aiming toward very low-power, low-complexity, low-price, and low application demand characteristics. At the physical layer, IEEE 802.15.4 offers three possible frequency ranges, although the most popular is the 2.4 GHz ISM band where 16 channels can be utilized offering the highest bit rates equal to 250 Kbps [6]. However, it is noted that at each particular moment, only one channel can be used, thus not being a multichannel protocol. Concerning the data transfer approaches, although IEEE 802.15.4 defines approaches for both contentionless and contention-based access schemes, the respective platforms usually implement and utilize only simple contention Carrier Sense Multiple Access (CSMA)-based approaches. Following such an approach, all nodes are peers (i.e., there is no coordinator) and sense the transmission medium for two reasons. On one hand, if a node wants to transmit a packet, it senses the medium until it is identified as idle and then transmits the packet. On the other hand, from the receiver perspective, a node senses the transmission medium in order to identify a packet transmission toward itself. Popular IEEE 802.15.4 based platforms include TelosB and MicaZ. TelosB comprises probably the most well-known WSN platform upon which many projects have been based on [7, 8]. As expected, it offers an IEEE 802.15.4 compatible RF transceiver which can deliver 250 Kbps bitrate at the 2.4 GHz frequency band. Processing is based on the widely utilized Texas Instrument MSP430 16-bit microprocessor. Concerning memory capabilities, the developer is provided with a 48 KB program flash, 10 KB data RAM, and a 1MB external flash. It is noted that, although now TelosB is available through Memsic Co. [9], there are other platforms which under different brand names offer identical characteristics such as [8, 10]. Shimmer (2R version) is also a very interesting case since it offers a very versatile environment integrating a wide range of sensors. MicaZ also comprises a prominent platform used in various scenarios, offering analogous characteristics as TelosB. The main difference concerns the processing module, which is based on an Atmel ATmega128L for both the radio and processing tasks, offering 128 KB program and 512 KB data memory, while TelosB motes are USB programmable and chargeable, thus offering enhanced usability compared to MicaZ where a separate programming module is required.

4.2.5 Bluetooth-Based Solutions

Bluetooth is a wireless radio specification designed to replace cables as the medium for data and voice signals between electronic devices. The specification is defined by the Bluetooth Special Interest Group (SIG) which is made up of over 1000 electronics' manufacturers. Intended primarily for mobile devices, Bluetooth's design places a high priority on small size, low power consumption, and low costs. The Bluetooth specification seeks to simplify communication between electronic devices by automating the connection process. Bluetooth radios operate in the unlicensed 2.4 GHz Industrial, Scientific, and Medical application (ISM) frequency range. This frequency is already widely used by devices such as microwave ovens, baby monitors, cordless telephones, and 802.11b/g wireless networking devices. In order to avoid interference from these devices, Bluetooth uses a technology called spread spectrum frequency hopping. Spread spectrum frequency hopping changes the transmission frequency up to 1,600 times per second across 79 different frequencies. As a result, interference on any one of those frequencies will only last a fraction of a second. This, coupled with the limited range of Bluetooth radio transmitters, results in a robust signal that is highly tolerant of other devices sharing the same frequency. Bluetooth devices automatically attempt to communicate whenever one device comes within the range of another. Bluetooth devices discover each other and initiate communication via inquiry and page transmissions. In addition, they have the ability to form ad hoc networks. The topology of these networks is both temporary and random. An ad hoc network of two or more Bluetooth devices is called a piconet. When two Bluetooth devices initiate a connection, they automatically determine if one device needs to control the other. Generally, the device that initiates the communication assumes the role of master and exercises certain controls over the other members of the piconet which are known as slaves. Upon establishing a piconet, the slave devices synchronize their frequency hopping sequence and system clock with that of the master in order to maintain their connection. A master device can have up to seven slaves. A slave in one piconet can also be the master in another, thus allowing piconets to overlap and interact forming what is known as a scatternet. Contrary to IEEE 802.15.4 based solutions, where all relative platforms are characterized by analogous capabilities, the platforms in Bluetooth-based solutions can vary significantly depending both on the version of the protocol supported, and even more on the characteristics of the specific implementation. Regarding data rates, solutions covering a wide range from 300 Kbps up to 1.5 Mbps can be found. Indicative examples of relative solutions include Shimmer [8] and MoviSens [11] platforms. The former is utilized in the Roving network-based Bluetooth modules [12].

4.2.6 Bluetooth Low Energy Based Solutions

Bluetooth Low Energy (BLE) represents a different technology from classic Bluetooth (and in fact incompatible technology) being promoted by the Bluetooth Special Interest Group (SIG). Benefiting by the huge successful classic Bluetooth, it shows significant dynamics compared to analogous technologies being incorporated in most mobile and embedded devices. Furthermore, it offers a high degree of flexibility concerning both implementation and communication approaches. Thus, multiple ways for nodes to communicate are supported through different data structure profiles so as to best fit the application requirements. Being a protocol in progress regarding various aspects, it also offers the added value of allowing developers to become members of the Bluetooth SIG, gain valuable insight on future features of the standards, and if possible even proposal respective approaches.

4.2.7 Bluetooth Low Energy Device Interfaces

Bluetooth Low Energy (BLE), a distinctive feature of the Bluetooth 4.0 specification, constitutes an emerging wireless technology for short-range communication which is expected to be increasingly employed in numerous control and monitoring applications that require low-power operation. Below, the main characteristics and mechanisms of the Bluetooth protocol stack are described, as well as the referenced topology and network roles which refer to the nodes in a Bluetooth network [13].

Based on the structure of the protocol stack used in previous versions, the BLE protocol stack consists of a host and a controller, which are separated by a standardized interface, the *Host Controller Interface (HCI)*. As shown in Fig. 4.2, the host

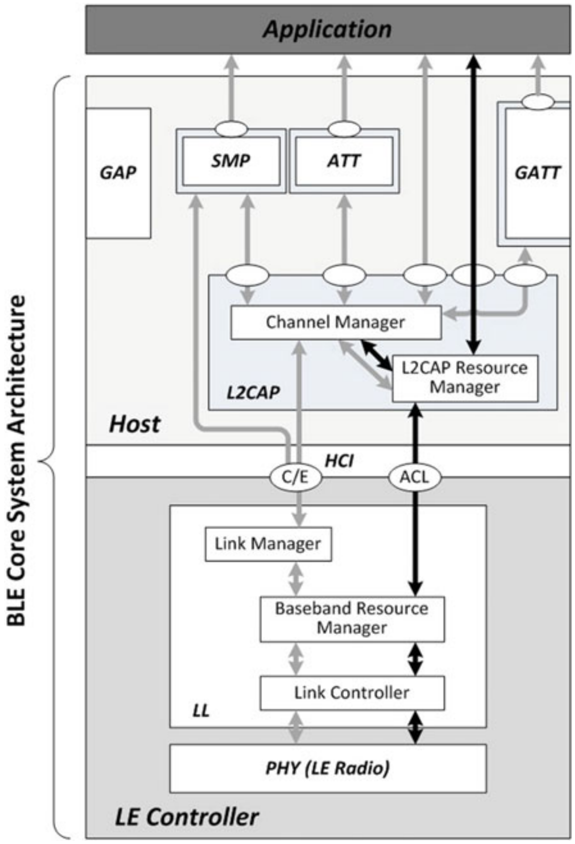


Fig. 4.2 Bluetooth Low Energy (BLE) core system architecture

is a logical entity defined as a set of layers below the application layer (or non-core profiles) and above HCI, typically running on an application processor and including the *Logical Link Control and Adaptation Protocol (L2CAP)*, the *Attribute Protocol (ATT)*, the *Generic Attribute Profile (GATT)*, the *Security Manager Protocol (SMP)*, and the *Generic Access Profile (GAP)*. On the other hand, the controller is defined as a set of layers below HCI, typically being implemented on a small *System-on-Chip (SoC)* that provides wireless communication via a radio module and including the physical layer and the link layer. An implementation of the host and the controller may contain the respective parts of the HCI, the upper, and the lower ones, respectively.

BLE defines 40 radio frequency channels in the Industrial Scientific Medical (ISM) band of 2.4 GHz with 2 MHz channel spacing. However, those PHY channels are used for different operations, namely advertising and data transfer, thus resulting in three advertising channels which are used for broadcast transmission, device discovery, and connection establishment, and 37 data channels (the rest of the band) which are used for bidirectional communication between connected devices. In order to deal with the interference imposed by other wireless technologies using the 2.4 GHz band (e.g., Wifi), as well as wireless propagation issues in general (i.e., multipath, fading), an adaptive frequency hopping mechanism is used, selecting one of the 37 available data channels for communication between connected devices during each particular time interval (the sequence of which is determined at the beginning of the connection), as shown in Fig. 4.3. The reached data rate is about 1 Mb/s. The link layer provides the basic acknowledgement/repeat request (ARQ) protocol in BLE. When a device only needs to broadcast data, it forms advertising

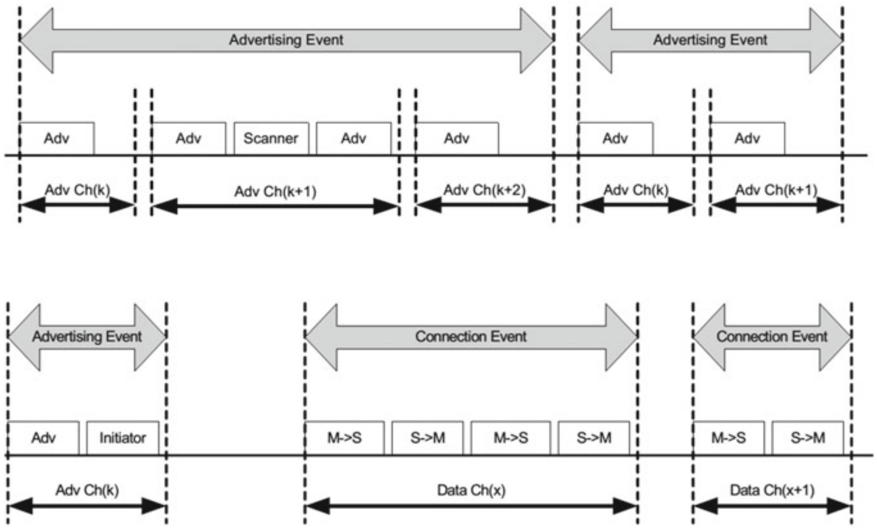


Fig. 4.3 Advertising events and connection events

packets and transmits them over the advertising PHY channels in the time intervals defined by the advertising events, and therefore it needs to become an advertiser. On the other hand, when a bidirectional data communication between two devices is required, they need to be connected to each other, thus forming a piconet, where the advertiser device becomes a slave device and the initiator device becomes the master device.

4.3 IoT Backend Infrastructure for Flight 4.0

Backend infrastructures comprise the second major cornerstone of IoT since they represent that base upon which applications, functionalities, and services are designed, implemented, commissioned, and managed. As mentioned in introduction and in relation to Fig. 4.1 respective upon respective technologies, all critical core services as well as end-user applications are developed. The main goal of this section is, on one hand, to identify the main goals/challenges such infrastructure should meet to offer viable solution to Flight 4.0 application scenarios. On the other hand, a well-focused survey is offered concerning prominent existing solutions highlighting the most relevant characteristics, features, and advantages each offers toward addressing presented challenges.

4.3.1 Key Challenges

Driven by the requirements posed by Flight 4.0 applications, we need platforms that have been designed specifically to address all these critical emerging challenges. Additionally, the respective requirements are considered in conjunction with the rapid and multifaceted advancements in embedded systems and end-to-end wireless communication technologies.

During the last years, IoT platforms have been employed for supporting Flight 4.0 applications of increasing complexity. Additionally, the respective complexity is quite multifaceted since it concerns communication, processing as well as data storage aspects of a complete Flight 4.0 system. Consequently, a critical objective for these platforms is to address all these challenges in the most efficient and versatile way.

Regarding the communication perspective of Flight 4.0 systems, typically, a short-range wireless communication protocol enables data aggregation to a central point (often indicated as Gateway). Although a plethora of different communication technologies (mainly originating from WSN research domain) are available, they offer diverse characteristics exhibiting high degree of incompatibility. In that respect, the forthcoming platforms must offer highly efficient gateway designs able to support the most prominent short-range wireless communication technologies such as IEEE 802.15.4 [14], ZigBee [15], Bluetooth [13], BLE [16], and Z-Wave [17]. Also, a critical goal of the design is to facilitate the continuous development and integration of new solutions. Communication complexity, however, is also related to the efficient transfer of data between the gateway and remote installations like service providers, databases, graphical user interfaces, etc. Consequently, forthcoming platforms must be enforced to use more contemporary approaches than the traditional HTTP communication approach, such as message passing communication aiming to support increased communication complexity.

Furthermore, the increased functional complexity calls for new processing capabilities in the future Flight 4.0 systems. In order to address the respective challenge, novel gateway designs must have the ability to support data acquisition by different modalities as well as exploiting different communication technologies. Additionally, the acquired data are synchronized and processed in a homogeneous way. In this way, sophisticated load balancing, data merging, QoS, prioritizing, and many more mechanisms can be supported. This approach also facilitates real-time data processing and event detection which is of paramount importance in demanding applications in the aviation domain. Another critical aspect of processing has to do with the ability to handle big data. This is due to the fact that Flight 4.0 systems can be effectively comprised of thousands of nodes potentially creating huge volumes of data. Given that the generated data are heterogeneous, the traditional model with homogeneous database schemes cannot be longer used. The forthcoming platforms must adopt heterogeneous database schemes, where each database scheme must be utilized for a single purpose, which perfectly fits the data that would be stored. Based on such technologies, the database to be formulated should be characterized by an increased degree of flexibility and reliability, where consequently highly efficient data queries can be defined, facilitating fast and accurate data analytics.

Data storage represents also an aspect where novel approaches must be adopted, especially with respect to distributed, networked, and collaborative systems. Forthcoming Flight 4.0 systems must be designed with a database storage plan, which supports a full set of distributed operations, utilizing a heterogeneous database scheme using SQL and NoSQL databases [18]. Based on this approach, a highly efficient interconnected and collaborative constellation of database facilities is provided that is able to support even the most demanding application requirements. Another critical advantage of the aforementioned distributed approach concerns the increased fault tolerance through the complete avoidance of single-point failure scenarios. It is also noted that the collaborative functionality is necessary to be provided also at gateway level, for such platforms. Specifically, using message passing communication technologies, it is possible to support complex collaborative communications between gateways, creating an ecosystem of intelligent processing entities that is able to better monitor, encode, and even interact with the physical world. In this way, the physical environment and the human activities can be effectively addressed.

Flight 4.0 systems are usually comprised of a number of highly scattered nodes being able to interact with the real world, thus supporting a wide range of diverse applications. Therefore, controlling, optimizing, adjusting, and enhancing a continuously expanding Flight 4.0 system by physically accessing each node or/and component is unpractical, cumbersome, and in many cases virtually impossible.

Consequently, it is critical to support real-time remote control of all components as well as on-the-fly functional enhancements and service optimization. Both these challenges are of paramount importance for a Flight 4.0 system design. Remote monitoring, management, and control are possible using message passing communication technologies like MQTT and MQTT-SN. Finally, a valuable characteristic of such a platform would be the capability to provide end users/service providers an easy way to implement, test, and finally deploy a new service.

4.3.2 State-of-the-Art Backend Infrastructure for Flight 4.0 and Beyond

Flight 4.0 is heavily dependent on communication for conveying sensor observations to controllers/actuators; thus, the design of the communication architecture is a critical requirement for system functionality. Although Flight 4.0 is an emerging domain, the IoT technology employed has been used in other domains as well, where the architectural evolution has been shifted into systems' integration, ensuring that Service-Oriented Architecture (SOA) will have a major role in many branches of technology.

In [19], it proposed an SOA-based middleware architecture, "WebMed", through which an interaction with physical devices becomes as easy as invoking a computation service. Emphasizing the basics of service-oriented guidelines, it is built a loosely coupled infrastructure that exposes the functionality of physical devices to the Web for application development.

In [20], it is proposed another SOA-based architecture using a standard-based software component technology, divided into a five-tier architecture, namely perceive, data, service, execution, and security tier. Although SOA implementations can be a solution for the requirements of specific companies, these implementations are expensive, time consuming, and complex. With regards to agility, the addition of new requirements for end users in SOA implementations requires a lot of complex configurations, which consume a lot of time.

In [21], a commercial IoT platform, enControl, is proposed as an end-to-end smart home solution. End users of EnControl are able to monitor and remotely control their homes through any internet-enabled device. The EnControl platform is divided into three main components, namely, Home devices, IoT Platform, Graphical User Interface. At home level, EnControl is composed by end devices (sensors and actuators) and Gateways, which communicate using Z-Wave radio technology. The communication between the home level devices and the core system is achieved by a REST API implementation, in which end user can trigger actions, using EnControl graphical user interface.

Another commercial IoT platform, PLAT.ONE [22], provides a complete solution for IoT. It utilizes a mixed SOAP-based and REST-based architecture for data collection and integration. In addition, it offers a graphical user interface for users to control the data streams and to develop their own components in order to control and view data streams. As far as implementation is concerned, it is implemented as an extendable and heterogeneous data storage scheme using SQL and NoSQL databases for storing data streams.

Finally, LairdTech [23] proposes an end-to-end platform, specialized for connecting cars. LairdTech has implemented a complete solution for cars, in which embedded devices communicate using Bluetooth low energy wireless technology, in order to collect data from different sensors and on top of that implement application for analysis.

All the above proposals although comprising valuable sources of information, ideas, and design aspects, they are also characterized by critical limitations and shortcomings. Probably, the most important concerns the fact that they are closed platforms not allowing the developer to apply and test novel approaches and extensions. Furthermore, all already existing solutions are based on specific communication protocols and technologies, thus offering low degree of adaptability to continuously evolving ICT domains. The same shortcoming comes into play with respect to how services can be developed and easily deployed, and multiple different databases can be supported according to specific requirements and many more engineering design and implementation aspects. In addition, solutions proposed for Flight 4.0 must target tackling heterogeneous communication technologies, using the MQTT-SN [24, 25], in order to hide the underlying complexity created by respective infrastructures. Last but not least, to avoid the problems of monolithic applications and take advantage of the SOA architecture benefits, the microservice architecture pattern has emerged as a lightweight subset of the SOA architecture pattern [26].

4.4 Conclusion

Over the last few years, Internet of Things and Service-Oriented infrastructures appear as prominent and mature research areas able to unite the physical and cyber domains facilitating the expansion of demanding application scenarios such as Flight 4.0. However, in order, the respective solutions need to effectively tackle heterogeneity while exhibiting high degree of flexibility and configurability. Driven by this requirement, this chapter tries to identify, present, and highlight the main requirements and challenges involved in such endeavors. To achieve this objective, a complete end-to-end architecture is divided into two main segments: On one hand, the IoT, wireless sensors part pertaining to flexible, robust, and versatile interaction with a wide range of heterogeneous sensors and actuators; and on the other hand, the backend architecture offering all required infrastructure allowing efficient data storage, analytics, monitoring, processing, and presentation. Finally, concerning both segments of a complete solution, state-of-the-art technologies are presented able to support the required services and offer competitive advantage for current but most importantly future Flight 4.0 platforms.

References

1. F. Hu, *Cyber-Physical Systems: Integrated Computing and Engineering Design* (CRC Press, Boca Raton, 2013)
2. S.K. Khaitan, J.D. McCalley, Design techniques and applications of cyberphysical systems: a survey. IEEE Syst. J. **9**(2), 350–365 (2015)
3. G. Hohpe, B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* (Addison-Wesley Professional, Boston, 2004)
4. J. Blanckenstein, J. Klaue, H. Karl, A survey of low-power transceivers and their applications. IEEE Circuits Syst. Mag. **15**(3), 6–17 (2015)
5. C.P. Antonopoulos, N.S. Voros, A data compression hardware accelerator enabling long-term biosignal monitoring based on ultra-low power IoT platforms. Electronics **6**(3), 54 (2017)
6. I. 8. L. S. Committee et al., IEEE Standard for Information technology- Telecommunication and information exchange between systems-Local and metropolitan area networks-Specific requirements Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendmentl: Radio Resource Measurement of Wireless LANs (2009), http://standards.ieee.org/getieee802/download/802.11.n-2009
7. D. Malan, T. Fulford-Jones, M. Welsh, S. Moulton, Codeblue: an ad hoc sensor network infrastructure for emergency medical care, in International workshop on wearable and implantable body sensor networks, Vol. 5 (Boston, MA, 2004)
8. Wireless Sensor Technology, Wireless IMU, ECG, EMG, GSR (2017), www.shimmersensing.com/
9. Memsic Leader in MEMS Sensor Technology (2017), www.memsic.com/
10. MTM-CM5000-MSP (2017) www.advanticsys.com/shop/mtmcm5000msp-p-14.html
11. movisens GmbH, (2017), www.movisens.com/en
12. Revoking Networks, Advanced User Manual Version 4.77. (2011)
13. Bluetooth Special Interest Group, Bluetooth Core Specification v4.0. (2010), www.bluetooth.org/en-us/specification/adopted-specifications
14. IEEE, IEEE 802.15.4-2015 IEEE Standard for Low-Rate Wireless Networks. Technical Report (IEEE, 2015)
15. Z. Alliance, ZigBee specification, 2008, in *ZigBee Document 053474r17* (2008)
16. S. Bluetooth, Specification of the bluetooth system-covered core package version: 4.0. (2010)
17. The open-zwave Open Source Project on Git Hub (2017), www.openhub.net/p/open-zwave
18. I. Mapanga, P. Kadebu, Database management systems: a nosql analysis. Int. J. Mod. Commun. Technol. Res. (IJMCTR) **1**, 12–18 (2013)
19. D.D. Hoang, H.-Y. Paik, C.-K. Kim, Service-oriented middleware architectures for cyber-physical systems (2011)
20. L. Hu, N. Xie, Z. Kuang, K. Zhao, Review of cyber-physical system architecture, in 15th IEEE International Symposium on.Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2012 (IEEE, 2012), pp. 25–30
21. Sensing and Control Systems (2017), www.sensingcontrol.com/
22. Data aware platforms deliver a differentiated service in M2M, IoT and Big Data (2017), www.openhub.net/p/openzwave
23. Vehicle Connectivity Solutions (2017), www.lairdtech.com/product-categories/vehicle-connectivitysolutions
24. U. Hunkeler, H. L. Truong, A. Stanford-Clark, MQTT-S-A publish/subscribe protocol for wireless sensor networks, in 3rd International conference on communication systems software and middleware and workshops, 2008 comsware (IEEE, 2008), pp. 791–798
25. A. Stanford-Clark, H. L. Truong, MQTT for sensor networks (MQTTSN) protocol specification version 1.2 (2008), http://mqtt.org/new/wpcontent/uploads/2009/06/MQTT-SN_spec_v1.2
26. M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas, S. Gil, Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud, in 10th Computing Colombian Conference (10CCC), 2015 (IEEE, 2015), pp. 583–590

Chapter 5
Big Data and Data Analytics in Aviation

Gerrit Burmester, Hui Ma, Dietrich Steinmetz and Sven Hartmannn

Abstract Big Data technology in the field of aviation has emerged in recent years. Continuously growing amounts of data sources such as sensors, radars, cameras, weather stations, airports, etc. produce terabytes of high dynamic data each second. The future aviation concepts require modern data storing, data processing, and data analyzing technologies. The extraction of meaningful knowledge from the given data is a major challenge, trends, cross-connection, correlations, etc. have to be identified. Real-time critical tasks increase additionally the technology requirements and need innovative solutions. The application of Big Data technology in aviation context optimizes safety aspects, fuel consumption, maintenance processes, flight scheduling, etc. This chapter describes a process of Big Data application and summarizes relevant actual Big Data methods in the aviation domain.

5.1 Introduction

Aviation is growing steadily, which contributes to increasing amount of data sets in this industry and leads to growing interest in using Big Data technologies [1]. The term Big Data refers to the big volume of the data sets in large numbers, their variety, and the velocity requirements of the provision of the data. Furthermore, it is characterized by variability, which addresses the consistency of a data set and veracity, which refers to the quality of a data set. Both result due to weak structure of a data sets, missing or inaccurate values [2, 3].

G. Burmester · D. Steinmetz · S. Hartmannn (✉)
Clausthal University of Technology, Clausthal-Zellerfeld, Germany
e-mail: sven.hartmann@tu-clausthal.de

G. Burmester
e-mail: gerrit.burmester@tu-clausthal.de

D. Steinmetz
e-mail: dietrich.steinmetz@tu-clausthal.de

H. Ma
Victoria University of Wellington, Wellington, New Zealand
e-mail: hui.ma@ecs.vuw.ac.nz

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_5

In aviation, Big Data has been receiving a wide interest. It is getting more and more important due to ever increasing amount of data through integration of emerging data sources, new technologies, increasing number of aircraft, and sensors in the past decades. The analysis and interpretation of produced data has the potential of saving resources such as operating costs or energy consumption and predict informations for decision-making, e.g., emergency situations [4]. Huge amounts of fast changing heterogeneous data are available in aviation through various sources such as all types of sensors from aircraft and airport, civil and military databases, maintenance centers, social media platforms, or Internet in general. The composition of data integration, data management, data preprocessing, data mining, and data visualization prepares, analyzes, and presents the data to be used in long term and real time. The results are provided for all stakeholders from flight crew and passengers to maintenance crew and system designers [5].

This chapter provides an overview about the Big Data and data analytics and their applications in aviation. It will further elaborate on the domain-specific challenges.

5.2 Big Data: An Overview

The literature offers many definitions of Big Data [6–9]. Generally, these describe Big Data as technology, which process complex data, high dynamic data, etc., which cannot be handled in traditional manners. It is classified through five characteristics [10–13]:

- **Volume**: The quantity and the number of the data sets can be so big that conventional methods cannot analyze the huge amount of data. In aviation, this includes all data sources from aircraft, airports, and institutions somehow connected to them, which could be databases of maintenance centers, weather stations, satellite networks, and the Internet in general.
- **Variety**: The data sets can have heterogeneous formats such as tables, videos, vectors, and more. Those data sets cannot be integrated easily, because of different structures, schema, and the general problem of merging sources, such as schedules of different airports, weather, and radar data or pilot-assistant systems.
- **Velocity**: The time to generate, analyze, and process data can be very small with respect to its volume and the speed of the growing of data sets. For example, there are around 5.000 sensors in a Pratt & Whitneys Geared Turbo Fan (GTF) generating over 10 GB of data per second to be processed in real time and evaluated with other Big Data sources [14].
- **Variability**: There can be inconsistencies in the data sets, which make it harder to process them. Such failures can arise from faulty sensors, errors in data collecting, or also through system failures.
- **Veracity**: The quality of the data sets can vary inside and between the different sets. An example in aviation is the reports of maintenance services, which can

differ in its execution or differences in the quality of sensors leading to uneven outputs.

The three characteristics volume, variety, and velocity are the most important prerequisite to define Big Data, since some data sets can have negligible problems with variability and veracity. The combination of the three main characteristics is mandatory for a use case to be considered as a problem, where Big Data technologies have to be used. In aviation, the size of data sets and the amount of data generated per second, analyzed, and interpreted for an application can be very high. There can be cases in terabytes per second scale, where the data sets have heterogeneous formats and types [14]. The quality of the collected data can vary in great extent also. It may require to be processed and transformed, concerning its quality and consistency. Deficiencies while recording or logging usually lead to incomplete data sets and eventually affect any analysis pattern itself. The data can have various types, but even the same type of data may require to be merged through robust and reliable procedures, which prevent creation of incorrect or even redundant data sets.

5.3 Big Data Challenges in Aviation

The ever increasing number of manned and unmanned systems in the last decades is leading to various challenging Big Data problems. Platforms produce data through communication between each other and their automation and autonomy require data-intensive activities such as machine vision and machine learning. Furthermore, networks of manned and unmanned platform and ground entities are envisioned to be coordinated and cooperated through shared data. Last but not the least, the number of platforms is expected to increase exponentially with the rise of urban aviation projects such Uber Elevate [15]. This can lead to the integration of problems that have so far been less strongly investigated in the aviation sector, such as platoon routing problems, which have been well researched for ground vehicles but only initially for aircrafts [16, 17]. Thereby, even more rapidly changing, heterogeneous data regarding fulfillment of user needs and safety of all participants has to be considered. Additionally, the interpreted information, such as historical data, is also growing rapidly in aviation.

Utilization of Big Data is becoming critical to avoid accidents, especially for unmanned platforms. For this purpose, large data stream arose from the communication between the aircraft and other sources have to be analyzed in real time to adjust, for example, the trajectory to prevent mid-air collision.

In many fields of aviation such as logistics, customer services, pilot, and passenger safety or marketing, big amounts of data needs to be comprehended to improve the processes regarding safety, efficiency, productivity, and economy. There is a growing interest in developing Big Data platforms for aviation. The civil aircraft Big Data platform which is proposed, in [18], collects data from aircraft, airlines, airports, and other services connected to aviation to provide an airline schedule and maintenance plan besides other benefits such as better service for customer and a bigger throughput of them, earlier adaption of weather conditions, savings in fuel and time, or also a better time management for passengers. In [19], a cloud-based avionics data system is presented with efficient functions of data collection, data classification management, storage, and analysis. The aim is to process big amounts of heterogeneous data in real time and with a good performance and stability.

5.4 Big Data Analytics

Large number of sensors, which are embedded in aircraft, and other aviation data sources produce large data sets every second. Further, real-time or near real-time analysis of data is utmost importance for flight-relevant predictions and especially in emergency situations. Collected and aggregated data needs to be preprocessed for data mining. Discovering and extracting meaningful pattern from large data sets includes machine learning and statistical techniques. Well-defined representation of discovered knowledge is necessary for its interpretation and consumption.

Big Data analytics involves five steps, which are essentially independent of its application field. Figure 5.1 depicts these steps applied to avionics.

Data from various heterogeneous sources such as sensors, cameras, radar, or weather have to be merged into a unifying structure. Initially, the data integration step combines different data sources to provide a unifying view of the data. Data management then comprises the provision and extraction of the relevant data sets to be analyzed. Data preprocessing cleans, normalizes, and reduces the data, to prepare it for the analysis process. Predictive and descriptive data mining techniques can then be applied to extract meaningful informations. Knowledge presentation is the final step of the Big Data analytics, which represents the extracted knowledge for different aviation applications.

5.4.1 Data Integration

In the first step of any approach, which processes Big Data, all data sources have to be merged, which means the seamlessly reconciling of various and autonomy relevant sources of data with heterogeneous structures into a uniform and suitable structure. In aviation, the sources contain various data warehouses and data streams from aircraft and airports as well as the inputs from their sensors and cameras and other objects inbound through the Internet of Things. Further sources contain weather and radar data, navigational charts, and furthermore Internet sources for any relevant information including social media and other platforms, which can be only weakly connected to the data sets and informations may only be deducted indirect through metadata. Also, many airlines participate in programs, which improve the general safety of flights such as Flight Operations Quality Assurance or Airline Safety Action

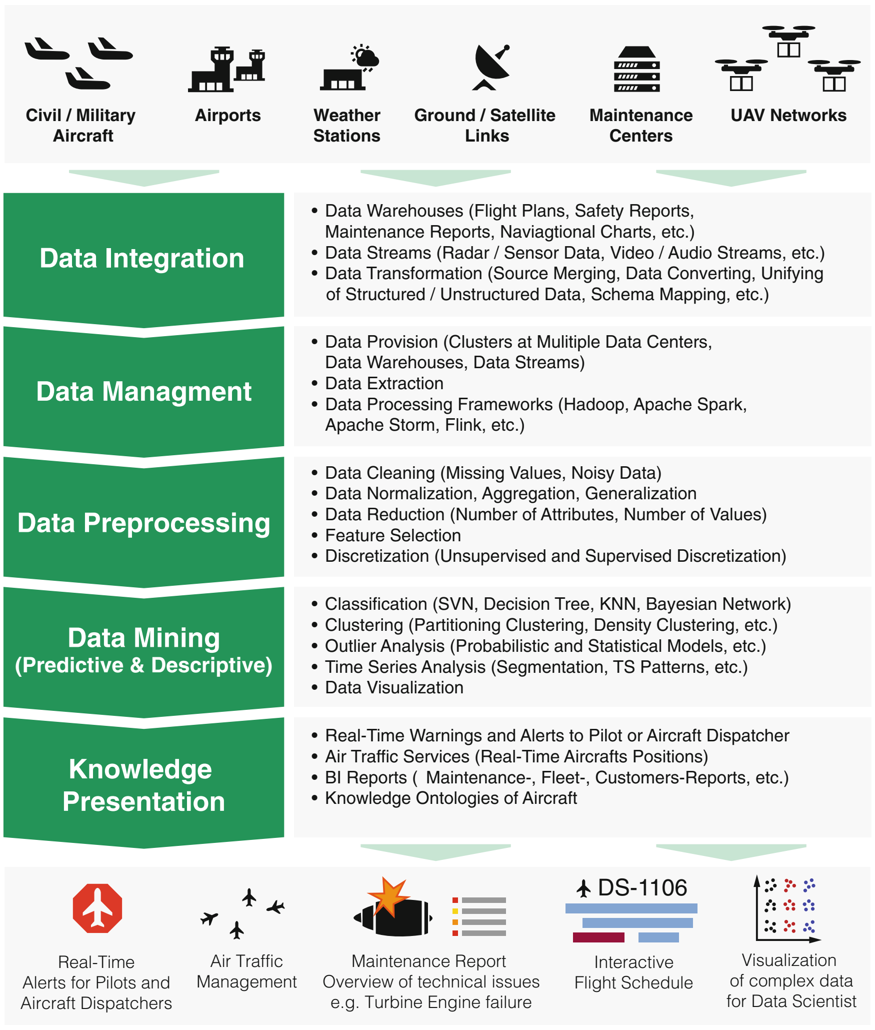

Civil / Military Aircraft
Airports
Weather Stations
Ground / Satellite Links
Maintenance Centers
UAV Networks
Data Integration
• Data Warehouses (Flight Plans, Safety Reports, Maintenance Reports, Naviagtional Charts, etc.)
• Data Streams (Radar / Sensor Data, Video / Audio Streams, etc.)
• Data Transformation (Source Merging, Data Converting, Unifying of Structured / Unstructured Data, Schema Mapping, etc.)
Data Managment
• Data Provision (Clusters at Mulitiple Data Centers, Data Warehouses, Data Streams)
• Data Extraction
• Data Processing Frameworks (Hadoop, Apache Spark, Apache Storm, Flink, etc.)
Data Preprocessing
• Data Cleaning (Missing Values, Noisy Data)
• Data Normalization, Aggregation, Generalization
• Data Reduction (Number of Attributes, Number of Values)
• Feature Selection
• Discretization (Unsupervised and Supervised Discretization)
Data Mining (Predictive & Descriptive)
• Classification (SVN, Decision Tree, KNN, Bayesian Network)
• Clustering (Partitioning Clustering, Density Clustering, etc.)
• Outlier Analysis (Probabilistic and Statistical Models, etc.)
• Time Series Analysis (Segmentation, TS Patterns, etc.)
• Data Visualization
Knowledge Presentation
• Real-Time Warnings and Alerts to Pilot or Aircraft Dispatcher
• Air Traffic Services (Real-Time Aircrafts Positions)
• BI Reports (Maintenance-, Fleet-, Customers-Reports, etc.)
• Knowledge Ontologies of Aircraft
Real-Time Alerts for Pilots and Aircraft Dispatchers
Air Traffic Management
Maintenance Report Overview of technical issues e.g. Turbine Engine failure
DS-1106
Interactive Flight Schedule
Visualization of complex data for Data Scientist

Fig. 5.1 Overview of Big Data analytics

Program, in which data sets are sometimes not accessible, heterogeneous regarding their structure and schema or only available through certain protocols [20]. Therefore, some problems, especially combining various database schemas or handling of different representations of data, aroused from the data sources themselves, have to be solved, so that the relevant sources can be transformed and merged.

At first the schema of the different data sources have to be unified, so that uneven source schema will be converted through a function identifying sets of attributes, which contain the same informations, and create a global schema for all sources

[21–23]. Building on the results, the sources get merged through record linkage techniques to handle data sets, which does not share a common identifier, but have the same schema, so that data sets referring to the same logical entity of various sources can be integrated, e.g., flight schedules from different airports, weather data from different sources, or a combination of sensors inside an aircraft. Since problems in the topic of Big Data often have data types of different nature, completely unstructured, or a highly heterogeneous structure traditional record linkage approaches are inefficient, because of the classification of Big Data sets in general (volume, variety and velocity) and its manifestations in aviation. To cope with these issues, [24] presents an overview of Big Data integration. Regarding the volume problem (of Big Data), parallel computing through programming models, e.g., MapReduce, is suggested for batch processing. The velocity aspect of Big Data can be managed through incremental clustering techniques, because the fast-changing data can make previous linkage results obsolete and computing power is wasted. Gruenheid et al. [25] present some incremental linkage algorithms, which can deal significantly better with the fast velocity and have a negligible loss in quality. The algorithms are not only able to merge updates in already created linkage but can also detect errors in the generated structure and repair them. The variety of heterogeneous and unstructured data, for example, maintenance orders or instructions in a free text format, is a big challenge to integrate them into some structure. New linkage techniques tagging and matching elements into an already existing structure are proposed as a solution to this aspect, but are highly dependent on the basis of analytics to design. Finally, the veracity and the variability property can be reduced through a combination of clustering and linkage techniques. Eventually, data fusion will be applied, which refers to the problem of combining sources defining the same things with contradicting facts. The essential need of data fusion evolved due to various false facts spread from Internet sources. In aviation, these problems occur relatively less, since there are more reliable sources to interpret, where it is more important regarding truthfulness to check if sensors are faulty. This topic will be treated more in detail in the section of Data Preprocessing.

5.4.2 Data Management

It is necessary to maintain a software environment for data management, which is able to handle all aspects and problems of Big Data, especially analyzing and processing huge amounts of heterogeneously data in short periods. The data management of Big Data enables the provision of relevant data sets to perform data analysis and includes infrastructure for fast propagation of data streams. To store and process these huge amounts of data the use of data-warehouse-solutions are often inefficient, since data warehouses are designed to deal with structured data. A well-working structure of storage, which can also cope with fast response times, is possible through clusters at multiple data centers. This system can handle the volume of Big Data better than traditional databases and enables to handle fast data streams. NoSQL databases store and retrieve flexible, complex, distributed data. The horizontal scalability provides a fast processing of large data, and therefore they are increasingly utilized in Big Data [26].

Stored data can be efficiently processed by data processing frameworks, e.g., Hadooop, Apache Storm, Apache Spark, etc., which can be categorized by their processing purposes into batch processing, stream-only, and hybrid frameworks. Batch processing is used to process large, non-dynamic data sets. Hadoop is a prominent framework, which provides batch processing on a distributed file system and implements a MapReduce programming model. Batch processing handles large data sets, where the time is not an important factor. Stream processing systems handle sequences of data and process each or few data items at the same time, for example, Apache Strom provides functionalities for fast distributed data stream processing. Hybrid paradigm includes batch and stream processing simultaneously, and it is a hybrid solution for general data processing. Apache Spark is a lightweight batch data processing framework with ability to process data streams. It can be combined with Hadoop and access different data sources, e.g., Cassandra [27–30].

5.4.3 Data Preprocessing

Faulty, incomplete and redundant information has a negative effect on the quality of real-word data sets. The poor data directly affects the quality of data mining performance. Data preprocessing deals with missing values, outliers, and noisy data to clean raw data. Data normalization and aggregation convert the data into suitable formats for data mining. Data reduction removes irrelevant and redundant attributes and attribute values.

For example, distinguishing between safety anomalies and wrong data is extremely important in aviation context. On the one hand, the noisy data can produce misleading safety warnings and, on the other hand, wrong filtered system failures can cause an accident in the worst case [31, 32]. Correct detection of system failures with sensor-data-driven techniques is a major challenge. Sensors can produce signal drifts caused by imprecise measuring, so that the using of redundant sensors and redundant analytics can help to detect sensor faults and to minimize noisy signals. The case study in [31] demonstrates a fault detection of the high-pressure compressor in an aircraft engine. The generated training data sets under different engine health and varying sensor noise conditions are assigned to different classes. The supervised methods help distinguish between sensor noise, sensor fault, and engine fault.

Further, part of data preprocessing is a reduction of streamed and/or collected data. The data intensity and domain complexity in aviation grows continuously, millions of samples are produced by sensors, cameras, actuators, network connectivity, and further services. The reduction of the data is mostly necessary to execute complex analysis in acceptable time. The idea is to produce the same analytical results as it would have been with reduced data [33].

5.4.4 Data Mining

Data mining discovers useful patterns, associations, and outliers, from large dynamic data sets to extract meaningful information. The future forecasts can be made by using prediction models. For example, the data mining techniques can help to identify individual reasons or combination of reasons for the aviation incidents and accidents. Reference [34] divides the data mining techniques into the following areas: classification (supervised learning), clustering, association analysis, time series analysis, and outlier analysis. The classification process assigns the data into predefined classes. Reference [35] uses classification techniques to predict warning level based on aviation data streams. Many classification techniques are mentioned in the literature, e.g., Support Vector Machine (SVM), Decision Tree, KNN, and Bayesian Network [34]. Reference [36] uses decision trees to predict failures and to identify cumulative effects such as mechanical vibrations, environmental temperature, humidity, power supply, air pressure, functional overload, etc., in combination, which are responsible for a failure. The decision tree construction process selects most informative pair of factors, e.g., environmental temperature and power supply. Automatically generated linear separation rule, from the training data set, divides the space into subsets. The selection of further factors is applied separately to determine subsets. The leaves represent classes and the branches decision rules. Every data input is classified or generalized by decision procedure. Reference [37] uses Bayesian networks to predict failure of aircraft tires. Predictions of tire conditions increase the aircraft safety and make the maintenance process more efficient. A concept of Dynamic Data-Driven Avionics Systems (DDDAS) has been proposed in [38] to analyze data streams from aircraft sensors and instruments in real time to enrich computational models for predicting aircraft performance more effectively and [39] presents a multimodal data error detection and recovery architecture. It is able to detect data errors in streaming applications to enable automated recovery, which could compensate incorrect data from sensor through the available redundancy of the data streams.

Clustering algorithms divide data set into different groups by predefined similarity criteria. Different clustering paradigms are mentioned in the literature [40], e.g., Partitioning Clustering, Hierarchical Clustering, Density-based Clustering, Grid-based Clustering, Spectral clustering, and further. Reference [41] groups similar aircraft trajectories and discovers common trajectories. Initially, the algorithm partitions the trajectories into line segments. The density-based clustering algorithm groups the corresponding line segments with small distance into cluster. Finally, the algorithm generates a common trajectory for each cluster. Reference [42] solves the similar problem based on the partitioning clustering method k-Means, which clusters the multidimensional trajectory points. Centroids of clusters are then used to construct a common trajectory.

5.4.5 *Data Visualization*

The new data mining technologies derivate more precise knowledge from large data sets, partly in real time or near real time. Representation of produced data and knowledge in the right context is very difficult task. Business Intelligence (BI) reporting is a widely used technique to analyze and represents historical, current, or predictive data. Reference [43] presents a BI report based on predictive maintenance system. The report summarizes predictive technical statuses of aircraft fleets. The interactive dashboards include aircraft positions, flight schedule, and a detailed list of technical issues. Well-defined filters allow to select relevant information according to the aircraft engineers' requirements. The provided information can help to decide about service location, maintenance process, or aircraft replacement.

5.5 Conclusion

The topic of Big Data is more and more relevant today in the aviation sector. It has a great impact in every economic field and is expected to get even more relevant in the future. Novel Big Data technologies make it possible to meet the increasing requirements for, e.g., UAV logistic challenges, optimization of aircraft routing, general aviation safety, etc. The aviation industry benefits from the utilization of Big Data methods, the process includes Data Integration, Data Management, Data Preprocessing, Data Mining, and Knowledge Presentation. This chapter provides an overview and summarizes the literature about Big Data in context of aviation with elaborating the relevant examples.

References

1. The data science revolution that is transforming aviation (2018), https://www.forbes.com/sites/oliverwyman/2017/06/16/thedata-science-revolution-transforming-aviation/. Accessed 12 Feb 2018
2. A.Y. Zomaya, S. Sakr, *Handbook of Big Data Technologies* (Springer, Berlin, 2017)
3. P. Russom et al., Big data analytics, in *TDWI Best Practices Report, Fourth Quarter*, vol. 19(4) (2011), pp. 1–34
4. H.-M. Chen, R. Schuetz, R. Kazman, F. Matthes, How Lufthansa capitalized on big data for business model renovation. MIS Q. Exec. **16**(1) (2017)
5. L.R. Poole, A. Catalano, Real time visualization of sensor data in aircraft, in *AUTOTESTCON 2004. Proceedings* (IEEE, New York, 2004), pp. 389–394
6. A very short history of big data (2018), https://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-bigdata/. Accessed 12 Feb 2018
7. Big data the next frontier for innovation (2018), https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation. Accessed 12 Feb 2018
8. T. Davenport, *Big Data at Work: Dispelling the Myths, Uncovering the Opportunities* (Harvard Business Review Press, 2014)

9. R. Bryant, R.H. Katz, E.D. Lazowska, *Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society* (2008)
10. S. Yin, O. Kaynak, Big data for modern industry: challenges and trends [point of view]. Proc. IEEE **103**(2), 143–146 (2015)
11. Die 9 V von Big Data (2018), https://digitales-wirtschaftswunder.de/die-9-v-von-big-data/. Accessed 12 Feb 2018
12. Four Vs Big Data (2018), https://www.ibmbigdatahub.com/infographic/four-vs-big-data. Accessed 12 Feb 2018
13. Updated for 2017: The V's of big data: velocity, volume, value, variety, and veracity (2017), https://www.xsnet.com/blog/updated-for-2017-the-vs-of-big-data-velocity-volume-value-varietyand-veracity. Accessed 12 Feb 2018
14. Big data in planes: new P and W GTF engine telemetry to generate 10GB/s (2018), https://www.vrworld.com/2015/05/08/big-data-in-planes-newpw-gtf-engine-telemetry-to-generate-10gbs/. Accessed 12 Feb 2018
15. Uber Elevate (2018), https://www.uber.com/info/elevate/. Accessed 12 Feb 2018
16. D. Steinmetz, G. Burmester, S. Hartmann, A fast heuristic for finding near-optimal groups for vehicle platooning in road networks, in *Proceedings of the International Conference on Database and Expert Systems Applications* (Springer, 2017), pp. 395–405
17. M. Simons, *Model aircraft aerodynamics* (Chris Lloyd Sales & Marketing, 2000)
18. S. Li, Y. Yang, L. Yang, H. Su, G. Zhang, J. Wang, Civil aircraft big data platform, in *IEEE 11th International Conference on Semantic Computing (ICSC), 2017* (IEEE, New York, 2017), pp. 328–333
19. W. Miao, D. Zheng, G. Hangyu, Y. Tao, Research on big data management and analysis method of multi-platform avionics system, in *IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)* (2017), pp. 757–761. https://doi.org/10.1109/ICIS.2017.7960094
20. D. Kulkarni, Y. Wang, M. Windrem, H. Patel, R. Keller, *Aviation Data Integration System* (2003)
21. S. Aulbach, T. Grust, D. Jacobs, A. Kemper, J. Rittinger, Multi-tenant databases for software as a service: schema-mapping techniques, in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (ACM, 2008), pp. 1195–1206
22. C. Batini, M. Lenzerini, S.B. Navathe, A comparative analysis of methodologies for database schema integration. ACM Comput. Surv. (CSUR) **18**(4), 323–364 (1986)
23. C. Esposito, M. Ficco, F. Palmieri, A. Castiglione, A knowledge-based platform for big data analytics based on publish/subscribe services and stream processing. Knowl. Based Syst. **79**, 3–17 (2015)
24. X.L. Dong, D. Srivastava, Big data integration, in *2013 IEEE 29th International Conference on Data Engineering (ICDE)* (IEEE, New York, 2013), pp. 1245–1248
25. A. Gruenheid, X.L. Dong, D. Srivastava, Incremental record linkage. Proc. VLDB Endow. **7**(9), 697–708 (2014)
26. A. Moniruzzaman, S.A. Hossain, Nosql database: new era of databases for big data analytics-classification, characteristics and comparison, in *arXiv preprint* (2013), arXiv:1307.0191
27. A. Dhar, U. Student, Big data technologies for batch and real-time data processing: A. Int. J. Eng. Sci. **15232** (2017)
28. J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
29. S. Salloum, R. Dautov, X. Chen, P.X. Peng, J.Z. Huang, Big data analytics on apache spark. Int. J. Data Sci. Anal. **1**(3), 145–164 (2016). https://doi.org/10.1007/s41060-016-0027-9.ISSN:2364-4168
30. N. H. Motlagh, T. Taleb, O. Arouk, Low-altitude unmanned aerial vehicles-based internet of things services: comprehensive survey and future perspectives. IEEE Intern. Things J. **3**(6), 899–922 (2016). https://doi.org/10.1109/JIOT.2016.2612119.ISSN:2327-4662
31. S. Sarkar, X. Jin, A. Ray, Data-driven fault detection in aircraft engines with noisy sensor measurements. J. Eng. Gas Turbin. Power **133**(8), 081602 (2011)

32. E.C. Larson, B.E. Parker, B.R. Clark, Model-based sensor and actuator fault detection and isolation, in *Proceedings of the 2002. American Control Conference, 2002*, vol. 5 (IEEE, New York, 2002), pp. 4215–4219
33. S. García, J. Luengo, F. Herrera, *Data Preprocessing in Data Mining* (Springer, Berlin, 2015)
34. F. Chen, P. Deng, J. Wan, D. Zhang, A.V. Vasilakos, X. Rong, Data mining for the internet of things: literature review and challenges. Int. J. Distrib. Sens. Netw. **11**(8), 431047 (2015)
35. A. A. Christopher, S. A. alias Balamurugan, Prediction of warning level in aircraft accidents using data mining techniques. Aeronautical J. **118**(1206), 935–952 (2014), pp. 935–952
36. V.A. Skormin, V.I. Gorodetski, L.J. Popyack, Data mining technology for failure prognostic of avionics. IEEE Trans. Aerosp. Electron. Syst. **38**(2), 388–403 (2002)
37. I.X. Castilho, *Fault prediction in aircraft tires using Bayesian Networks* (2015)
38. S. Imai, A. Galli, C.A. Varela, Dynamic data-driven avionics systems: inferring failure modes from data streams. Proc. Comput. Sci. **51**(Supplement C) (2015), pp. 1665–1674. https://doi.org/10.1016/j.procs.2015.05.301. ISSN: 1877-0509
39. R. Klockowski, S. Imai, C.L. Rice, C.A. Varela, Autonomous data error detection and recovery in streaming applications. Proc. Comput. Sci. **18**, 2036–2045 (2013)
40. K.-C. Wong, A short survey on data clustering algorithms, in *2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI)* (IEEE, New York, 2015), pp. 64–68
41. J.-G. Lee, J. Han, K.-Y. Whang, Trajectory clustering: a partition-andgroup framework, in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data* (ACM, New Jersey, 2007), pp. 593–604
42. S. Ayhan, H. Samet, Diclerge: Divide-cluster-merge framework for clustering aircraft trajectories, in *Proceedings of the 8th ACM SIGSPATIAL IWCTS* (Seattle, WA, 2015)
43. Predictive Maintenance System - PowerBI (2018), https://powerbi.microsoft.com/en-us/industries/airline/. Accessed 12 Feb 2018

Chapter 6
Ontologies in Aeronautics

Carlos C. Insaurralde and Erik Blasch

Abstract Avionics systems are getting increasingly sophisticated, airspaces are densely occupied, and aircraft are desired to fly in more adverse weather conditions. These conditions increase the complexity of Air Traffic Management (ATM) as aviators and airspace controllers struggle to maintain safety while cross-checking multi-source information, including information from Unmanned Aerial Systems (UASs). Hence, future ATM decision-support systems are required not only to be autonomous and reliable complex decision-making processes with minimal human intervention, but also must be able to deal with UAS ATM (UTM). This chapter presents the implementation of Ontologies for NextGen Avionics Systems (ONAS) for UTM. The ONAS approach consists of an operation framework and an ontology-based tool, called Avionics Analytics Ontology (AAO), to support decision-making in advanced ATM/UTM systems. The AAO entails a cognitive ATM/UTM architecture for avionics analytics where an ontological database captures information related to weather, flights, and airspace. The AAO-based decision-making process supports human Situation AWareness (SAW) as well as machine Situation Assessment (SA). The ONAS approach presented is intended to be initially used in civil aviation. A use case along with two different scenarios is presented for an ATM/UTM system. The scenarios represent realistic flight situations (based on dataset from a flight tracking service) where the ATM/UTM decisions made are supported by the AAO.

6.1 Introduction

Avionics analytics in aerospace is getting rapidly sophisticated with such improvements in Command, Control, Communication, Computers and Intelligence (C4I). One of the main purposes of C4I systems is to support Communication,

C. C. Insaurralde (✉)
Teesside University, Middlesbrough, UK
e-mail: c.c.insaurralde@tees.ac.uk

E. Blasch
US Air Force Research Lab, Rome, NY, USA
e-mail: erik.blasch.1@us.af.mil

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_6

Navigation, and Surveillance (CNS) applications such as Air Traffic Management (ATM). Decisions in ATM are the result of the combination of a large number of factors (e.g., weather forecasts, flight profiles, airports, Unmanned Aerial Vehicles (UAVs), etc.). The information fusion complexity is even more increased when considering airspace users from different aviation sectors such as scientific, recreational, commercial, civil, and military aviation.

ATM situation demands a huge workload on aircraft pilots and Air Traffic Controllers (ATC) who must prioritize flight trajectories, safety, and messaging while cross-checking information coming from the different sources, including Unmanned Aerial Systems (UASs). The coordination of UAS and different stakeholders with ATM, aka Unmanned Aerial System Traffic Management (UTM), further challenges aerospace information management systems by requiring aerospace information management systems to be efficient with larger amounts of data. The cooperative effort between air traffic controllers and management systems creates a need for decision-making support for ATM/UTM to deal with the combination of information from different sources (i.e., weather forecasts, flight profiles, airports, UAVs, etc.).

A key avionics design challenge for future ATM Decision-Support Systems (DSS) is autonomous, agile, and reliable complex decision-making with minimal human intervention [1]. For example, automation is required to combine dynamic multiple data inputs within an Integrated Modular Avionics (IMA) [2]. Agility requires the ability to adapt to change. Autonomy is also afforded from machine analysis of route changes, airspace de-confliction, Performance Based Navigation (PBN) [3], and power assessment [4]. Hence, the Federal Aviation Association (FAA) Next Generation (NextGen) [5] and Single European Sky ATM Research (SESAR) [6] ATM systems require a certain degree of autonomy as well as a man–machine collaborative operation mode to minimize the need for aviators and controllers intervention.

This chapter presents the implementation of Ontologies for NextGen Avionics Systems (ONAS). It discusses an operation framework and an ontology-based process to support decision-making in advanced ATM/UTM systems called Avionics Analytics Ontology (AAO). The ONAS approach includes a cognitive ATM/UTM architecture for avionics analytics (the AAO), Situation AWareness (SAW), and Situation Assessment (SA). It is intended to be used in civil aviation; however, it could potentially have a future use (on-board) in autonomous UAVs (to increase autonomy) and even be considered in ATM operations [7]. This chapter shows a use case for the ONAS proposed that includes two scenarios for an ATM/UTM system. The scenarios consider semantics from updates of weather profiles, airport maps, and route plans. The scenarios are meant to represent realistic flight situations since they make use of real-time airspace Automatic Dependent Surveillance-Broadcast (ADS-B) information provided by Flightradar24 [8] (a flight tracking service), and describe flight situations where the decisions made are supported by the proposed AAO approach. The experimental results present a Dynamic Data Driven Application Systems (DDDAS) [9] example that leverages contextual models to support real-time performance updates for system control.

The rest of the chapter is organized as follows. Section 6.2 discusses the role of ontologies in ATM along with a review of existing technologies. Section 6.3 introduces foundations and backgrounds for ontological decision-making support in avionics with a focus on ONASs. Section 6.4 describes the AAO as the main pillar for the ontological decision-making support defined by the ONAS approach. Section 6.5 presents a use case along with two scenarios as application examples of the AAO. The last section presents the concluding remarks and the way forward for the AAO.

6.2 Ontologies in Air Traffic Management

There is an increasingly emergent interest in the use of ontologies for ATM and aerospace technologies [10, 11]. For instance, the FAA NextGen [5] and the Single European Sky ATM Research (SESAR) [6] systems are planning to incorporate ontologies for named events. Figure 6.1 highlights an example of how ontologies are included in an avionics system analysis. Using the incoming data from weather, flight profiles, and airports, that data needs to be accessed, structured, and normalized over related conditions. Structuring the data is enabled with templates and ontologies. The structured ontology organizes the information (including syntactic and semantic metadata) for analytic processing and reporting. The resulting analytics supports visualization and decision-making for aviators and air traffic controllers. Examples

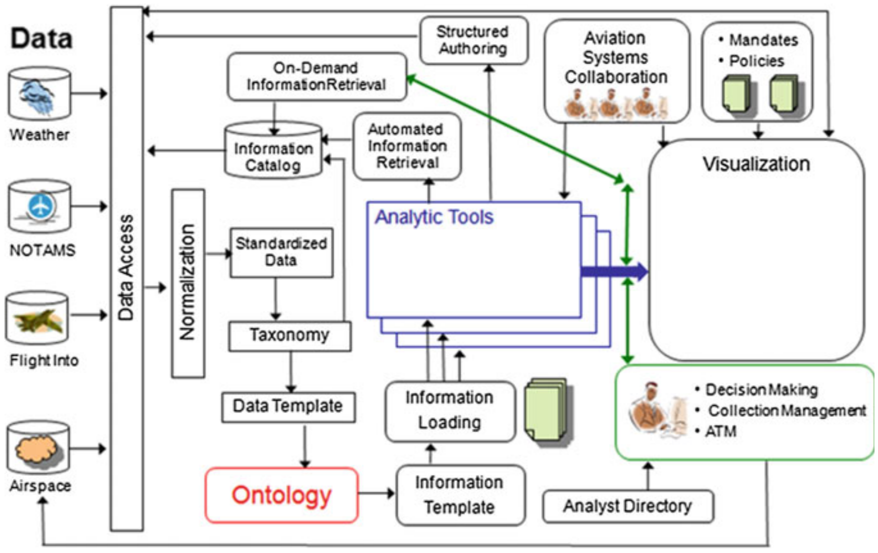


Fig. 6.1 Ontologies for avionics analytics

include mandates, current reports, and airspace information. Hence, ontologies afford a common method to organize, process, and share data.

System Wide Information Management (SWIM) including the ATM Information Reference Model (AIRM) [12], the Information Service Reference Model (ISRM), and the SWIM Technical Infrastructure (SWIM-TI) are being developed for air traffic management [13]. The concept of SWIM is an emerging concept to manage information for aviation systems for various ATM networks [14]. An AIRM example requiring ontologies is semantic filtering of NOTices to AirMen (NOTAM) [15].

Information fusion systems rely on the data that is acquired, processed, and utilized. Three steps are important in the analysis which includes data normalization, standardization, and templating for situation analysis:

- **Data Normalization**. A fundamental knowledge of all fusion designs is that data is related to the collection as it is most likely processed as a probability (e.g., Bayes theorem). To construct a data collection into a format for analysis is the process of normalization. For instance, the absolute value of 10,000 m is relative to the environment. If a UAV was to be separated by the 10 Km limit of aircraft, it would be conservative for close-air operations. Hence, the safe distance threshold should be normalized to the size of the aircraft.
- **Standardized Data**. Data standardization includes the units, terminology, and definitions associated with the information. For the UAV airspace example, the information should be standardized in time (zulu), distance (meters), and direction. An example of an ontology is Above Ground Level (AGL).
- **Data Template**. To determine a situation, a third step is useful (although there could be others), such that the template of knowledge that is available. A template could include the geographical terrain, cognitive processes, or action states. The template enables a rapid understanding of the situation from normal operations.

Key developments for Single European Sky ATM Research (SESAR) and the Federal Aviation Administration (FAA) Next Generation Air Transportation System (NextGen) include the potential for ontological capabilities. Rainer Koelle and Walter Strijland [16] outlined progress in 2013 for semantic assurance for systems engineering in SESAR/NextGen which provides ATM security. A key example for NextGen system includes developments in the weather ontology, implemented in three operational capability phases [17]:

- **Initial (2013)**: Significantly enhanced weather infrastructure providing modestly improved meteorological data to all users of the NextGen air transportation system.
- **Midterm (2016)**: NextGen begins to implement automated decision assistance tools and algorithms for managing the airspace, requiring high-resolution weather forecasts and observations with more accuracy and precision.
- **Farterm (2022)**: NextGen weather must meet all meteorological and engineering performance requirements to support the NextGen traffic management systems.

Another example is the European AIRM. Specifically, they looked at NOTices to AirMen (NOTAM). NOTAMs provide weather and emergency updates to aviators in the form of text messages. Using Ontology Web Language (OWL), the system seeks a semantic-based aeronautical information management system. With semantic reasoning and digital NOTAMs, efforts were underway to bring structure to the knowledge gained from text-based information.

6.3 Information Fusion

This section introduces information fusion concepts as foundations and backgrounds for ontological decision-making support in avionics with a focus on ONASs.

6.3.1 Situation Awareness

The decision-making process is based on the four-stage loop called Observe-Orient-Decision-Act (OODA) [18]. The OODA loop is essential for situation awareness in information fusion [19]. Figure 6.2 shows a SAW model.

SAW allows systems to understand dynamic and complex environments, and operate with them. Cognitive SAW can be divided into three separate levels: perception of the elements in the environment, comprehension of the current situation, and projection of future status [19].

The concepts of the OODA loop enable a processing of information. The *Observation* stage is the SAW perception level. The *Orientation* stage considers the information acquired from the Observation stage and the knowledge represented by the ontology, to understand the situation (SAW comprehension level). The *Decision*

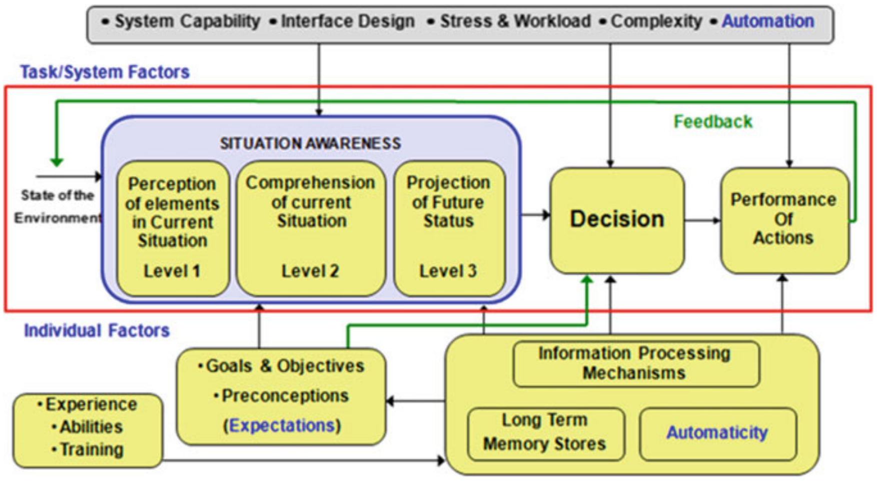

• System Capability • Interface Design • Stress & Workload • Complexity • Automation
Task/System Factors
Feedback
SITUATION AWARENESS
State of the Environment
Perception of elements in Current Situation Level 1
Comprehension of current Situation Level 2
Projection of Future Status Level 3
Decision
Performance Of Actions
Individual Factors
• Goals & Objectives • Preconceptions (Expectations)
• Experience • Abilities • Training
Information Processing Mechanisms
Long Term Memory Stores
Automaticity

Fig. 6.2 Situation AWareness (SAW) model

stage is carried out at the SAW projection level. The *Action* stage closes the OODA loop by carrying out plans according to the adaption made in the previous stage.

SAW involves the events, states, conditions, and activities of the environment dynamics over time and space from which some situations arise (in particular, those changes that occurred in the environment over some time interval). A *situation* is defined by a specific state after a sequence of events (with intermediate states, and activities with pre/post-conditions). The situation is concerned with the comprehension of the environment features, and with the evolvement of these features over time.

SAW decision-making mechanisms are critical for problem-solving processes that are preformed every time step for a situation from which data is collected and processed.

6.3.2 *Situation Assessment*

Situation assessment takes place at level 2 (SAW comprehension) in data fusion models. The Data Fusion Information Group Model (DFIG) [20, 21] levels are listed below and shown in Fig. 6.3.

- **Level 0**: Data Assessment
- **Level 1**: Object Assessment
- **Level 2**: Situation Assessment
- **Level 3**: Impact Assessment
- **Level 4**: Process Refinement / Resource Management

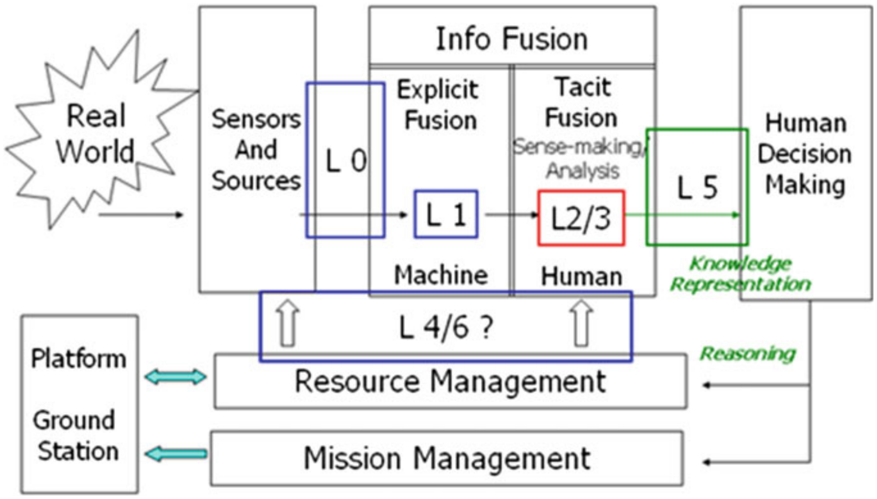


Fig. 6.3 Data fusion information group (DFIG) model

- **Level 5**: User Refinement / Cognitive Management
- **Level 6**: Action Refinement/ Mission Management

In the DFIG model, the goal was to separate the Information Fusion (IF) (L0-L3) and Information Management functions (L4-L6) [22, 23].

For UTM systems, there is both the resource management across Sensors, Users, and the Mission (SUM) to coordinate with the objects, situations, and threats. The elements of the airspace need to be provided to air traffic controllers for enhanced SAW. Two integral concepts for Level 5 User Refinement information Fusion are displays to support usability [24], and information management systems that are trustworthy [25]. A binding element between the levels of information fusion is an ontology to reduce uncertainty [26, 27].

6.4 Ontological Decision-Making Support

This section describes the AAO which is the main pillar of the ontological decision-making support defined by the ONAS approach proposed.

6.4.1 Cognitive Model

The cognitive model represents the knowledge required to endow the avionics decision-making support with intelligence. It relies on data, information, and knowledge. However, they are not definitely the same but related. Data are usually raw or preprocessed. Information is data with meaning. Information becomes knowledge when there is a purpose and a potential to generate action. Knowledge is the intellectual machinery which makes possible to achieve goals (by carrying out actions) and creates new information [28].

Knowledge can formally be represented by means of *Description Logic* (DL). The DL architecture has two main components: TBox and ABox [29]. TBox entails inclusion assertions about properties from concepts and roles. Abox entails instance assertions such as those for individual objects [30]. The method includes the following:

- **TBox** component is a *terminological* formalism (terminology; system description in terms of controlled vocabularies); whereas
- **ABox** component is an *assertional* formalism (assertions about individuals).

The combination of the TBox and the ABox forms the knowledge base which is described through ontologies. The main ontology elements are concepts (classes), properties, instances (individuals), and assertions. A *concept* represents a set of entities or things within a domain. *Properties* define either relations between an individual and a value, or between two individuals, called data type properties and object properties, respectively.

The TBox module defines the concepts and properties in a domain in addition to specifying terminological axioms for every atomic concept. Axioms are used to constrain the range, and domain of the concepts, e.g., an airplane is an aerospace vehicle that has navigation capabilities. *Assertions* are statements about facts or beliefs. The ABox module contains a finite set of assertions for the classification of individuals and their properties.

Inference over the ontology (TBox and ABox) is provided by a *reasoner*. The knowledge representation approach proposed in this chapter is based on ontology and reasoner as described above. An ontological database captures information (data along with meaning) as to concepts, entities, and relations in order to build knowledge related to weather, flights, and airspace. The ontology enables Artificial Intelligence (AI) reasoning to make decisions based on the knowledge stored and the current situation estimates.

6.4.2 Semantic Ontology

The ontology syntax (symbols and rules) is based on the DL syntax structure. However, the implementation language for the ontology ultimately defines the syntax to semantically specify and describe ontology elements. The OWL and the Protégé tool [31] are selected to realize the ontology for the approach proposed.

The main OWL components to be created are the concepts (classes), properties for individuals, and instances of classes (individuals). These components are set for Avionics Analytics Ontology (AAO) as follows:

- **Classes (concepts)** are conceptually defined as classes (special datatype) in object-oriented programming languages. Thus, they can be atomic classes (stand-alone ones) or associate classes (subclasses) along with is-a links. The main AAO classes are vehicle (aircraft), radar, criteria, pilot, route, airport, runway, status, airspace, weather, and metrics. Figure 6.4 shows the above is-a relations between classes.
- **Properties (roles)** are basically relationships between classes (or eventually individuals). The OWL allows for properties on objects (based on classes) or data (specific values). The first version of AAO only includes properties for objects as follows: `hasRadar`, `hasPilot`, `hasRoute`, `hasTakeoff`, `hasLanding`, `hasAirspace`, `hasRunnay`, `hasStatus`, `hasVeracity`, and `hasWeather`.
- **Individuals (instances)** are instances of classes (objects), e.g., a Boeing 747-800 is an individual (instance of the class aircraft).

DL operators are considered as different types of property restrictions in ontologies: (1) quantifier restrictions such as existential and universal restrictions, (2) has value restrictions (counting operators such as less than or equal to and more than or equal to), as well as (3) cardinality restrictions such as minimum and maximum cardinality restrictions. Also, complex classes can be created by means of simpler classes described based on logical operators like OR and AND.

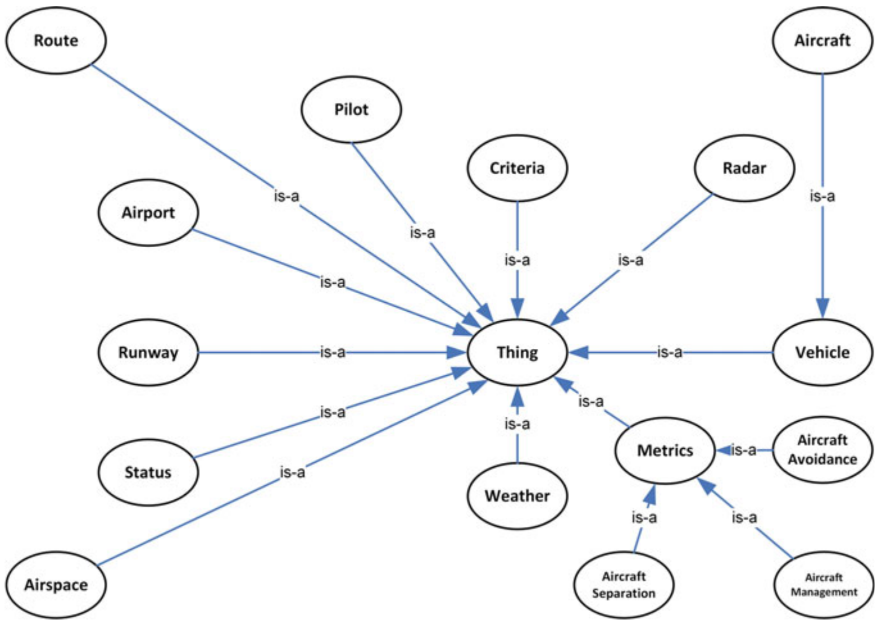


Fig. 6.4 Main OWL components of the AAO

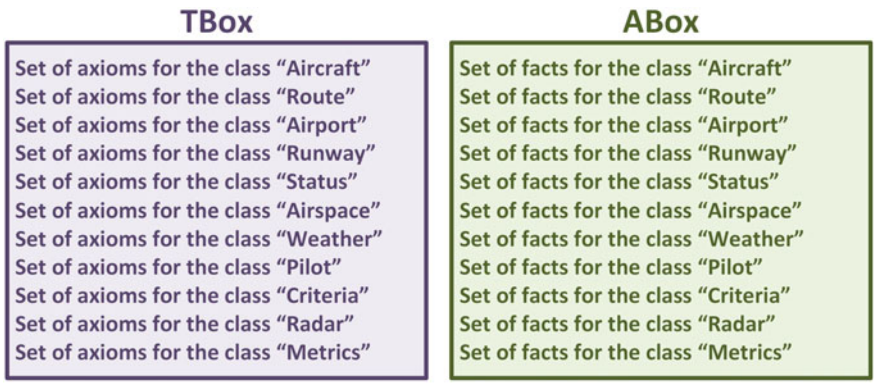
Property restrictions along with classes and individuals are the building block to define axioms. Terminological axioms (usually based on operators such as inclusion, equivalence, etc.) are in the TBox, e.g., `Aircraft_A` *subclass of* `AircraftcannotLand` *and* `AircraftcanTakeoff`, and `ClearSky` *subclass of* `GoodWeather` *and* `VeryGoodWeather`. A set of assertional axioms (facts or assertions) are in ABox, e.g., `AircraftcanLand` *equivalent to* `Aircraft` *and* (`hasRoute` *only* `Landing`), and `VeryGoodWeather` *equivalent to* `Weather` *and* (`ClearSky` *or* `CloudedSky`).

The ABox and the TBox form the AAO knowledge base and are shown in Fig. 6.5. Details of the TBox and ABox axioms are available in [32].

6.4.3 Ontological Reasoning Process

Reasoners are the engine for the knowledge-based queries. Reasoners not only apply inference rules but also check semantic consistency on ontologies. These reasoning engines can deduce logical questions from axioms defined in ontologies. Figure 6.6 shows the asserted classes (considering properties and terminological and assertional axioms) of the AAO along with their is-a relations.

Figure 6.7 shows the inferred classes of the AAO as result of executing the reasoner. Figure 6.7 shows some examples of AAO inferences as follows (from top to


TBox

Set of axioms for the class "Aircraft"
Set of axioms for the class "Route"
Set of axioms for the class "Airport"
Set of axioms for the class "Runway"
Set of axioms for the class "Status"
Set of axioms for the class "Airspace"
Set of axioms for the class "Weather"
Set of axioms for the class "Pilot"
Set of axioms for the class "Criteria"
Set of axioms for the class "Radar"
Set of axioms for the class "Metrics"

ABox

Set of facts for the class "Aircraft"
Set of facts for the class "Route"
Set of facts for the class "Airport"
Set of facts for the class "Runway"
Set of facts for the class "Status"
Set of facts for the class "Airspace"
Set of facts for the class "Weather"
Set of facts for the class "Pilot"
Set of facts for the class "Criteria"
Set of facts for the class "Radar"
Set of facts for the class "Metrics"

Fig. 6.5 AAO knowledge base: TBox and Abox

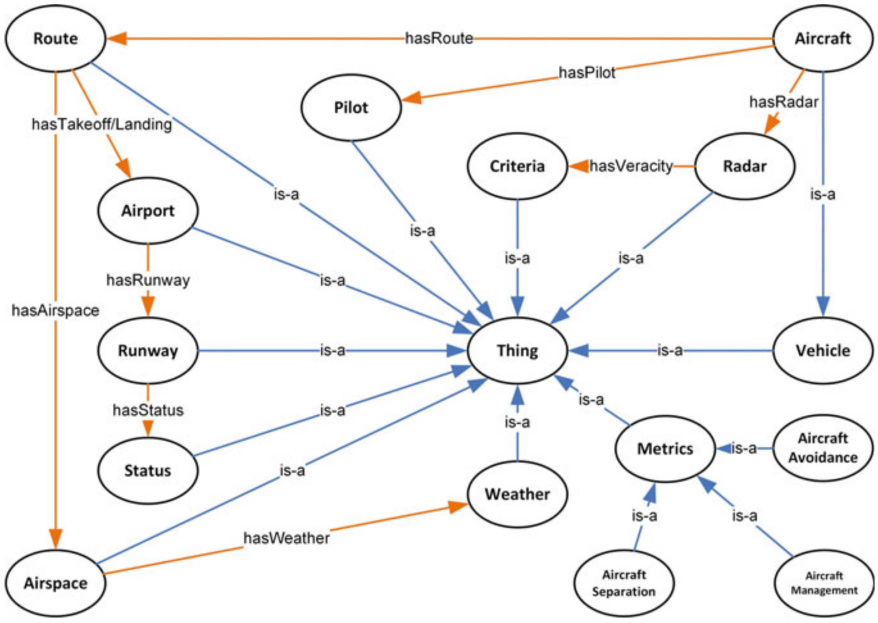
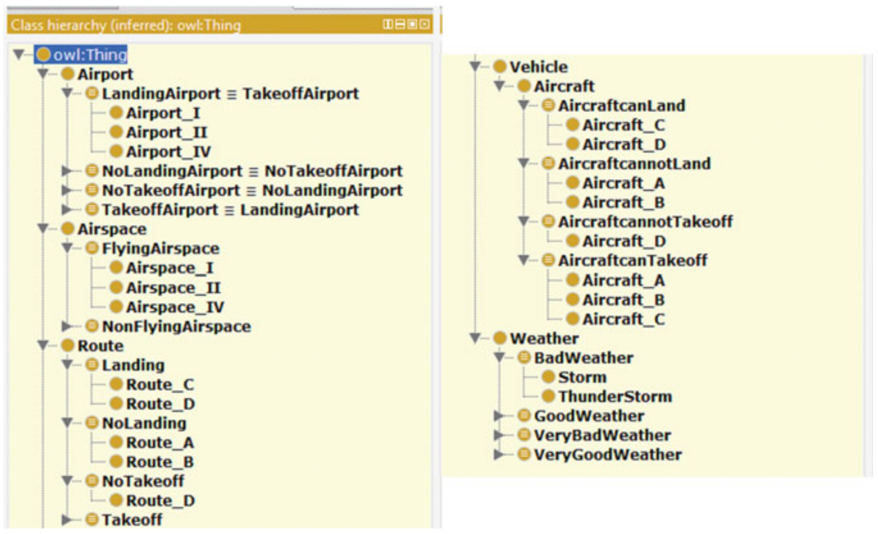


Fig. 6.6 Asserted AAO classes

bottom). Airport I, II, and III are take-off and landing airports (aircraft can take-off and land). Airspace I, II, and IV are flying airspace. Routes C and D have landing runways. However, Route D has no take-off runway. Aircraft C and D can land in their corresponding airports (e.g., Aircraft C with Route C). Bad weather includes storms and thunderstorms as supplied by NOTAMs.



```
Class hierarchy (inferred): owl:Thing
owl:Thing
    Airport
        LandingAirport ≡ TakeoffAirport
            Airport_I
            Airport_II
            Airport_IV
        NoLandingAirport ≡ NoTakeoffAirport
        NoTakeoffAirport ≡ NoLandingAirport
        TakeoffAirport ≡ LandingAirport
    Airspace
        FlyingAirspace
            Airspace_I
            Airspace_II
            Airspace_IV
        NonFlyingAirspace
    Route
        Landing
            Route_C
            Route_D
        NoLanding
            Route_A
            Route_B
        NoTakeoff
            Route_D
        Takeoff
    Vehicle
        Aircraft
            AircraftcanLand
                Aircraft_C
                Aircraft_D
            AircraftcannotLand
                Aircraft_A
                Aircraft_B
            AircraftcannotTakeoff
                Aircraft_D
            AircraftcanTakeoff
                Aircraft_A
                Aircraft_B
                Aircraft_C
    Weather
        BadWeather
            Storm
            ThunderStorm
        GoodWeather
        VeryBadWeather
        VeryGoodWeather
```

Fig. 6.7 Inferred AAO classes

6.5 Use Cases

This section presents application examples of the approach proposed in this chapter. They are based on realistic scenarios.

6.5.1 Airspace Situation

The case study is meant to be as realistic as possible. Thus, it involves a dataset from Flightradar24 [8]. The dataset records all flights of aircraft with ADS-B transponders. It has 390,607 records generated between 17:00 and 18:00 UTC on April 1, 2017 (approx. 109 records streamed per second) for flights all around the world. Nevertheless, there is about a revisit rate of 30 second on every aircraft.

The airspace area of interest is that from the US airspace, entailing arrivals from the east of Los Angeles International Airport (LAX), departures and arrival from the north of Dallas/Fort Worth International Airport (DWF), departures from Lake City International Airport (SLC), and departures from Tampa International Airport (TPA). The flights considered are Flight AA2250 (American Airlines from DFW to LAX), Flight AA2629 (American Airlines from SLC to DFW), and Flight AA2488 (American Airlines from TPA to DFW). The radars that detect the above flights (according to the ADS-B dataset used) are F-KLGB3, T-KFUL8, and F-KEMT2 for Flight AA2250. T-KNFW5, T-KDFW1, and F-KDAL1 for Flight AA2629, and

T-MLAT2, and T-F5M for Flight AA2488. They usually track aircraft depending on how far aircraft are from the radars and what destinations the flights have.

F-KEMT2 usually tracks aircraft approaching the US west coast (southwest locations) for LAX, T-KFUL8, and F-KLGB3 usually track aircraft approaching or departing from the east of LAX. T-KNFW5 often tracks aircraft coming from the northwest of DFW, and T-KDFW1 and F-KDAL1 usually track landed or arriving/departing flights in DFW. T-MLAT2 tracks aircraft leaving the west coast of Florida, and T-F5M tracks aircraft coming from the southeast of DFW.

6.5.2 Application Scenario 1: UAVs Not Present

Scenario 1 considers the above three flights, i.e., Flight AA2250 (American Airlines from DFW to LAX), Flight AA2629 (American Airlines from SLC to DFW), and Flight AA2488 (American Airlines from TPA to DFW), and the four airports mentioned above.

Flight AA2250 took off from DFW and is scheduled to land in LAX. Flight AA2629 took off from SLC and is scheduled to land in DFW. Flight AA2488 took off from TPA and is scheduled to land in DFW. It is assumed that weather conditions are very bad in the airspace en route to DFW for Flight AA2629 (from SLC). However, weather conditions are good for DFW and in the airspace en route to DFW for Flight AA2488 (from TPA) of DFW by the time both flights have scheduled their landing. The weather is very good for LAX.

LAX has four runways: runway 06L/24R, runway 06R/24L, runway 07L/25R, and runway 07R/25L. All of them are fully available (take-off and landing). DFW has seven runways: runway 13L/31R, runway 13R/31L, runway 17C/35C, runway 17L/35R, runway 17R/35L, runway 18L/36R, and runway 18R/36L. Two runways are fully available for take-off and landing, three runways are only available for take-off, one runway is only available for landing, and the remaining runway is not available at all. Runways at SLC are assumed to be operational just for take-off, and runways at TPA are assumed to be all fully operational. The status of the runways is not actually the ones from the date of the dataset since they are assumed to make scenario 1. Figure 6.8 shows scenario 1.

The information provided by the AAO can be visualized by air traffic controllers to support their decisions on the above situation (also, aviators and pilots of remotely piloted aircraft could make use of this information). They can run AAO queries as to the impact of weather conditions on the flights and availability of runways. The query results are suggestions about what to do or best to do. Figure 6.9 shows queries for the above inquiry and their responses from the AAO.

The query results suggest that (from left to right) (1) Flight AA2250 will be able to land as planned (actually, it is what really happened), (2) Flight AA2629 should be advised to change the original route in order to avoid the bad weather, and then land as expected in DWF, and (3) Flight AA2488 will not be able to land as it is usual

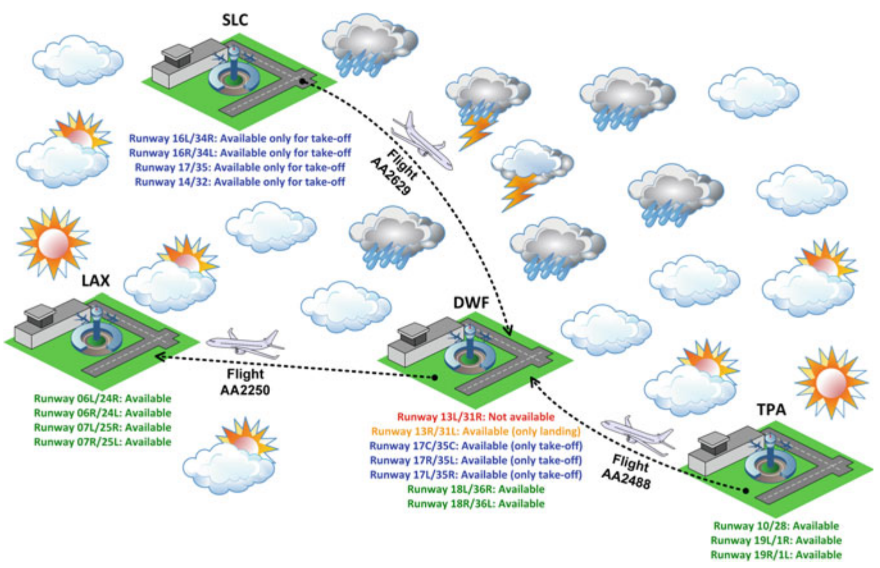
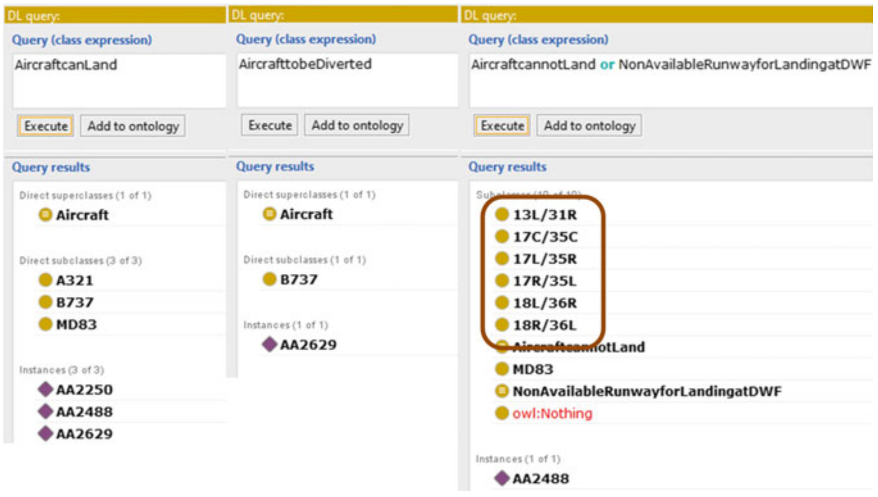

SLC
Runway 16L/34R: Available only for take-off
Runway 16R/34L: Available only for take-off
Runway 17/35: Available only for take-off
Runway 14/32: Available only for take-off
Flight AA2629
LAX
Runway 06L/24R: Available
Runway 06R/24L: Available
Runway 07L/25R: Available
Runway 07R/25L: Available
Flight AA2250
DWF
Runway 13L/31R: Not available
Runway 13R/31L: Available (only landing)
Runway 17C/35C: Available (only take-off)
Runway 17R/35L: Available (only take-off)
Runway 17L/35R: Available (only take-off)
Runway 18L/36R: Available
Runway 18R/36L: Available
Flight AA2488
TPA
Runway 10/28: Available
Runway 19L/1R: Available
Runway 19R/1L: Available

Fig. 6.8 Scenario 1



DL query:
Query (class expression)
AircraftcanLand
Execute | Add to ontology
Query results
Direct superclasses (1 of 1)
Aircraft
Direct subclasses (3 of 3)
A321
B737
MD83
Instances (3 of 3)
AA2250
AA2488
AA2629

DL query:
Query (class expression)
AircrafttobeDiverted
Execute | Add to ontology
Query results
Direct superclasses (1 of 1)
Aircraft
Direct subclasses (1 of 1)
B737
Instances (1 of 1)
AA2629

DL query:
Query (class expression)
AircraftcannotLand or NonAvailableRunwayforLandingatDWF
Execute | Add to ontology
Query results
Subclasses (10 of 10)
13L/31R
17C/35C
17L/35R
17R/35L
18L/36R
18R/36L
AircraftcannotLand
MD83
NonAvailableRunwayforLandingatDWF
owl:Nothing
Instances (1 of 1)
AA2488

Fig. 6.9 Query results to assess situation from scenario 1

done for aircraft coming from the southeast of DWF (runway 13L/31R, 17L/35R, 17C/35C, and 17R/35L) but should be advised to land in 18L/36R, 18R/36L, or 13R/31L (west runways).

6.5.3 Application Scenario 2: UAVs Present

Scenario 2 considers Flight AA2250 when it is about to land in LAX (17:11:57 17:16:00 UTC). The airplane has descended from altitude 2575 feet down to 2200 feet (17:11:57 17:12:22 UTC), and from 750 feet down to 0 (17:13:57 17:16:00 UTC). Scenario 2 supposes there are a Large UAV (large-unmanned helicopter which wheelbase is 10 m) and a Medium UAV (a recreational drone which wingspan is 1.8 m) nearby LAX airport by the time Flight AA2250 is descending as specified above. Both UAVs are flying around LAX airport during the landing of Flight AA2250. These UAVs fly high enough to dangerously come close to Flight AA2250 while landing. The *large* UAV (LUAV) is of concern for the descent from 2575 to 2200 feet, and the *medium* UAV (MUAV) for the descent from 750 to 0 feet. The former has a contactable remote pilot and is more than 800 m (first descent segment) and 100 m (second descent segment) away from Flight AA2250. The latter has a non-contactable pilot and is less than 300 m (first and second descent segment) away from Flight AA2250. Figure 6.10 shows scenario 2.

Scenario 2 also estimates veracity of the AAO-based decision-making process by only assessing sensitivity of the information provided by the radars. The range defined for sensitivity is as follows: 0–5%: Very low sensitivity, 5–25%: Low sensitivity, 25–70%: Regular sensitivity, 70–95%: High sensitivity, and 95–100%: Very high sensitivity. The veracity metrics (total veracity) is given by the multiplication of each sensitivity involved in the above process.

Figures 6.11 and 6.12 show AAO query results (including sensitivity metrics) for scenario 2 along with AAO queries for each of the radars that detect the UAVs. T-KFUL8 detects Flight AA2250 (from 2575 to 2200 feet), F-KEMT2 detects Flight AA2250 when it descends from 750 to 0 feet, and

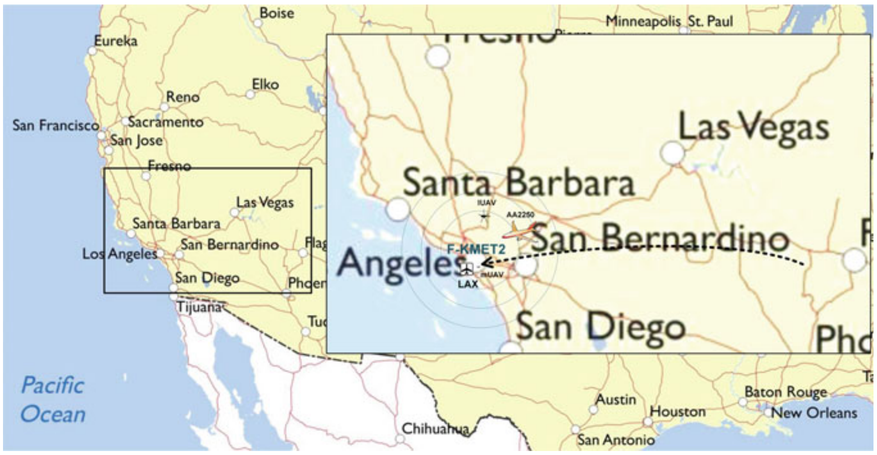
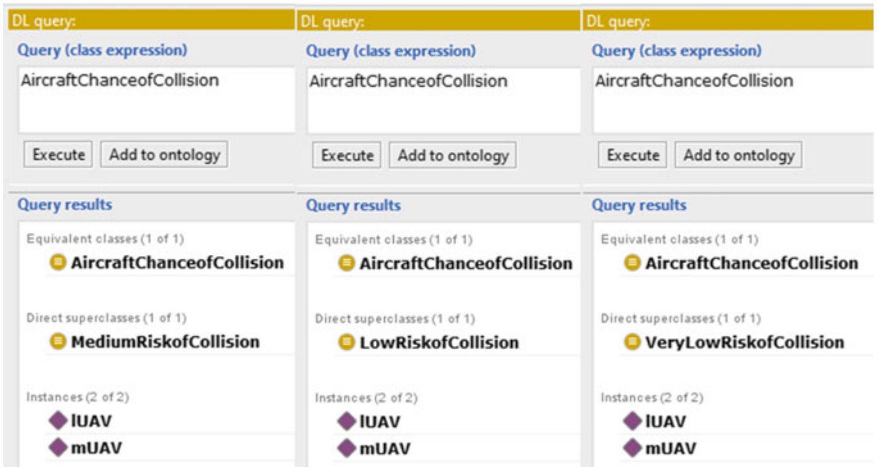


Fig. 6.10 Scenario 2

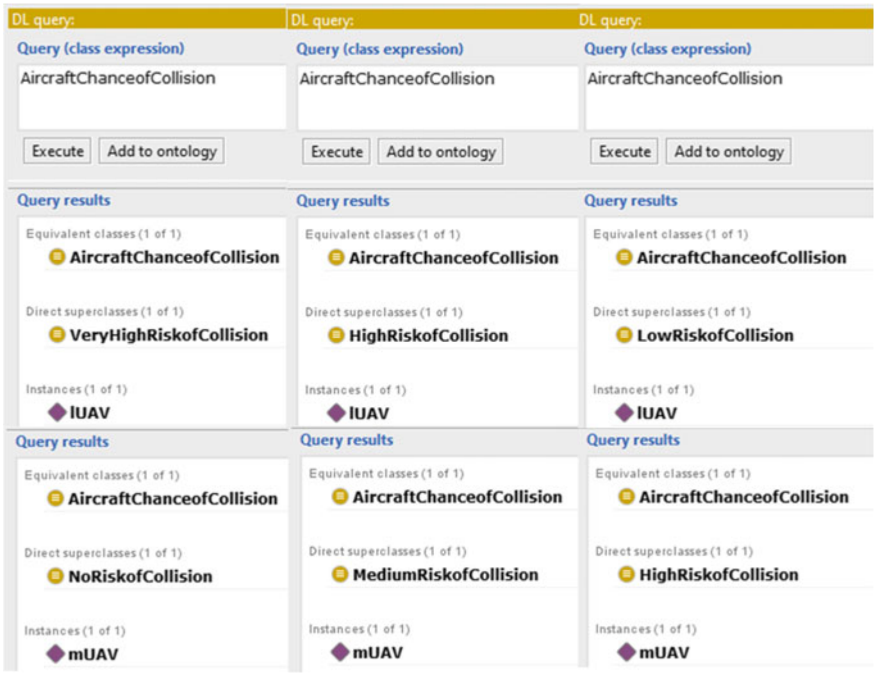


DL query:
Query (class expression)
AircraftChanceofCollision
Execute | Add to ontology
Query results
Equivalent classes (1 of 1)
AircraftChanceofCollision
Direct superclasses (1 of 1)
MediumRiskofCollision
Instances (2 of 2)
lUAV
mUAV

DL query:
Query (class expression)
AircraftChanceofCollision
Execute | Add to ontology
Query results
Equivalent classes (1 of 1)
AircraftChanceofCollision
Direct superclasses (1 of 1)
LowRiskofCollision
Instances (2 of 2)
lUAV
mUAV

DL query:
Query (class expression)
AircraftChanceofCollision
Execute | Add to ontology
Query results
Equivalent classes (1 of 1)
AircraftChanceofCollision
Direct superclasses (1 of 1)
VeryLowRiskofCollision
Instances (2 of 2)
lUAV
mUAV

Fig. 6.11 Query results to assess situation from scenario 2 for descent from 2575 to 2200 feet



DL query:
Query (class expression)
AircraftChanceofCollision
Execute | Add to ontology
Query results
Equivalent classes (1 of 1)
AircraftChanceofCollision
Direct superclasses (1 of 1)
VeryHighRiskofCollision
Instances (1 of 1)
lUAV
Query results
Equivalent classes (1 of 1)
AircraftChanceofCollision
Direct superclasses (1 of 1)
NoRiskofCollision
Instances (1 of 1)
mUAV

DL query:
Query (class expression)
AircraftChanceofCollision
Execute | Add to ontology
Query results
Equivalent classes (1 of 1)
AircraftChanceofCollision
Direct superclasses (1 of 1)
HighRiskofCollision
Instances (1 of 1)
lUAV
Query results
Equivalent classes (1 of 1)
AircraftChanceofCollision
Direct superclasses (1 of 1)
MediumRiskofCollision
Instances (1 of 1)
mUAV

DL query:
Query (class expression)
AircraftChanceofCollision
Execute | Add to ontology
Query results
Equivalent classes (1 of 1)
AircraftChanceofCollision
Direct superclasses (1 of 1)
LowRiskofCollision
Instances (1 of 1)
lUAV
Query results
Equivalent classes (1 of 1)
AircraftChanceofCollision
Direct superclasses (1 of 1)
HighRiskofCollision
Instances (1 of 1)
mUAV

Fig. 6.12 Query results to assess situation from scenario 2 for descent from 750 to 0 feet

F-KLGB3, T-KFUL8, and F-KEMT2 can detect both UAVs. T-KFUL8 is considered to have a sensitivity of 1, F-KEMT2 is considered to have a sensitivity of 0.8, and F-KLGB3 is considered to have a sensitivity of 0.5 when detecting the UAVs in the first descent segment (2575 to 2200 feet). This assumption is done by taking T-KFUL8 as a reference radar since it detects Flight AA2250. Therefore, veracity is 45% when the decision is made based on the above three radars detecting the UAVs, 80% when the decision is made based on T-KFUL8, 50% when the decision is made based on F-KLGB3 and F-KEMT2, and 100% when the decision is made based on T-KFUL8.

The query inference results suggest that (from left to right) (1) LUAV and MUAV have some chances of collision (medium risk of collision) and the veracity of this query is based on a sensitivity of 100% when T-KFUL8 detects the UAVs and Flight AA2250 (2) LUAV and MUAV have low chances of collision (low risk of collision) and the veracity of this query is based on a sensitivity of 80% (low to make a trusted decision) when F-KEMT2 detects the UAVs, and (3) LUAV and MUAV have little chances of collision (very low risk of collision) and the veracity of this query is based on a sensitivity of 50% (very low to make a trusted decision) when F-KLGB3 detects the UAVs. The first suggestion of a collision is the most veracious out of the three suggestions.

F-KEMT2 is considered to have a sensitivity of 1.0, T-KFUL8 is considered to have a sensitivity of 0.75, and KLGB3 is considered to have a sensitivity of 0.55 when detecting the UAVs in the second descent segment (2575 to 2200 feet). This assumption is done because F-KEMT2 is located closer to the UAVs than the other two radars in question. Therefore, the veracity of the top left query is 41.25% when the decision is made based on the above three radars detecting the UAVs, 75% when the decision is made based on T-KFUL8, 55% when the decision is made based on F-KLGB3 and T-KFUL8, and 100% when the decision is made based on F-KEMT2.

The query inference results suggest that (from left to right) (1) LUAV has very high chances of collision (very high risk of collision) and the veracity of this query is based on a sensitivity of 100% (very high to make a trusted decision) when F-KEMT2 detects the UAVs. On the other hand, MUAV has no chances of collision (no risk of collision) and the veracity of this query is based on the same sensitivity (100%); (2) LUAV has high chances of collision (high risk of collision) and the veracity of this query is based on a sensitivity of 75% (relatively high to make a trusted decision) when T-KFUL8 detects the UAVs, and MUAV has some chances of collision (medium risk of collision); and (3) LUAV has high chances of collision (low risk of collision) and the veracity of this query is based on a sensitivity of 41.25% (low to make a trusted decision) when the three radars are considered for the detection of the UAVs, and MUAV has high chances of collision (high risk of collision). The first suggestion of a LUAV collision is the most veracious out of the three suggestions.

These scenarios with UAVs in the airspace describe potential events with clutter airspaces composing commercial aircraft and unmanned aerial vehicles. In both cases, the ontology support alerts and warnings, along with veracity metrics, to air traffic controllers (ATCs). The ATCs could communicate with the aircraft, gather more information from the airspace, or imitate methods for collision avoidance.

6.6 Concluding Remarks and the Way Forward

An implementation of *Ontologies for NextGen Avionics Systems* (ONAS) has been presented. An operation framework and an ontology-based process to support decision-making in advanced ATM/UTM systems have been proposed along with examples of implementation. The AAO implementation is a simple but useful proof of concept for ONAS. The proposed ONAS approach includes a cognitive ATM/UTM architecture for avionics analytics and Situation AWareness (SAW) to maintain safe distances and alternative route planning for different weather conditions. The SAW approach proposed is intended to be used in civil aviation with potential use on board in autonomous UAVs in order to increase autonomy. Results from two scenarios as application examples have been discussed. The realistic scenarios are simple, but provide a discussion for which an ATM/UTM system which considers semantics from updates of weather maps, airport maps, and route maps. The above scenarios have discussed flight situations where the controllers' decisions can be made by means of the support of the ontological approach proposed.

Future research work will involve methods to improve the AAO implementation and veracity metrics [33]. One of the relevant approaches as an interesting veracity metric to be considered for further investigation is the big data veracity index [34]. It is based on three main dimensions to define veracity: objectivity (subjectivity), truthful (deception), and credibility (implausibility). The index approach deserves attention, but some research is required to deal with artificial autonomy (DSS) since the potential tools to support such a metric index are too human-oriented. The challenge is to develop supporting tools that allow for machine veracity metrics, e.g., radars. On the other hand, some future refinement on the integration of veracity (uncertainty) into the AAO will enhance usefulness. One of the inspiring methodologies (to deal with probabilistic uncertainty when making decision) is the Bayesian networks, e.g., BasesOWL [35] which is suitable for ontologies. These enhancements in the ontology will support contextual reasoning over data, environment, and decision-making [36].

References

1. E. Blasch, P. Paces, P. Kostek, K. Kramer, Summary of avionics technologies. IEEE Aerosp. Electron. Syst. Mag. **30**(9), 6–11 (2015)
2. T. Gaska, C. Watkins, Y. Chen, Integrated moduLar avioincs - past, present, and future. IEEE Aerosp. Electron. Syst. Mag. **30**(9), 12–23 (2015)
3. A. Helfrick, The centennial of avioncs: our 100-year trek to performancebased navigation. IEEE Aerosp. Electron. Syst. Mag. **30**(9), 36–45 (2015)

4. J. Leuchter et al., Investigation of avionics power switch loading versus aircraft electromagnetic compatability. IEEE Aerosp. Electron. Syst. Mag. **30**(9), 24–34 (2015)
5. F. A. Association, Federal Aviation Association Next Generation (NextGen) (2017), https://www.faa.gov/nextgen
6. E. Commission, Single European Sky ATM Research (SESAR) (2017), http://ec.europa.eu/transport/modes/air/sesar/index_en.htm
7. R.M. Keller et al., Semenatic representation and scale-up of integrated air traffic management data, in *International Workshop on Semantic Big Data* (2016)
8. F. AB, Flightradar24 Live Air Traffic (2006), https://www.flightradar24.com
9. E. Blasch, Enhanced air operations using JView for an air-ground fused situation awareness UDOP, in *AIAA/IEEE Digital Avionics Systems Conference* (2013)
10. E. Blasch, Ontologies for NextGen avionics systems, in *AIAA/IEEE Digital Avionics Systems Conference* (2015)
11. E. Blasch, M. Belanger, Agile battle management efficiency for command, control, communications, computers and intelligence (C4I). Proc. SPIE **9842** (2016)
12. EUROCONTROL. AIRM Primer (2015), https://www.eurocontrol.int/sites/default/files/content/documents/sesar/8.1.3.d47-airm-primer-v4.1.0.pdf
13. C. Flachberger et al., Collaboration in crisis management - learning from the transportation domain, in *Future Security* (2015)
14. M. Zhao et al., The research synopsis about SWIM in china, in *IEEE 12th International Symposium on Autonomous Decentralized System* (2015)
15. F. Burgstaller et al., AIRM-based, fine-grained semantic filtering of notices to airmen, in *Integrated Communication, Navigation and Surveillance Conference* (2015)
16. R. Koelle, A. Tarter, Towards a distributed situation management capability for SESAR and NEXTGEN, in *Integrated Communication, Navigation and Surveillance Conference* (2012)
17. R. Koelle, W. Strijland, Semantic driven security assurance for system engineering in SESAR/NextGen, in *Integrated Communication, Navigation and Surveillance Conference* (2013)
18. J.R. Boyd, *The Essence of Winning and Losing* (1996)
19. E. Blasch, D.A. Lambert, P. Valin, M.M. Kokar, J. Llinas, S. Das, C.-Y. Chong, E. Shahbazian, High level information fusion (HLIF) survey of models, issues, and grand challenges. IEEE Aerosp. Electron. Syst. Mag. **27**(9), 4–20 (2012)
20. E. Blasch, A. Steinberg, S. Das, J. Llinas, C.-Y. Chong, O. Kessler, E. Waltz, F. White., Revisiting the JDL model for information Exploitation, in *International Conference on Information Fusion* (2013)
21. R. So, L. Sonenberg., Situation awareness in intelligent agents: foundations for a theory of proactive agent behavior, in *IEEE/WIC/ACM International Conference on Intelligent Agent Technology* (2004), pp. 86–92
22. E. Blasch, S. Plano, Level 5: user refinement to aid the fusion process. Proc. SPIE **5099** (2003)
23. E. Blasch, I. Kadar, K. Hintz, J. Biermann, C.C.S. Das, Resource management coordination with Level 2/3 fusion issues and challenges. IEEE Aerosp. Electron. Syst. Mag. **23**(3), 32–46 (2008)
24. M.C. Dorneich et al., Evaluation of information quality and automation visibility in information automation on the flight deck, in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (2015), pp. 284–288
25. J. Rushby, Evaluation of information quality and automation visibility in Information automation on the flight deck, in *International Conference on Distributed Computing and Internet Technology (ICDCIT)*, vol. 9581 (2016), pp. 19–29
26. E. Blasch, Ontological issues in higher levels of information fusion: user refinement of the fusion process, in *International Conference on Info Fusion* (2003)
27. P.C.G. Costa et al., Towards unbiased evaluation of uncertainty reasoning: the URREF ontology, in *International Conference on Info Fusion* (2012)
28. G. Schreiber et al., *Knowledge Engineering and Management - The CommonKADS Methodology* (MIT Press, Cambridge, 1999)

29. F. Baader, et al. (eds.), *The Description Logic Handbook - Theory*, Implementation and Applications (MIT Press, Cambridge, 2003)
30. G.D. Giacomo, M. Lenzerini, TBox and ABox reasoning in expressive description logics. Tech. report, AAAI Technical Report WS-96-05. 1996
31. S. University, The Protégé Wiki (2017), http://protegewiki.stanford.edu/wiki/Main_Page
32. C.C. Insaurralde, E. Blasch., Ontological knowledge representation for avionics decision-making support, in *AIAA/IEEE Digital Avionics Systems Conference* (2016)
33. C.C. Insaurralde, E. Blasch., Veracity metrics for ontological decision- making support in avionics analytics, in *AIAA/IEEE Digital Avionics Systems Conference* (2017)
34. T. Lukoianova, V. Rubin., Veracity roadmap: is big data objective, truthful and credible? in *Advances in Classification Research Online* (2014)
35. Z. Ding et al., BayesOWL: Uncertainty Modelling in Semantic Web Ontologies, in *Soft Computing in Ontologies and Semantic Web*, vol. 204 (2006) pp. 3–29
36. L. Snidaro, J.G. Herrero, J. Llinas, E. Blasch (eds.), *Context-Enhanced Information Fusion: Boosting Real-World Performance with Domain Knowledge* (Springer, Berlin, 2016)

Chapter 7
Advances in Software Engineering and Aeronautics

Shafagh Jafer, Umut Durak, Hakan Aydemir, Richard Ruff and Thorsten Pawletta

Abstract Avionics, like any other safety-critical real-time systems, pose unique challenges on system design, development, and testing. Specifically, the rigorous certification process mandated for avionics software calls for additional attention. The DO-178C Software Considerations in Airborne Systems and Equipment Certification provides detailed guidelines to ensure safety measures. This chapter gives a different angle to avionics development and certification, highlighting model-based approaches for advancing the design, development, testing, and maintenance of airborne software systems. Modern software engineering processes such as agile and scrum are discussed as the new techniques in speeding up the certification hurdle, while achieving higher return on investment.

7.1 Introduction

Complexity of modern aircraft is increasing day by day in order to enhance safety and boost the economy of flight while number of passengers and flights are increasing drastically. To provide safety and comfort for passengers as well as pilots, the

S. Jafer (✉)
Embry Riddle Aeronautical University, Daytona Beach, FL, USA
e-mail: jafers@erau.edu

U. Durak
German Aerospace Center (DLR), Braunschweig, Germany
e-mail: umut.durak@dlr.de

H. Aydemir
Turkish Aerospace Industries (TAI), Ankara, Turkey
e-mail: haydemir@tai.com.tr

R. Ruff
The MathWorks, Dallas, TX, USA
e-mail: richard.ruff@mathworks.com

T. Pawletta
Wismar University of Applied Sciences, Wismar, Germany
e-mail: thorsten.pawletta@hs-wismar.de

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_7

number of avionics functions added to avionics architecture increase every year. The commonly known architecture to implement flight control and other avionics systems such as navigation, communication or radar was the federated architecture. In the 70s, all functions were separated. Starting from the 80s some of related functions became integrated, and eventually in the 90s all functions became integrated.

In federated architecture, each function had its own fault-tolerant computer system. The most important advantage of federated architecture was that a fault in any function would not affect any other function. Managing vendor providers were relatively easy because requirements could be defined separately for each avionic equipment. The only common element between functions was the data communicated. Shared resources such as sensor data used by each function were kept in limited numbers as much as possible and resource sharing was provided via deterministic secured data buses. Of course, this architecture required more hardware which eventually means more weight, yielding higher fuel consumption. The increasing amount of features boosted the complexity, yielding higher development and maintenance cost, therefore creating a large burden for modern aircraft.

Integrated Modular Avionics (IMA) is proposed as an alternative concept for avionics architecture and became popular in the 90s [1, 2]. The main idea of IMA is sharing same hardware for different functions. This concept reduces the cost of hardware, wiring, and accordingly design and maintenance. An example of IMA is the Airplane Information Management System (AIMS) which is used in Boeing 777 [3]. The AIMS is the brain of the Boeing 777 and it combines primary flight display, navigation display, Engine-Indicating and Crew-Alerting System (EICAS) display, flight planning, navigation, performance management, airplane and engine monitoring, communications management, digital flight data acquisition, and data conversion gateway functionalities into a single integrated system.

An important keyword of IMA is partitioning. Partitioning is crucial in IMA to stop fault propagation from one application to another. It utilizes appropriate hardware and software mechanisms to provide strong fault containment [4]. Partitioning can be provided above an operating system layer or above a minimal kernel with most operating system services which rely on the same hardware. Two dimensions were introduced in partitioning: spatial partitioning and temporal partitioning. Spatial leads to the principle of ensuring that the software in one partition can neither change any data nor issue a command to any device on another partition. Temporal partitioning dictates the principle to ensure that the activities in one partition do not disturb the timing of events in other partitions. An example partitioning architecture is depicted in Fig. 7.1.

Certification of avionics is a big challenge. Each airborne equipment has its own criticality levels and should have been appropriately certified. ED-12C/DO-178C Software Considerations in Airborne Systems and Equipment Certification [5], which is developed by RTCA (Radio Technical Commission for Aeronautics) and EUROCAE (a nonprofit organization providing a European forum for resolving technical problems with electronic equipment for air transport), is the primary reference for software certification. It describes criticality levels associated with the failure conditions (Table 7.1). After a safety assessment and hazard analysis has been conducted

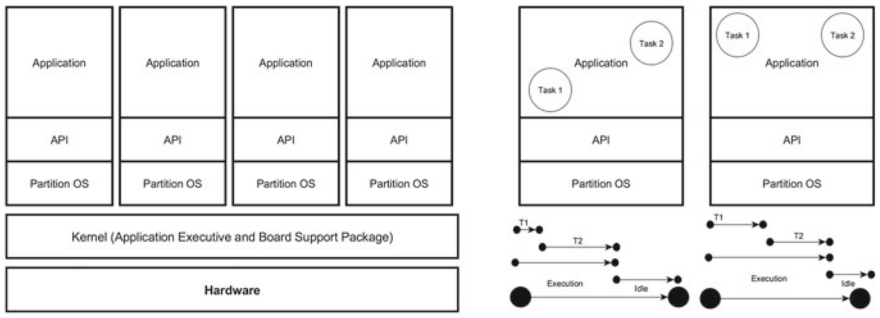


Fig. 7.1 An example partitioning architecture

Table 7.1 ED-12C/DO-178C criticality levels

Level	Failure condition
A	Catastrophic
B	Hazardous
C	Major
D	Minor
E	No safety effect

by examining the effects of a failure condition in the system on the aircraft, crew, and passengers, a criticality level is identified for the software. A Design Assurance Level (DAL) and a set of certification requirements will then follow.

In IMA architecture, each function in each partition can have its own assurance level. It allows to assign different DAL for each function. Thus, it is possible to design a flight control computer that has both a Level A autopilot function and a Level D maintenance function. Without partitioning, all software certification processes should be completed according to Level A requirements. Maintenance parts should be implemented in a separate hardware, which will add cost and time. Partitioning provides the possibility to implement software with different assurance levels in the same hardware.

On one hand, IMA architecture separates main framework and applications so each application can be provided by different subcontractors; on the other hand, it makes integration more complex. According to Watkins and Walter [6], before transitioning from federated architectures to IMA, the system integrator needs to be confident in its ability to perform the integration process, which includes increased interface definition and management, resource allocation and management, and system configuration analysis and generation.

Avionics software development has unique challenges on system design, development, and testing. The characteristics of avionics software can be introduced with six important qualities: Functionality, Reliability, Maintainability, Portability, Usability, and Efficiency.

Functionality is one of the main qualities of any product that describes how well it suits its intended purpose. Functionality is an utmost importance in avionics software. It shall provide exactly all functionalities as specified in its requirements.

Reliability is the quality of a product which describes how reliable it is. The reliability required by a complete aircraft system provides a bound on the reliability required by the avionics and software embedded in it [7]. Avionics software failure leads to catastrophic results, thus, mandating developers to consider software failures and provide mitigation mechanism.

Maintainability is a quality which designates the ease of preserving a product from failure or decline. It corresponds to the ability to identify and fix a fault within a software component [8].

Portability is being defined as how the software can adapt to changes in its target hardware environment. Porting an avionic software on a different hardware is a highly expensive and time-consuming process. There is a recent initiative which targets increasing portability of avionics softwares, Future Airborne Capability Environment (FACE) Consortium [9], by defining a modular, and portable software environment supporting IMA in avionics systems.

Usability designates the ease of use in a product. The nature of avionics software is embedded, running on various special hardware to deliver expected functionality. Pilots, crew, or operators control these functions via user interfaces or hard buttons or keyboards. Pilots or operators can undergo many situations while onboard, requiring them to take actions promptly. Usability of avionics software is crucial in order to reduce human operator error.

Efficiency is concerned with system resources used when providing the required functionality [8]. As with reliability, the causes of performance inefficiency are often found in violations of good architectural and coding practices which can be detected by measuring the static quality attributes of an application [10].

The authors promote model-based approaches and agile development practices as the major advancements in recent software engineering landscape. After an introduction that furnishes an overview about the avionic systems and software, the chapter will present advances in avionics software development with a special emphasis on agile development and model-based approaches as its enabler.

7.2 Advances in Avionics Software Engineering

7.2.1 Extreme, Agile, Scrum, and Beyond

Developing safety-critical systems is more challenging than developing non-safety-critical systems. While safety-critical systems stress on high quality, low costs, and schedules, they are subject to a rigorous certification processes in order to meet required safety standards. The purpose of the certification process is to ensure that the system to be used in a specific environment under specific conditions is safe. The Federal Aviation Administration (FAA) is imposing safety requirements on the development and verification of airborne avionic systems stated in DO-178C [5]. The latest DO-178C guidance includes modern technologies and methodologies necessary to achieve a more reliable and safe system within a constrained time and cost. It also describes various guidelines to engineer (design, specify, develop, test, and deploy) a software component and all associated equipment with a certain level of safety that complies with the FAA airworthiness requirements. Although these standards define the requirements for a process to remain compliant when used to develop a safety-critical system, the standards do not specify which process to use. Thus, software developers can use any preferred processes if it meets the objectives and safety standards of DO-178C, providing the option to use Agile, Formal Methods (supported by DO-333 Formal Methods Supplement [11]), and Model-Driven Development (detailed in DO-331 supplement [12]) with all their associated advantages in the certification and development process.

Most common Agile methodologies are scrum [13], XP (eXtreme Programming) [14], Crystal [15], FDD (Feature-Driven Development) [16], TDD (Test Driven Development) [17], and ASD (Adaptive Software Development) [18]. These methodologies share the same philosophy and promote the values of agile manifesto; but from the implementation perspective, each has its own terminology, practices and tactics.

While Agile is significantly accepted in industry, it is not as widely practiced in safety-critical systems development because of its undisciplined nature when it comes to documentation and lack of rigorous verification and validation techniques. Additionally, using the agile process as a standalone method to develop a safety-critical system has been quite challenging due to the quality control mechanisms used by agile such as informal reviews and pair-programming, which have not assured developers or authorities that the product is safe [19]. On the other hand, formal Methods [20] are not preferred in a conventional software development process, but instead are used partially in the verification process of safety-critical systems. Many claim that using formal methods can increase project budget and the time to market because of the extra workload added in the design and analysis phases of the Software Development Life Cycle (SDLC) [21]. Traditionally, simulation and rigorous testing methods are used to verify and validate system specification. However, applying these techniques allow for partial verification of system behavior instead of proving the correctness of the entire system [22, 23]. A distinct advantage of formal Methods is providing precise and unambiguous descriptions and mechanisms that facilitate the development of safety-critical systems in a more robust fashion.

The growing complexity of software systems is the motivation behind the current introduction and acceptance of model-based approaches [24, 25] and tools in the avionics industry, as it has proven to reduce the gap between the abstract level of system requirements and software implementation. Through the use of supporting tools and technologies such as automatic code generation, formal specification, simulation and verification tools, model-based approaches have improved the efficiency of complex system development and provided the same level of quality as traditional processes in a shorter time and lower cost. Model-based approaches use models that define a system at multiple levels of abstraction and from a variety of perspectives through transforming and analyzing models. Therefore, incorporating an agile process with formal methods and model-based approaches might result in a more promising solution that will solve the current safety-critical software development issues related to project budget and completion time.

Furthermore, there has been a systematic and multidisciplinary study to find out the effectiveness of model-based approaches in broad terms. For example, there is currently a massive focus on the integration of model-based approaches and agile processes in safety-critical software development. For example, a new software development process has been proposed in [26] that merges advantages of both agile and model-based development practices to establish a Model-Based Agile (MBA) process that capable to satisfy the DO-178 software development objectives. MBA process significantly relies on the use of Unified Modeling Language (UML) that preserves the iterative and incremental nature of gile technique in order to capture, refine and verify systems requirements. IBM tools such as Rhapsody [27] and DOORS [28] are example tools for managing requirements; where they have demonstrated compliance with DO-178C requirements. This allows developers to maintain traceability of top-level system properties and requirements to the specification model, as dictated by the standards.

7.2.2 Designing with Models

Model-Based Design (MBD) [29] is a methodology that accelerates the development time and reduces the overall cost of designing complex avionics. At the core of MBD is a model that defines the system and is used throughout the development process. This model includes not only the algorithms that will form the implementation, such as an autopilot, but also environmental and physical aspects of the system that impact the design. These models might involve many different domains, including mechanical, electrical, and hydraulic systems. For avionics, the physical and environmental models can be derived from empirical data, from testing, such as wind tunnel data, wind tunnel data to define the aerodynamic characteristics, or first-principle equations of motion that define the motion of the vehicle. MBD enables the modeling and simulation of systems that span these disparate domains in a single environment. This common framework means engineers can collaborate, providing valuable insight into the interaction between components. MBD provides a common language to share information between departments, as well as between manufacturers and suppliers. With MBD, engineers can perform both component-level and system-level simulations and testing at all stages in the development process. This lets teams detect errors earlier in the design process when they are much less costly to fix. In addition, the ability to simulate the system means engineers can find incompatibilities between components earlier, reducing the cost of integration. The continuous verification and validation aspect of MBD makes it especially valuable, accelerating the design time and reducing errors. A traditional development workflow can be broken down into five steps:

1. Requirements specification
2. Design
3. Implementation
4. Integration
5. Test and verification

With MBD, these same components exist; however, they are not performed sequentially, but iteratively throughout the development process. Figure 7.2 illustrates a typical workflow for MBD. Starting with a set of requirements (and possibly some research information), the team develops a model of the system. As the design progresses, simulation verifies that the design meets the requirements and that the system performs as desired. At this stage in MBD, areas where the requirements are incomplete or possibly conflicting can be identified. This enables requirement engineers to go back and update the requirements as needed. In a traditional development workflow, these inconsistencies are found much later in the development process, when it costs significantly more to fix them. As the design matures, the team can implement their design using automatic code generation. Engineers can test and verify the resulting code using the same test cases used to test and verify the model. Once the implementation has been tested and verified, it can be integrated with other design components and hardware. The integrated system can then be tested and verified. The process of designing and testing, implementing and testing, and integrating and testing can be repeated as the system matures throughout the development workflow. With MBD, the test and verification phase occurs with each of the preceding phases: design, implementation, and integration, and not at the end of the development process, as with a traditional development workflow. Continuous test and verification is the key aspect of MBD that enables teams to find and correct errors much earlier in the development process, reducing both the cost of development and the time to market.

Embedded processors have become ubiquitous in everyday life, from smartphones to washing machines, from environmental controls in automobiles to entertainment systems on commercial aircraft. In avionics, the computing power these new embedded processors provide has led to many advances, including new and improved sensors and actuators, cockpit visualization, and fly-by-wire systems that work across a network to share data. These enhancements have in turn increased the complexity of these systems dramatically. While complexity has increased, it does not change the level of testing required to meet a standard such as DO-178C [5], however it does increase the amount of testing needed to ensure the safety of avionics systems, and subsequently, the number of artifacts needed to confirm compliance. Using MBD enables teams to automate many of these tasks, reducing the manual effort a traditional development process requires. Using MBD, engineers can link system requirements to the model and to the resulting automatically generated code. Automated code reports provide the required traceability between the code and the requirements

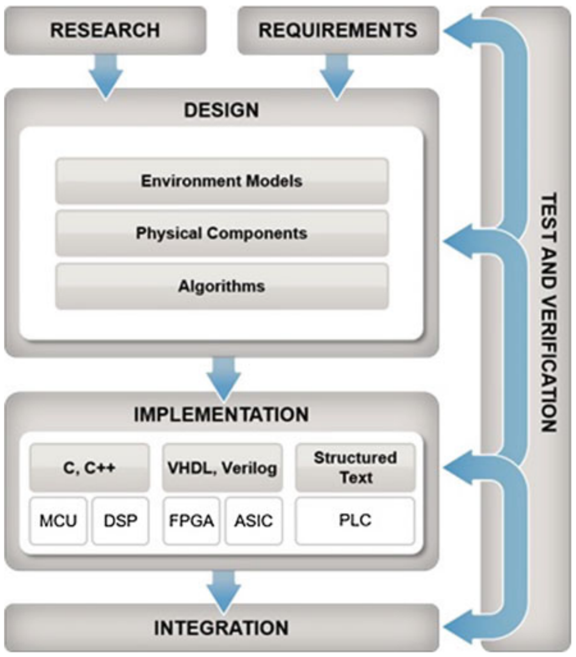


Fig. 7.2 Workflow for MBD

document. In MBD, test cases can be created and used throughout the development process and test results reports are automatically produced. As with the model and the resulting automatically generated code, the test cases can be linked to the requirements. Teams can use verification and validation tools to provide MCDC coverage of the model and the resulting automatically generated code to demonstrate full testing. These tools also verify that the code is equivalent to the model to eliminate the need for code reviews, and check the code for potential run-time errors. These automations can also be used on future projects, reducing development costs even further. Combined with the continuous testing and verification, it is this automation that makes MBD so valuable for avionics system development. To ensure the successful use of MBD for avionics software development, it is imperative that the development process be defined with MBD in mind. This means determining which tools for MBD the team will use, the automations they will employ, and the artifacts that will be required to show compliance with the software standard. MBD has been used successfully for many avionics applications, improving the final products while reducing the development cost. Airbus used MBD to develop a fuel management system for the A380 and for the air conditioning software for the EC130 helicopter. Lockheed Martin employed MBD for the Joint Strike Fighter control law development. BAE uses MBD across diverse disciplines from developing flight control computers for a business jet to an autopilot for a UAV, as well as a software-defined radio application and a mobile antenna pointing system. Bell Helicopter developed the world's first fly-by-wire helicopter, the Bell 525 Relentless, using MBD. MBD is becoming increasingly prevalent in avionics software development. With its advantages in reducing development cost, it is often being specified as the method to use for new projects. Suppliers and subcontractors are being directed to use MBD and to provide component models that can be used in the system integrators system model. The DO-178 standard was recently updated to DO-178C, along with addendum DO-331, which specifically addresses the use of MBD for safety-critical software in avionics. However, many current avionics systems are upgrades or modifications of existing systems, developed using more traditional approaches. Switching to MBD can be deemed to be too expensive or too risky due to the necessary changes in process and the learning curve. Nonetheless, there are ways to leverage MBD for existing systems. The recommended method is to use MBD to develop a single component of a system. The team can then apply the lessons learned and automations from this component to other components. This phased approach lets an organization adopt MBD gradually and avoid the challenges of switching completely from a traditional approach to MBD. Another reason sometimes cited for not adopting MBD is concern over the quality, efficiency, and readability of automatically generated code. Over the past 20 or more years, many advancements in automatic code generation have addressed these initial concerns about automatically generated code. For teams adopting MBD for avionics software development, the benefits of using MBD quickly eclipse the challenges of using a new development workflow. Current tools for MBD continue to evolve, making MBD easier to adopt, and further improving its high return on investment.

7.2.3 Developing with Models

A main concern when constructing complex safety-critical systems is the production of proof, which can be used to validate systems to an acceptable level of confidence. Such acceptance level is achieved when errors in requirements, design, and implementation of a system have been identified and fixed, and any possible effects of remaining errors are automatically mitigated by the system architecture. Model-Driven Development (MDD) generalizes and extends the MBD by proposing the formal modeling of all the system properties to support overall system development processes, which includes generation of requirements, specification, and design at different system levels, as well as the validation and verification of systems that contain associated subsystems and components [30]. In this sense, it covers the MBD. MDD shifts the immediate focus from generation of data and proof documents to the generation of project work products (e.g. design models, development schedules, analysis models, etc.). Such approach improves software development practice by promoting [31]:

- Increased productivity and reduced cost: Producing artifacts and code from models, which increases developer productivity as well as overall development cost.
- Documentation: with MDD, documents are generated from models, ensuring consistency and information availability within the models on a daily basis, rather than in documents that are difficult to navigate. Tools such as IBM Rational SoDA and IBM Rational Software Architect Report Generator automatically generate documentation.
- Maintainability: MDD offers a maintainable architecture where changes are made rapidly and consistently, enabling effective integration of components onto new technologies.
- Consistency: Automatically generated artifacts straight from models, ensures efficiency and consistency.
- Delayed technology decisions: early MDD phases focus on modeling activities, which help in postponing the implementation details (e.g. choice of certain technology platform or product version) until a later time, where additional details become available. In domains with lengthy development cycles, such as aviation, this is crucial as target platforms may not even exist when initiating the development activities.
- Automation: is the main feature that distinguishes MDD from other approaches. Automation ensures that software components are consistently implemented, which leads to an improved product quality with shorter project costs and shorter delivery time.
- Additional benefits of using MDD include the use of common design artifacts, formal proofs, facilitating traceability, improving communication and insight among project participants, as well as easier maintenance [24].

7.2.4 Managing Variability

With the advent of mass customization ideas, systems are now coming with a divergent set of end user requirements, which lead to vast amount of variants. The ability of a system or an artifact to be configured, customized, or extended for employment in a particular context is defined as variability [32].

Two main approaches for modeling variability are named as positive and negative. Positive variability involves starting from a minimal core and selectively adding additional parts based on the configuration, whereas in negative variability parts are taken away selectively from a maximum overall model [33].

Variability mechanisms can be defined at various levels of abstraction, ranging from metamodels to implementation of the source code. Referring to the layers of the Meta-Object-Facility (MOF) hierarchy [34], they can appear on the M2, M1, or the M0 layers. Variability modeling can be achieved in M2 layer using metamodels combined with automated model transformations to executable system models in M1 layer. Metamodeling is often conducted using general- purpose modeling languages such as UML [35] or feature models [36]; specific software tools for requirement or variant management such as pure variants [37]; or domain-oriented approaches and tools such as in [38].

In the engineering domain, the executable system models in M1 are often design models that are developed using MBD environments such as MATLAB/Simulink. Since the necessary model transformations or tool integrations are a particular challenge, in engineering applications variability is often encoded in the M1 layer within the executable system models. Examples of such approaches particularly for the MATLAB/Simulink environment can be found in [39–41].

Among others, Software Product Lines (SPL) can be presented as the widely employed approach for developing systems that are characterized by a high degree of variability [42]. The Carnegie Mellon Software Engineering Institute defines SPL as a set of software-intensive systems with a common and managed set of features for a mission [43]. They provide variation points where different variants of the products can be derived for varying features based on varying requirements.

Variability management is regarded as a key development practice in ever growing cyber-physical system domain. Examples from automotive domain would include [41, 44, 45]. Sozen and Merlo state that regarding the size and complexity of model avionics software, adapting variability management and SPL approaches improves the development process, reduce the cost, and improve the time to market [46]. Some of the publications that report experiences from avionics software development include Dordowsky et al. who utilized a SPL approach for the development of NH90 helicopter avionics system [47, 48], and Braga et al. who used feature modeling and SPL for the development of software for unmanned air vehicles [49].

7.2.5 Testing with Models

Modeling which involves developing mathematical models of physical systems and formal models of computation is an essential component of design in aeronautical systems [50]. Models are not only utilized to simulate the systems for analysis and the evaluation of design decisions, but also, as mentioned in design with models section, by means of automated and semi-automated methods implementation is synthesized from the models.

The heterogeneity and complexity of system components and the networking between them correspond to a complexity and modularity in system models. On these grounds, an adequate test coverage is only possible with a large number of test cases. Therefore, adaptability, flexibility, and automation are desirable quality attributes of the testing infrastructures.

Model-Based Testing (MBT) proposes automating test case generation from a test specification, also called a test model, instead of implementing test cases manually [51]. It further enhances the flexibility and adaptability of the testing infrastructure by automating the test case design [52].

Roßner et al. [53], introduce MBT as a process which is aligned with model-based development (Fig. 7.3). In MDD, a formal system model is first derived based on the

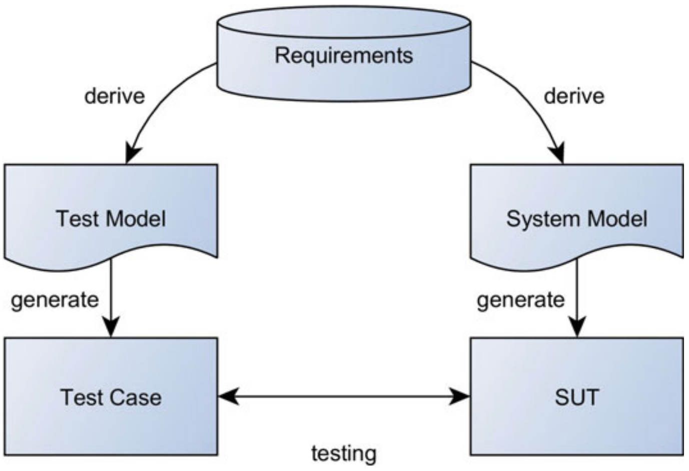


Fig. 7.3 Model-based testing

system requirements. It represents a simplification of the structural and behavioral relationships of real system components. In the next step, executable model components can be generated from the formal system model. MBT uses the same system requirements to derive a test model that is able to generate one single test case or a test suite for a System Under Test (SUT). Test cases are abstracted in a test model that describes the intended behavior of the SUT, which needs to be tested. Then an MBT tool generates a set of executable test cases from that test model [52].

One of the initial ideas about employing MBT with MBD was introduced by Zander [54]. Her Model-in-the-Loop for an Embedded System Test (MiLEST) provides well-structured libraries for test data generation, test control, and test evaluation for MATLAB/Simulink. Almost at the same time, MathWorks started providing Simulink Verification and Validation [55] for the realization of MBT in MATLAB/Simulink. As in MiLEST, Simulink Verification, and Validation also provided library blocks that target test functions.

The application of MBT in aeronautics domain has long been investigated. The Test and Automation Framework (TAF) approach of Lockheed Martin which provides model-based test vector generation and test automation features was one of the early examples [56, 57]. In another study, Stallbaum and Rzepka propose a UML profile to support MBT in the avionics domain [58]. A recent study proposes a MBT process, Fail-Safe MBT, for the development of aeronautical software [59]. It aims at semi-automatically derive means for compliance. Different from the avionics applications, Durak et al. proposes a MBT approach for objective fidelity assessment in flight simulators [60].

7.2.6 Software Evolution and Modernization

The laws of software evolution state that software systems, which are actively used and embedded in a real-world domain are subject to an inevitable change and evolution [61]. Evolution causes aging and erosion in software systems, which appear as sourceless executables, dead data, dead code, inconsistencies, and missing capacities [62]. Software eventually degrade in performance, become buggy and lose their customers and market [63].

Overall replacement of existing systems is risky and not always economical. Therefore, reengineering is introduced to tackle software aging and erosion. It focuses on preserving the knowledge embedded in the legacy system by proposing an evolutionary maintenance of the legacy systems in order to reduce risk and cost [64]. Efforts to define a standard reengineering process for software systems have led to an Object Management Group (OMG) standard, Architecture-Driven Modernization (ADM), which specifies approach for the modernization of existing systems with a broad focus on all aspects of the current system and a promise of transformation to target architectures [65].

ADM advocates a model-based approach to software modernization. Knowledge Discovery Metamodel (KDM) is utilized to extract the knowledge from software assets [66]. Model transformations are then recommended as a means of modernization for legacy assets.

Modernization of software assets has also been a topic in aerospace. In 2001, Norton and Decyk presented a code modernization approach for legacy mission scientific software [67]. Sarkarati et al. introduce an ADM approach for legacy space data systems in 2008. Similarly, Durak in his 2015 paper [68] utilized an extension to KDM for ADM of legacy flight simulation assets.

7.3 Conclusion

Model-based approaches are emerging as economical techniques and enablers of agile practices in developing safety-critical systems where tremendous attention must be paid to satisfy certification requirements. They not only aim at the requirements and design phase, but also provide numerous automation and traceability measures throughout the entire software development life cycles. This plays a key role in addressing certification needs while saving big on resource and efforts allocation. This chapter takes a model-based approach in exploring certification and safety requirements for designing, developing, testing, and modernizing airborne software systems. Challenges in each phase are discussed and model-based approaches for tackling them are highlighted.

References

1. C.B. Watkins, Integrated modular avionics: managing the allocation of shared intersystem resources, in *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA* (IEEE, 2006), pp. 1–12
2. RTCA. DO-297: Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations. Technical report, RTCA (2005)
3. B. Aleksa, J.P. Carter, Boeing 777 airplane information management system operational experience, in *Digital Avionics Systems Conference, 1997. 16th DASC AIAA/IEEE*, vol. 1 (IEEE, New York, 1997), pp. 1–3
4. J. Rushby, Partitioning in avionics architectures: requirements, mechanisms, and assurance. Technical report, SRI International (2000)
5. RTCA/EUROCAE. ED-12C/DO-178C Software considerations in airborne systems and equipment certification. Technical report, EUROCAE (2012)
6. C.B. Watkins, R. Walter, Transitioning from federated avionics architectures to integrated modular avionics, in *2007 IEEE/AIAA 26th Digital Avionics Systems Conference* (2007)
7. G.E. Migneault, Software reliability and advanced avionics, in *Proceedings of the May 19–22, 1980, National Computer Conference* (ACM, New York, 1980), pp. 715–720
8. S. Furnell, U.G. Bleimann, P. Dowland, O. Schneider, in *Proceedings of the Eighth International Network Conference (INC 2010)*. Lulu.com (2010)
9. T.O. Group, Technical Standard for Future Airborne Capability Environment (FACE) (2017), http://www.opengroup.org/face
10. C. Jones, O. Bonsignour, *The Economics of Software Quality* (Addison-Wesley Professional, USA, 2011)
11. RTCA/EUROCAE. DO-333/ED-216 Formal Methods Supplement to DO-178C and DO-278A. Technical report, EUROCAE (2012)
12. RTCA/EUROCAE. E-218/DO-331 Model-based development and verificationsupplement to ED-12C and ED-109A. Technical report, EUROCAE (2012)
13. K. Schwaber, M. Beedle, *Agile Software Development with Scrum*, vol. 1 (Prentice Hall, Upper Saddle River, 2002)
14. K. Beck, *Extreme Programming Explained: Embrace Change* (Addison-Wesley Professional, USA, 2000)
15. A. Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams* (Pearson Education, London, 2004)
16. S.R. Palmer, M. Felsing, *A Practical Guide to Feature-Driven Development* (Pearson Education, London, 2001)
17. K. Beck, *Test-Driven Development: By Example* (Addison-Wesley Professional, USA, 2003)
18. J. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems* (Addison-Wesley, USA, 2013)
19. R.A. Kemmerer, Integrating formal methods into the development process. IEEE Softw. **7**(5), 37–50 (1990)
20. J.M. Wing, A specifier's introduction to formal methods. Computer **23**(9), 8–22 (1990)
21. S. Wolff, Scrum goes formal: Agile methods for safety-critical systems, in *Proceedings of the First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches* (IEEE Press, New York, 2012), pp. 23–29
22. H. Walters, Hybrid implementations of algebraic specifications, in *International Conference on Algebraic and Logic Programming* (Springer, Berlin, 1990), pp. 40–54
23. V. Carchiolo, M. Malgeri, G. Mangioni, Hardware/software synthesis of formal specifications in codesign of embedded systems. ACM Trans. Design Autom. Electron. Syst. (TODAES) **5**(3), 399–432 (2000)
24. J. Kasser, Model-based systems engineering: back to the future? in *Asia-Pacific Council on Systems Engineering (APCOSE) Conference* (Yokohama, Japan, 2013)

25. I. Amundson, L. Shipton, A. Liu, M. Nowak, Toward efficient model-based development of aerospace applications, in *15th AIAA Aviation Technology, Integration, and Operations Conference* (2015), p. 2741
26. D.J. Coe, J.H. Kulick, A model-based agile process for DO-178C certification, in *Proceedings of the International Conference on Software Engineering Research and Practice (SERP). The Steering Committee of The World Congress in Computer Science*. Computer Engineering and Applied Computing (WorldComp) (2013), p. 1
27. IBM. Rational Rhapsody Family (2017), http://www-03.ibm.com/software/products/en/ratirhapfami
28. IBM. Rational DOORS (2017), http://www-03.ibm.com/software/products/en/ratidoor
29. F. Paterno, *Model-Based Design and Evaluation of Interactive Applications* (Springer Science & Business Media, Berlin, 2012)
30. S. Beydeda, M. Book, V. Gruhn et al., *Model-Driven Software Development*, vol. 15 (Springer, Berlin, 2005)
31. P. Swithinbank, M. Chessell, T. Gardner, C. Griffin, J. Man, H. Wylie, L. Yusuf, *Patterns: Model-Driven Development Using IBM Rational Software Architect* (IBM, International Technical Support Organization, 2005)
32. R. Capilla, J. Bosch, K.-C. Kang et al., Systems and Software Variability Management, in *Concepts Tools and Experiences* (2013)
33. I. Groher, M. Voelter, Expressing feature-based variability in structural models, in *Workshop on Managing Variability for Software Product Lines* (Citeseer, 2007)
34. OMG, OMG Meta Object Facility (MOF) Core Specification Version 2.5.1. Technical report, Object Management Group (2016)
35. H. Gomaa, *Designing Software Product Lines with UML* (IEEE, New York, 2005)
36. K. Lee, K.C. Kang, J. Lee, Concepts and guidelines of feature modeling for product line software engineering, in *International Conference on Software Reuse* (Springer, Berlin, 2002), pp. 62–77
37. D. Beuche, Modeling and building software product lines with pure:: variants, in *Proceedings of the 16th International Software Product Line Conference*, vol. 2 (ACM, New York, 2012), pp. 255–255
38. M. Krausz, M. Zimmer, H.C. Reuss, OverNight Testing-The fully automated simulation environment for evaluation of car concepts ONT, in *Simulation Notes Europe: Ontologies in Modelling and Simulation*, vol. 2, pp. 87–94 (2014)
39. A. Haber, C. Kolassa, P. Manhart, P.M.S. Nazari, B. Rumpe, I. Schaefer, First-class variability modeling in matlab/simulink, in *Proceedings of the Seventh International Workshop on Variability Modelling of Software Intensive Systems* (ACM, New York, 2013), p. 4
40. J. Weiland, P. Manhart, A classification of modeling variability in simulink, in *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems* (ACM, New York, 2014), p. 7
41. M. Schulze, J. Weiland, D. Beuche, Automotive model-driven development and the challenge of variability, in *Proceedings of the 16th International Software Product Line Conference*, vol. 1 (ACM, New York, 2012), pp. 207–214
42. K. Pohl, G. Böckle, F.J. van Der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques* (Springer Science & Business Media, Berlin, 2005)
43. Software Product Lines (2017), http://www.sei.cmu.edu/productlines/
44. C. Dumitrescu, R. Mazo, C. Salinesi, A. Dauron, Bridging the gap between product lines and systems engineering: an experience in variability management for automotive model based systems engineering, in *Proceedings of the 17th International Software Product Line Conference* (ACM, New York, 2013), pp. 254–263
45. S. Thiel, A. Hein, Modelling and using product line variability in automotive systems. IEEE Softw. **19**(4), 66–72 (2002)
46. N. Sozen, E. Merlo, Adapting software product lines for complex certifiable avionics software, in *Proceedings of the Third International Workshop on Product Line Approaches in Software Engineering* (IEEE Press, New York, 2012), pp. 21–24

47. F. Dordowsky, W. Hipp, Adopting software product line principles to manage software variants in a complex avionics system, in *Proceedings of the 13th International Software Product Line Conference* (Carnegie Mellon University, 2009), pp. 265–274
48. F. Dordowsky, R. Bridges, H. Tschope, Implementing a software product line for a complex avionics system, in *2011 15th International Software Product Line Conference (SPLC)* (IEEE, New York, 2011), pp. 241–250
49. R.T. Braga, O. Trindade Jr, K.R. Branco, J. Lee, Incorporating certification in feature modelling of an unmanned aerial vehicle product line, in *Proceedings of the 16th International Software Product Line Conference*, vol. 1 (ACM, New York, 2012), pp. 249–258
50. J.C. Jensen, D.H. Chang, E.A. Lee, A model-based design methodology for cyber-physical systems, in *2011 7th International Wireless Communications and Mobile Computing Conference (IWCMC)* (IEEE, New York, 2011), pp. 1666–1671
51. J. Zander, I. Schieferdecker, P.J. Mosterman, *Model-Based Testing for Embedded Systems* (CRC Press, USA, 2011)
52. M. Utting, B. Legeard, *Practical Model-Based Testing: A Tools Approach* (Morgan Kaufmann, USA, 2010)
53. T. Roßner, C. Brandes, H. Goetz, M. Winter, Basiswissen Modellbasierter Test. Dpunkt. verlag, 2012
54. J. Zander-Nowicka, Model-based testing of real-time embedded systems in the automotive domain (2009)
55. Simulink Verification and Validation (2017), http://www.mathworks.com/products/simverification/
56. L.M. Boden, R.D. Busser, Adding natural relationships to Simulink models to improve automated model-based testing, in *The 23rd Digital Avionics Systems Conference, 2004. DASC 04*, vol. 2 (IEEE, New York, 2004), pp. 6–B
57. R.D. Busser, M.R. Blackburn, A.M. Nauman, T.R. Morgan, Reducing cost of high integrity systems through model-based testing, in *The 23rd Digital Avionics Systems Conference, 2004. DASC 04*, vol. 2 (IEEE, New York, 2004), pp. 6–B
58. H. Stallbaum, M. Rzepka, Toward DO-178B-compliant test models, in *2010 Workshop on Model-Driven Engineering, Verification, and Validation (MoDeVVa)* (IEEE, New York, 2010), pp. 25–30
59. B. Gallina, A. Andrews, Deriving verification-related means of compliance for a model-based testing process, in *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th* (IEEE, New York, 2016), pp. 1–6
60. U. Durak, A. Schmidt, T. Pawletta, Model-based testing objective fidelity evaluation of engineering and research flight simulators, in *Proceedings of AIAA Modeling and Simulation Technologies Conference* (Dallas/TX, USA, 2015)
61. M.M. Lehman, Programs, life cycles, and laws of software evolution. Proc. IEEE **68**(9), 1060–1076 (1980)
62. G. Visaggio, Ageing of a data-intensive legacy system: symptoms and remedies. J. Softw. Evol. Process **13**(5), 281–308 (2001)
63. D.L. Parnas, Software aging, in *Proceedings of the 16th International Conference on Software Engineering* (IEEE Computer Society Press, 1994), pp. 279–287
64. H.M. Sneed, Estimating the costs of a reengineering project, in *12th Working Conference on Reverse Engineering* (IEEE, New York, 2005), p. 9
65. V. Kshusidman, ADM transformation, in *ADM Task Force* (White Paper, 2008), http://www.omg.org/adm/ADMTransformartionv4.pdf
66. R. Pérez-Castillo, I.G.-R. De Guzman, M. Piattini, Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems. Comput. Stand. Interfaces **33**(6), 519–532 (2011)
67. C. Norton, V. Decyk, Re-engineering legacy mission scientific software, in *AIAA Space 2001 Conference and Exposition*. American Institute of Aeronautics and Astronautics (2001)
68. U. Durak, Extending the Knowledge Discovery Metamodel for architecturedriven simulation modernization. Simulation **91**(12), 1052–1067 (2015)

Part III
The Challenges

Chapter 8
Towards Autonomy and Safety for Unmanned Aircraft Systems

Christoph Torens, Johann C. Dauer and Florian Adolf

Abstract This chapter describes unmanned aircraft with respect to autonomy and safety aspects of aerospace. The focus will be on unmanned aircraft systems, however most of the principles regarding safety and automation are valid for both, manned and unmanned aviation. As a means to assure safety for aircraft, safety assessments, development processes, and software standards have been established for manned aviation. In this context, design-time assurance of software will be discussed. Another key component of the safety concept for manned aviation is the onboard pilot. The pilot supervises and validates the system behavior and develops a gut feeling if the system is okay, due to his onboard presence. This is not possible for an unmanned aircraft. Human supervision will be remotely located. Therefore, an extensive discussion on runtime assurance and automated supervision will be a part of this work. Furthermore, with the growing degrees of automation and upcoming autonomy of the aircraft, one pilot might have to supervise more than one aircraft at the same time. Unmanned aircraft are expected to be integrated into civil airspace in the near future, possibly in very large quantities. The autonomy of these unmanned aircraft and the absence of a pilot onboard the aircraft is a source of concern. However, the automation and autonomy can also support safety. The interdependence between safety and autonomy will be discussed in this chapter. The challenge regarding unmanned aircraft is that the same level of safety can be maintained. In this context, this chapter will discuss the impact of new and upcoming regulations and standards for unmanned aircraft regarding a holistic approach to the assessment of risk and their impact on autonomy and safety.

C. Torens (✉) · J. C. Dauer · F. Adolf
German Aerospace Center (DLR), Braunschweig, Germany
e-mail: christoph.torens@dlr.de

J. C. Dauer
e-mail: johann.dauer@dlr.de

F. Adolf
e-mail: florian.adolf@dlr.de

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_8

8.1 Introduction

With all the advances in flight that have been presented, one particular technology that must be discussed is the introduction of unmanned aircraft systems (UAS). Technology also advances significantly in manned aviation, however in the unmanned aircraft, the absence of a pilot onboard is a disrupting change and a challenge at the same time. Unmanned aircraft becomes possible only as a result of previous technological advances that can be built upon. In particular, reliable data links and powerful/miniaturized onboard computing are a key prerequisite for UAS.

The concept of unmanned aircraft impacts airspace as a whole, in aspects of underlying technology, operational use cases, airspace integration, and certification. Although with existing technology a high degree of automation can already be realized, there is a concern that the same level of safety will not be realized for UAS, as there is no pilot on board the aircraft. Today, one of the main responsibilities of a pilot is the continuous supervision and validation of the system behavior according to the expectations and knowledge of the pilot. Pilots may even develop a gut feeling if the aircraft is working properly, due to their onboard presence. This is not possible for an unmanned aircraft. There might be a pilot present for each UAS at a ground control station. However, his situational awareness is significantly reduced by his remote location from the aircraft alone. In more advanced scenarios, there might not even be a continuous human supervision. Or one pilot might have to supervise a whole fleet of unmanned aircraft.

This is a technological challenge, a lot of additional functionality has to be developed to enable remote operation or even autonomy. Additionally, this is a challenge, especially from a certification perspective. Today's certification concepts rely on the pilot as a safety barrier. In fact, this human element is often considered to have a failure rate of zero. With UAS, this central element of existing safety concepts is eliminated. However, the level of safety must not be reduced by the introduction of unmanned aircraft. As a result, the verification, validation, and certification of safety-critical systems of the UAS is a key element.

In this chapter, we chose the term *autonomy* being aware that it has been discussed controversially within the unmanned aircraft community. There is a tendency that capabilities under research and not yet fully understood at the time are often referred to as autonomous capabilities. While research on these subjects matures it is often replaced by *automation* as the capability is technically achieved by some form of automation. There are also approaches to standardize the concept of autonomy. The term started in the context of early flight control systems merely stabilizing the aircraft during flight. Afterward, autonomous referred to components in the context of *decision-making* for automation of missions. Currently, it is concerned with supervision and monitoring of the system itself, constantly evaluating the system's state of safety. We believe, the number of functions of unmanned aircraft will increase drastically in the near future enabling increased levels of automation. Judging from the past years, this development will occur in ways often not compatible to how safety is assured in the context of manned aviation. As accepting a lower level of safety is not an option, new methods to assess and ensure safety will have to be explored.

8.2 Challenges for Unmanned Aircraft Systems

For the integration of unmanned aircraft into segregated or non-segregated airspace and a broad use of UAS, the regulation for this new category of aircraft as well as the aircraft airworthiness is a great challenge. There is currently no finished regulation for certification and accreditation for the use of civil unmanned aircraft. The airspace is a safety-critical domain; therefore, safety is the major issue for integration of unmanned aircraft into the airspace, specifically safety targets for UAS are unclear. From this perspective, the following problems arise:

- How should UAS be regulated, and how can regulation cope with the heterogeneity of UAS?
- How can the same level of safety, trust, and situational awareness be achieved without a pilot being on board the aircraft and autonomous functions being in control?
- To which safety level do unmanned aircraft have to be designed and to which rules and standards do unmanned aircraft have to comply?

8.2.1 Challenge: Regulatory Framework for Heterogeneous UAS

One of the main barriers to the rapid growth of commercial unmanned aircraft is the missing regulation and integration into the airspace. The missing regulation is the result of a major concern for safety of people in the air and on the ground. The first attempt to establish a regulation for unmanned aircraft was started in 2005 by ICAO. In 2011 ICAO published Circular 328 [1], a document in which ICAO underlined that UAS should demonstrate equivalent levels of safety as manned aircraft and thus meet the same government rules for flight and flight equipment. By applying the same rules of manned aircraft for unmanned systems, the complex hardware and software systems that provide control of the unmanned aircraft need to address compliance with established standards such as DO-178C and DO-254 for software and hardware development.

As the term unmanned aircraft systems refer to tiny multicopters, in the same way it refers to aircraft of several tons operating above the current traffic, unmanned aircraft systems and underlying technologies are very heterogeneous. This makes it difficult for regulatory authorities to make rules about unmanned aircraft in general. Additionally, manufacturers and consortium have problems defining adequate standards and operational procedures. It is not reasonable to assume the same regulation for the multitude of unmanned aircraft, or to demand the same amount of rigor for a failsafe system across all UAS. Instead, rules should be different for manned aircraft and scalable toward the specific properties of a UAS. As a result, airworthiness standards for UAS are still in development. Some updates are briefly presented in Sect. 8.5.

Compared to manned aviation UAS offer new possibilities in use cases and mission designs. Enabling these new use cases is another challenge regarding regulations, standards, and public acceptance. From the perspective of safety assurance, the diversity poses a great challenge that is partly answered by a new classification recently published by the EASA (European Aviation Safety Agency) [2–4]. Three categories are proposed, labeled *open*, *specific*, and *certified*. UAS under the open category involves hardly any risks and there is no certification necessary. The certified category handles all UAS imposing equivalent risks as in manned aviation and requires more traditional means of certification. This class of unmanned aircraft is risk-wise located between the open and certified and contains a great diversity. A scaling of effort to ensure safety based on the actually involved risks in operation is introduced. A safety assessment method called *specific operation risk assessment* is used to determine the measures necessary to achieve an equivalent level of safety as in manned aviation. This concept will be further discussed in Sect. 8.4.

8.2.2 Challenge: Assurance of Autonomy

Autonomous UAS clearly differs from manned aircraft. There is no onboard pilot to take control in case of an emergency. The onboard pilot is replaced by the software that has to take over functions which the pilot would usually perform, resulting in software systems of increasing complexity. Specification and modeling of autonomy itself is difficult, there exist various frameworks for describing and modeling autonomy on different levels. Particularly interesting for autonomous UAS is the autonomy levels for the unmanned systems framework from NIST [5, 6] and an adaptation of this framework for UAS [7, 8]. Furthermore, the software needs to be very reliable, since the pilot cannot simply take over control. In particular, the assurance of autonomy is a topic of research. Some aspects regarding safety and risk for unmanned aircraft have been previously researched by the authors, additional information regarding failsafe systems can be found in [9, 10].

In today's aircraft, there are already a lot of supporting systems for the pilot, increasing the overall level of automation. However, the pilot is the main fallback layer in case of a system failure or any unforeseen event. The pilot can assess when a system behaves unexpectedly or in contradiction to the overall situation and then deactivate systems, switch to backup systems, or possibly do a manual control. However, this approach of a pilot on board as a fallback layer cannot be applied to the unmanned aircraft. The onboard systems must always retain a certain level of reliability and safety. And all of this critical information has to be transmitted to a ground control station. The remote pilot, who monitors one or more aircraft from the ground, is dependent on these support systems because of the spatial separation from the aircraft. Furthermore, additional functionality, supporting even higher levels of automation are required for the indirect control and monitoring of unmanned aircraft.

For achieving safe systems, in general, according to Leveson, there are four types of design techniques: hazard elimination, hazard reduction, hazard control, and damage minimization [11]. This technique should be used in the listed precedence order for a specific hazard. In the context of autonomous software systems, hazard elimination would mean to remove a software fault from the program by software verification. For this, traditional methods can be used like compliance to rigorous software development standards [12], particularly extensive software testing, but also formal methods [13] to find and eliminate the error, cf. Sect. 8.2.3. These techniques are briefly considered in Sect. 8.3, a lot of the literature can be found on this topic, however, huge efforts have to be invested to make the software safe. Hazard reduction could be implemented by a holistic risk assessment and reduction of the risk toward the environment, this concept of hazard reduction is used by the aforementioned specific category that will be discussed in more detail in Sect. 8.4. Hazard control can be considered a threat barrier in the context of the specific category concept, and will also be discussed in more detail in Sect. 8.4. Damage minimization would map to a harm barrier of the specific category concept and will be discussed in Sect. 8.3.2 as well as in Sect. 8.4.

8.2.3 Challenge: Software Verification and Certification

Due to the safety-critical nature of the domain, the well-established standard DO-178C [12] dictates development and verification activities. A key characteristic for development of aircraft functions is the target level of safety. For a manned aircraft, e.g., CS25, this is strictly defined, and since people are on board, a catastrophic failure has to be extremely improbable, which relates to a failure rate of 10^{-9}. It is yet unclear how safety levels should be interpreted for UAS, however the resulting design assurance level (DAL) significantly determines the necessary efforts to comply to standards.

For the most critical DAL, more than half of the overall prescribed activities are related to verification and validation. These standards, definitions, procedures, and risk assessments, however, have been developed for manned aircraft and it is unclear if and how existing regulation can be applied to unmanned aircraft. With undeniable similarities between manned and unmanned aircraft, there are also a number of significant differences. Most obviously, the pilot (if there is any) remains unharmed in case of a crash or failure of the unmanned aircraft. Additionally, the heterogeneous multitude of classes and sizes of unmanned aircraft open up the question of a scaled approach toward the assessment of risk.

ASTM Standard for Small UAS

Due to the high requirements for software development in aerospace, there are efforts to develop alternative standards as a means of compliance for certification purposes. ASTM developed a standard “Standard Practice for Ensuring Dependability of Software Used in Unmanned Aircraft Systems (UAS)” [14], published in 2016. The standard is intended for small UAS (25 kg) operations, where it is assumed that the risk will vary based on the concept of operations and the environment. However, it was mentioned that a use for other UAS operations in coordination with certification authorities should be possible. The scope of the standard includes safety and security aspects of software regarding UAS operation. It should be noted, that the standard can be used standalone or in combination with DO-178C. It does not replace or supersede other standards. The standard describes safety and security requirements on a very high level. The standard makes use of three requirement tiers that correspond to the design assurance levels/software levels that are known from DO-178B/C.

A direct comparison of these documents is difficult, since the level of abstraction of the ASTM standards is much higher than the descriptions of DO-178C. However, the ASTM standard for dependability of software is more extensive, since it additionally covers aspects of emergency response plans, software security, and competency and training of people involved. As an additional guidance, a table in the annex of the standard references various documents for each of for the required artifacts, e.g., the medical standard IEC 62304 as a guidance for developing the organizational software plan. The amount of documents referenced by this standard results from a comprehensive comparison of software standards from multiple domains. However, using all of the referenced documents for a particular project does not seem practical. A brief comparison between DO-178C and the ASTM standards for software dependability in Table 8.1. Specifically for small UAS however, the standard can act as a good reference for verification and certification activities.

ASTM Standard To Safely Bound Flight Behavior of UAS

In 2017, ASTM published a "Standard Practice for Methods to Safely Bound Flight Behavior of Unmanned Aircraft Systems Containing Complex Functions" [15]. This standard addresses runtime assurance to supervise specific functionality. It is not intended as a general guidance to develop safe software, but is specifically focused on enabling the use of uncertified software for safety- critical functions of the aircraft. The standard describes a reference architecture to integrate the so-called non-pedigreed software components into the system. The architecture utilizes a safety monitor that supervises relevant inputs, e.g., environmental data/sensor inputs, flight state, GPS. In normal operation, a non-pedigreed complex function is in control. In case a specified limit is violated or a defined event occurs, the architecture switches control from the complex function to a recovery control function. This recovery control function takes control and ensures that the UAS remains within predefined limits.

Table 8.1 Comparison of software assurance levels across standards

Failure condition category	RTCA DO-178C	ASTM software dependability
Catastrophic	Level A	Tier 3
Hazardous	Level B	Tier 3
Major	Level C	Tier 2
Minor	Level D	Tier 1
None	Level E	–

This standard may play an important role to introduce model concepts and methodologies for the automation and autonomy of UAS. Runtime monitoring is an active research area and the with this document a reference architecture to implement runtime monitoring is presented that can be used as a basis for discussion with certification authorities. Additional details on the use of runtime monitoring, specifically toward the assurance of autonomy is presented in Sect. 8.3.2.

8.3 Design-Time and Runtime Verification

Verification methodologies can be categorized as design-time verification and runtime verification. Design-time verification would map to the aforementioned technique of hazard elimination in the context of safety. Traditional testing methodologies, such as static tests, unit testing, software-in-the-loop testing, fall into this category. For autonomous systems, the state space of possible executions is often considered too large to be handled by traditional testing alone. As a result, in recent years, there is a growing interest regarding the use of formal methods for verification of complex systems. With DO-178C [12], the software standard directly supports the use of formal methods with a designated supplement DO-333 [13]. Formal methods can be used in various ways to support verification activities. There is a growing number of research regarding formal methods. Although there are several forms of formal methods, one particular kind will be discussed that is of particular interest regarding the assurance of autonomy, runtime monitoring.

Monitoring is the concept of supervising specific values and properties of a system. Runtime monitoring does this in parallel to the running system, in this case an aircraft in flight. For specific purposes, this is already done in several layers throughout the aircraft system as well as aircraft operations. For example, some low-level tasks exist with automated forms of monitoring, but a lot of monitoring tasks are manual. In particular, the final as well as high-level task of supervision of the flight itself is still performed by a pilot.

8.3.1 Using Runtime Monitoring for System Analysis

Runtime verification has the potential to play a major role in the development, testing, and operational control of unmanned aircraft. During development, debugging is a key activity. It should be noted, that runtime monitoring can also be utilized in the context of design-time verification in the sense that it can help to analyze and debark the system at design time. In particular, during development, log files are inspected manually to find unexpected system behaviors. However, the manual analysis quickly becomes infeasible if multiple interacting subsystems need to be considered or if complex computations have to be carried out to correlate the data. Runtime verification can automate this task and thus make debugging dramatically more efficient. During testing, runtime verification can be used to monitor the functional correctness of the system behavior. Runtime verification can be integrated into software- and hardware-in-the-loop simulations as well as into full-scale flight tests. In contrast to simple unit testing, a property that is formalized into a runtime monitor can thus not only be tested in a test fixture, but reused in all test phases.

8.3.2 *Using Runtime Monitoring for Assurance of Behavior*

Runtime monitoring can be used to supervise specific aircraft functions, as described in Sect. 8.2.3. This is particularly useful to assure and constraint the behavior of autonomous functions. The monitoring acts as an independent layer of verification for any activity of the autonomous system. To achieve this, it is necessary to formalize constraints and limitations of the autonomous function.

During operation, runtime verification can further be used to monitor the system and the validity of environment assumptions. The reason that system failures happen during a flight is often not because of implementation errors, but because unforeseen events occur and the requirement assumptions are no longer valid. Integrating runtime monitoring into operational control makes it possible to enforce safety limitations, such as constraints on the altitude, speed, geographic location, and other operational aspects that increase the risk emerging from the unmanned aircraft. If all else fails, the runtime monitor can also initiate contingency procedures and failsafes, see Fig. 8.1.

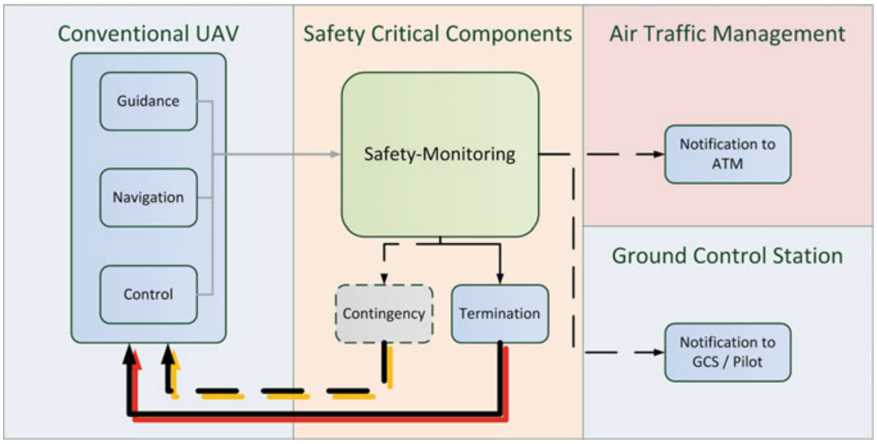


Fig. 8.1 Monitoring approach for safety assurance of UAS

8.4 Holistic Safety Approach

Unmanned aircraft systems vary in size from small multicopters to large-scale aircraft of several tons and so does the financial background between their business cases. The methodology of ensuring safety that manned aviation has worked with evolved over the past decades but is not applicable to many of these business cases due to the extensive costs of certification. Furthermore, the risks involved in operation of small-scale drones do not justify the effort of such a certification. Therefore, the need arose to ensure safety to the same degree of manned aviation but considering the actual risks involved in the operation.

8.4.1 Specific Operation Category

The specific category was designed to cover this class of unmanned aircraft. A process considering the specific operation of a certain UAS to assess the risks and determine the necessary safety measures and certification steps was developed [16]. This so-called specific operation risk assessment (SORA) considers the unmanned aircraft system itself and its mission, but also education and training of the personnel involved, the environment consisting of airspace and overflown ground area, and the background of the operator possessing the aircraft, see Fig. 8.2. From the perspective of the SORA, all these aspects and their interaction are open to discussion for each single application, which is the main reason the SORA process is considered a holistic approach by its authors.

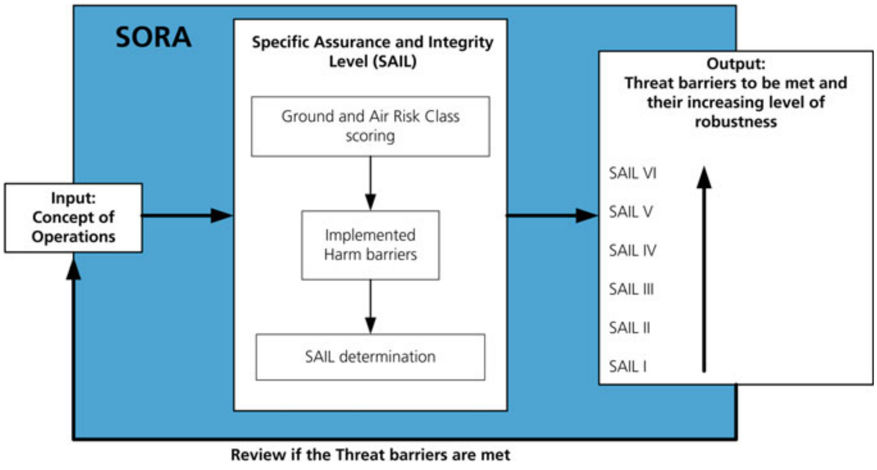

SORA
Specific Assurance and Integrity Level (SAIL)
Ground and Air Risk Class scoring
Implemented Harm barriers
SAIL determination
Input: Concept of Operations
Output: Threat barriers to be met and their increasing level of robustness
SAIL VI
SAIL V
SAIL IV
SAIL III
SAIL II
SAIL I
Review if the Threat barriers are met

Fig. 8.2 Overview of the SORA process

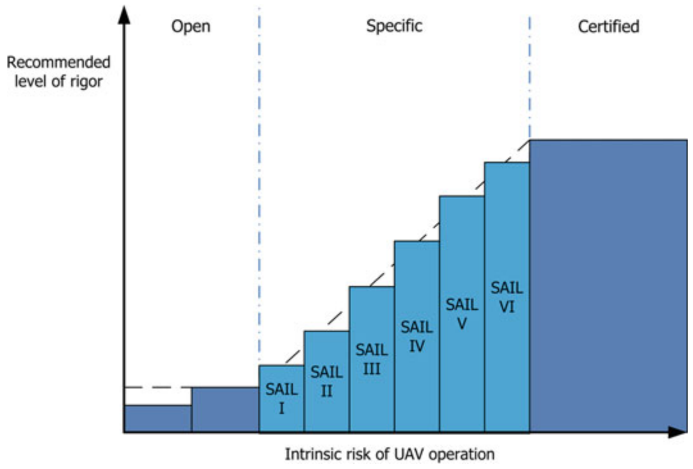


Fig. 8.3 Scaled approach toward risk using SAIL levels

All this information is compiled in one document called the concept of operation (ConOps).[1] This document represents the starting point for the SORA process, which assesses the intrinsic risks based on the aircraft and where it is operated. The amount of risk involved is reflected in a single metric, called the Safety Assurance and Integrity Level (SAIL), see Fig. 8.3. From this level, the required safety barriers[2] should be followed to achieve and guarantee safety.

In other words, the SORA provides a formalism to scale the effort to achieve safety depending on the actual risks involved in the operation and it does so for each individual application. However, two aspects are yet to be considered. First, while manned aviation looks back on many decades of experience on handling risks to ensure safety and evolved over that time period, the SORA process is created artificially and has been applied only in few cases. Specifically, the balancing of the different safety barriers and the operation involved risks will have a significant impact on safety and effort. If the SORA over- or underestimates the risks, either the certified system is insufficiently safe or the steps to prove safety is more extensive than necessary.

Second, the process of SORA is used on each application individually. As the realizations of UAS vary, so do their application and hence the results of and pitfalls in applying the SORA process. The SORA simplifies handling applications of reduced

[1]The term holistic is used in this context to express a contrast to manned aviation, where the aspects of the ConOps as defined by SORA are handled separately.

[2]In fact, the SORA differs between the so-called harm and thread barriers. While the first modifies the intrinsic risks by supporting safety even if an unmanned aircraft goes out of control, the second reduces the likelihood that it goes out of control in the first place. For simplicity reasons, we stick to the combination of both and call them safety barriers here.

risks. However, many open questions remain to handle systems of medium or higher risks that can stretch close to being equivalent to the certified category.

The process will mature with each application. One such example that might not a priori be considered within the scope of the specific category is the DLR project ALAADy [17–19]. However, as the specific category abolishes existing weight limitations, within this project aircraft of more than two tons, take off weight are studied in very low-level flight to conceptually deliver cargo in different use cases. The results of this project are used in the design of the SORA process and are used to evaluate the feasibility.

Concluding, for a class of unmanned aircraft systems, the specific category provides an alternative means to achieve safe operation compared to investing into technical reliability.

8.4.2 Key Technologies for a Risk-Based Methodology

In the past years, a significant amount of research can be found focusing on functions and capabilities of unmanned aircraft. To successfully exploit the potential of the risk-based safety argumentation, we believe that additional questions have to be answered in the near future:

Air Traffic Integration

Especially in very low-level altitudes, new concepts handling the different varieties of unmanned aircraft have to be designed. The number of potential airspace participants is huge and varying. The above argumentation will greatly benefit from new concepts to dynamically assign airspaces [20] and possibly consider the operational risks for airspace integration [19]. A crucial aspect is harmonization of detect and avoid technologies both cooperative and non-cooperative (see above).

Safe Operation Monitoring

As mentioned above onboard monitoring of the operation conditions can be used to assess the safety of a system during operation and thus reduce the effort of a priori assurance. Combining with the above risk-based argumentation, this technology can become a key enabler for the broad spectrum of technologies providing high levels of automation but being difficult to qualify. For this case of monitoring, the conditions supervised are the operation limitations used within the SORA process. Already available special cases are position-based containment systems often referred to as geofencing, cf. [21, 22]. Monitor systems can then be used to trigger contingencies once the approved operation limitations impend to be violated.

Datalinks

Datalinks are used to command and control the unmanned aircraft. The SORA enables to handle the reliability of the datalink in the context of the intended mission. This significantly reduces the overall requirements on the datalink and possibly enables alternative technologies to the so far dominating direct radio link and satellite communication infrastructures. Especially, the interplay between a safe operation monitor and automatic contingency management as independent means to ensure safe operation of the aircraft has the potential to reduce the safety requirements on the datalink and ground control station.

Risk-Based Aircraft Configurations

One of the crucial aspects of the SORA is the assessment of risks on ground [16]. Here, the ground impact of a crashing aircraft significantly influences the risk categorization. In this context, the safety characteristics of the aircraft configuration itself become performance criteria on a comparable level as classical one as flight speed, endurance, and take off and landing distances, see [18] as an example. With other words, it might in this context be preferable to use an aircraft with little ground impact than a high-performance configuration in the classical sense.

Scalable Industry Standards

Currently, standards to be applied in aviation are tools to achieve certain safety targets. A risk-based perspective allows scaling both, the effort to ensure integrity of a function but also the level of rigorousness in assurance. Corresponding industry standards will have to be developed. This concept is comparable to the design assurance levels of the ARP4754 [23] or DO-187C [12] for components of the aircraft or software. However, these levels focus on the impact a component failure has on the aircraft. Instead, the scaling in the context of SORA refers to the impact an aircraft failure has on its environment. Corresponding standards not yet exist, and existing standards are not yet allocated to the different SAIL levels. Filling this gap will, however, enable broader and safer application of UAS technology.

8.5 Certified Unmanned Aircraft Systems

Regulations for certified unmanned aircraft are still in development. However, with the growing pace of upcoming regulations and changes regarding the specific category, there are also new developments regarding traditional regulations and standards, also for manned aviation. This section will discuss new regulations regarding performance-based standards as well as upcoming efforts for regulation and standardization of the autonomous manned transporter.

8.5.1 Performance-Based Standards

Recently, the concept of performance-based standards has been introduced into certification specifications. EASA and FAA are in the process of reorganization of the current CS-23 standard. The certification specifications will be replaced by objective requirements that are designed independent. The proposed new CS-23 will also replace CS-VLA, due to the higher level of abstraction. This enables reduced efforts in changing and adapting future means of compliance. It is expected that additional standards will follow this example.

8.5.2 Urban Air Mobility

The aforementioned specific category introduces a scaled approach between open/uncertified UAS and classes of unmanned aircraft for which a full certification according to existing airworthiness standards is foreseeable. Especially for next- generation lower level airspace services—manned or unmanned—can be summarized as urban air mobility (UAM) which may not necessarily be limited to urban operation. It rather stands for a synonym of one of the most complex high-risk-type operations that needs to be supported by trustworthy autonomy as a system, i.e., in the aircraft and on the ground. With advances in sensor fusion, environment sensing, automated decision-making, and flight control technology as well as an increase of UAS platform maturity, the efforts and costs to achieve high levels of barrier implementations as well as high levels of design assurance may decline over time. The specific category may comprise a feasible path to introduce and mature new technologies that otherwise are prohibitively expensive and risky to introduce toward increasingly autonomous manned aircraft. However, at the same time, current passenger aircraft—from commercial airliners to general aviation and even ultralight sports aircraft—are getting increasingly automated or receive an increase in automated decision aids for the human pilot. As a result, a whole new kind of aviation industry along with the partnership between research organizations and universities is looking into autonomous passenger carrying aircraft (e.g., so-called air taxis) that either are built to become increasingly autonomous or even autonomous by design, i.e., a human pilot will be onboard until a certain trust into pilotless operation exists. Several companies target this potential mass market. The market is particularly dynamic for short haul connections (e.g., less than 30 min flight time) and may require vertical take off and landing (VTOL) configurations. Prominent companies in this arena at the writing of this summary are the mobility service Uber's program Elevate,[3] Airbus A^3's Vahana,[4] Volocopter,[5] Lilium,[6] Ehang.[7] Many more similar projects exist with a growing number of contributors and competition. Thus, one may look into online up-to-date resources for a more comprehensive set of current platforms and

[3]https://www.uber.com/info/elevate/, accessed 02-02-2018.

[4]https://www.airbus-sv.com/projects/1, accessed 02-02-2018.

[5]https://www.volocopter.com/en/, accessed 02-02-2018.

[6]https://lilium.com/, accessed 02-02-2018.

[7]http://www.ehang.com/ehang184/, accessed 02-02-2018.

projects, e.g., “eVTOL News”,[8] or “Dragon Flyer”.[9] The regulatory framework, for the airframe as well as its autonomy technology package, is under intensive discussion worldwide. So far, it remains unclear if novel autonomous passenger carrying aircraft are subject to general aviation, especially the reorganized Part-23 (FAA CS-23) which allows for novel performance-based rather prescriptive acceptable means of compliance (AMC). Other aircraft categories for rotorcraft (FAA CS-27) may follow toward such a regulatory reorganization. Since there are passengers on board, these types of aircraft will have very rigorous objectives and activities as their core certification requirements. How much today’s certification requirements for manned aviation are identical or even adequate for those novel aircraft is an ongoing and challenging discussion.

Besides these regulatory challenges, the evolving markets require a business case assessment in order to validate affordability, customer requirements (e.g., novel type of ride quality metrics) and most importantly external stakeholder requirements (e.g., noise, environment footprint, etc.). Autonomy and the assurance thereof can contribute feasible solutions to these challenging requirements. However, it may take decades until state- of-the-art autonomous functions proved themselves to be trustworthy enough. The stakeholders within the UAM community expect the safety targets (e.g., risk ratios, or target levels of safety) to be similar to today’s large commercial airliners, such that a projected fleet size and number of operations per day can proof probabilities of a catastrophic event in the order of $10^{-}9$ or less.

8.6 Conclusion

There are several challenges toward the introduction of unmanned aircraft into the airspace, in particular regulation, certification, safety, and the heterogeneity of UAS. Additional challenges come into play when autonomous UAS are considered, since traditional certification approaches cannot handle autonomy. At the same time, autonomy can be used to implement failsafe system behavior and thus improve the overall system safety. In this context, approaches to solve or mitigate these challenges have been discussed.

From a regulatory perspective, the new concept of the specific operation category is a framework to enable operation of UAS in a defined, restricted context in a scalable manner. The approach assures safety using a specific operation risk assessment that uses a holistic view on the combination of UAS, operation, environment, and pilot. On a technical level, runtime monitoring of the UAS plays an important role to enable analysis of complex systems, identify the system status, assurance of safe behavior. In particular, the possibility to supervise and constrain autonomous behavior is identified as a key element to establish trust into autonomous UAS, using an independent component. Additionally, a new standard regarding the safe bounding

[8]http://evtol.news/, accessed 04-02-2018.

[9]http://dragonflyer.biz/, accessed 04-02-2018.

of complex functions combines regulatory efforts with technical advances of runtime monitoring. The standard describes a framework architecture and requirements for monitoring and triggering contingency actions.

As a result, recent developments and the combined use of new regulatory approaches, new developed standards, and the concept of runtime monitoring to implement fail safety gives a good basis for the assurance of autonomy for UAS and thus possibly hint toward a timely integration of autonomous UAS into airspace. Even if the regulatory framework will be adapted, implementations, use cases, and the practical implications will be interesting to observe. As a second step of adaptation of UAS, the topic of unmanned passenger transport and air taxis has the potential to further revolutionize the aerospace domain.

References

1. ICAO. Cir 328 AN/190 unmanned aircraft systems (UAS). Technical report, ICAO (2011)
2. EASA. Concept of operation for Drones: a risk based approach to regulation of unmanned aircraft. European Aviation Safety Agency (2015)
3. EASA. Introduction of a regulatory framework for the operation of unmanned aircraft, technical opinion. European Aviation Safety Agency (2015)
4. EASA. Introduction of a regulatory framework for the operation of Drones, notice of proposed amendment, NPA 2017-5. European Aviation Safety Agency (2017)
5. H.-M. Huang, E. Messina, J. Albus, Autonomy levels for unmanned systems (ALFUS) framework - volume II: framework models version 1.0. Technical report, NIST special publication 1011-II-1.0. National Institute of Standards and Technology (NIST) (2007)
6. H.-M. Huang, Autonomy levels for unmanned systems (ALFUS) framework - volume I: terminology. Technical report, NIST special publication 1011-I-2.0. National Institute of Standards and Technology (NIST) (2008)
7. F. Kendoul, Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. J. Field Robot. **29**(2), 315–378 (2012). https://doi.org/10.1002/rob.20414. ISSN: 1556-4967
8. F. Kendoul, Towards a unified framework for UAS autonomy and technology readiness assessment (ATRA), in *Autonomous Control Systems and Vehicles*, ed. by K. Nonami, M. Kartidjo, K.-J. Yoon, A. Budiyono. Intelligent Systems, Control and Automation: Science and Engineering, vol. 65 (Springer, Japan, 2013), pp. 55–71. https://doi.org/10.1007/978-4-431-54276-6_4. ISBN: 978-4-431-54275-9
9. C. Torens, F.-M. Adolf, Fail-safe systems from a UAS guidance perspective, *Encyclopedia of Aerospace Engineering* (Wiley, New York, 2010). https://doi.org/10.1002/9780470686652.eae1147. ISBN: 9780470686652
10. C. Torens, F. Adolf, P. Faymonville, S. Schirmer, Towards intelligent system health management using runtime monitoring, in *AIAA Information Systems-AIAA Infotech @ Aerospace* (American Institute of Aeronautics and Astronautics (AIAA), 2017). https://doi.org/10.2514/6.2017-0419
11. N.G. Leveson, *Safeware: System Safety and Computers* (ACM, New York, 1995)
12. RTCA/EUROCAE. ED-12C/DO-178C software considerations in airborne systems and equipment certification. Technical report, EUROCAE (2012)
13. RTCA/EUROCAE. DO-333/ED-216 formal methods supplement to DO-178C and DO-278A. Technical report, EUROCAE (2012)
14. A. C. F. on Unmanned Aircraft Systems. F3201 - 16 standard practice for ensuring dependability of software used in unmanned aircraft systems (UAS). ASTM Subcommittee F38.01 on Airworthiness (2016)

15. A. C. F38. Designation: F3269 - 17 standard practice for methods to safely bound flight behavior of unmanned aircraft systems containing complex functions. ASTM Subcommittee F38.01 on Airworthiness (2017)
16. JARUS. Guidelines on specific operations risk assessment (SORA). Technical report, Joint Authorities for Rulemaking of Unmanned Systems (2017)
17. J.C. Dauer, S. Lorenz, J.S. Dittrich, Automated low altitude air delivery, *Deutscher Luft- und Raumfahrtkongress* (Braunschweig, Germany, 2016), pp. 1–8
18. Y. Hasan, F. Sachs, J.C. Dauer, Preliminary design study for a future unmanned cargo aircraft configuration, *Deutscher Luft- und Raumfahrtkongress* (Braunschweig, Germany, 2016)
19. N. Peinecke, A. Volkert, B. Korn, Minimum risk low altitude airspace integration for larger cargo UAS, in *Integrated Communications Navigation and Surveillance Conference (ICNS 2017)* (IEEE Press, 2017)
20. P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, J. Robinson, Unmanned aircraft system traffic management (UTM) concept of operations, in *16th AIAA Aviation Technology, Integration, and Operations Conference. AIAA 2016-3292, Washington DC* (2016)
21. E. Dill, S. Young, K. Hayhurst, Safeguard - an assured safety net technology for UAS, in *35th Digital Avionics Systems conference (DASC), Sacramento, CA* (2016)
22. C. Torens, F. Nikodem, J.C. Dauer, J.S. Dittrich, Onboard functional requirements for specific category UAS and safe operation monitoring, in *6th CEAS Conference. CEAS, Burcharest, Romania* (2017)
23. S. International. ARP4754A - Guidelines for development of civil aircraft and systems (2010)

Chapter 9
Keeping up with Real Time

Reinhard Wilhelm, Jan Reineke and Simon Wegener

Abstract This chapter is concerned with timing verification of future avionics software. We critically review a recent CAST position paper, identified as CAST-32A, about certification issues connected to the use of multi-core architectures and show that it leaves several issues unresolved. It introduces robust partitioning as a requirement for the feasibility of timing verification, but fails to precisely define it. We give a precise notion of robust partitioning that guarantees temporal isolation and, therefore, allows for separate timing analysis of tasks running on individual cores. Sometimes, complete temporal isolation is impossible to achieve or will lead to very poor resource allocation. In an ideal setting, one could analyze the timing behavior of a set of applications executed on several cores in a compositional way. We discuss the requirements for a correct analysis of the interference on the shared resources of multi-core processors. Finally, we show how to configure an existing multi-core architecture to enable compositional timing analysis.

9.1 Introduction

Many embedded control systems in airplanes are time critical, and this will remain to be so. Fly-by-wire in planes dictates hard real-time requirements for embedded control subsystems. The criticality of most of these systems requires verification as stated by international standards such as DO-178-C. The satisfaction of real-time

R. Wilhelm (✉) · S. Wegener
AbsInt Angewandte Informatik GmbH, Saarbruecken, Germany
e-mail: wilhelm@absint.com

S. Wegener
e-mail: swegener@absint.com

R. Wilhelm · J. Reineke
Saarland University, Saarbruecken, Germany
e-mail: reineke@cs.uni-saarland.de

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_9

requirements is practically verifiable, i.e., sound and efficient verification methods are available for quite powerful microprocessors, used in the safety-critical embedded systems industry.

The history of worst-case execution time (WCET) analysis started when Alan Shaw proposed *timing schemata* for the determination of bounds on the execution time of programs [1]. These use structural induction over the source program. Induction begins with execution-state independent, i.e., constant execution times for memory accesses and elementary operations. This assumption was no longer valid when high-performance microprocessors were introduced in embedded systems, which increased the average-case performance through the use of caches, pipelines, and speculation. The execution times of instructions became dependent on the execution state of the architecture, e.g., the time for a memory access became dependent on the state of the cache.

The first static timing analysis for such a complex processor was described in [2]. It used abstract interpretation to compute invariants about the execution states at all program points and exploited these invariants to safely bound the execution times of instructions [3]. This approach, instantiated for quite complex processors became the de-facto standard in industrial applications. The advent of multi-core architectures increased the complexity of static timing analysis to unacceptable orders of magnitude. Both the analysis performance was decreased and the precision of the results was reduced by the interference on shared resources [4].

The structure of this chapter is as follows: Sect. 9.2 describes and comments on a recent CAST position paper about certification issues connected to the use of multi-core architectures [5]. This position paper is shown to leave several issues unresolved. In particular, *robust partitioning* is introduced as a requirement for the feasibility of verification. However, this notion is imprecisely defined. Section 9.3 makes precise a notion of robust partitioning that would guarantee temporal isolation and, therefore, allow separate timing analysis of tasks running on individual cores. Sometimes, complete temporal isolation is impossible to achieve or will lead to very poor resource allocation. Section 9.4 discusses the requirements for a correct analysis of the interference on the shared resources of multi-core processors. Finally, Sect. 9.5 shows how to configure an existing multi-core architecture to enable compositional timing analysis.

9.2 Certification Rules for Multi-core Processors in Avionics Systems

CAST-32A [5] is a position paper about the use of multi-core processors (MCPs) in safety-critical avionics systems, which has been coordinated among representatives from certification authorities in North and South America, Europe, and Asia. It is concerned with topics that could impact safety, performance, and integrity of airborne avionics systems. In this chapter, we focus on the problems for the real-time behavior caused by using MCPs and ignore safety and integrity aspects.

The paper assumes the following development process including the verification of timing requirements: Physical properties have dictated the required reaction times of applications. These lead to what CAST-32A calls the *allocation of execution time*. Timing verification needs to show that the allocated times suffice to execute the applications. As far as possible, resources like memory, caches, time slots in a chosen bus protocol are statically allocated, and a scheduling strategy per core is chosen.

CAST-32A contains a clear analysis of the problems connected to the use of multi-core platforms (MCPs) with shared resources for the implementation of time-critical avionics applications.[1] *The contention for shared resources between applications usually causes delays in access to the resources. These delays are a form of time interference between applications, which can cause applications to take much longer to execute than when executing on their own.* This *could prevent them from having sufficient time to complete the execution of their safety-critical functionality.*

So, the execution time of an application is composed of the time, T_e, it would need if all resources were exclusively available for it and the additional delays, T_i, it encounters due to interferences by co-running applications accessing the same resources. Any design and implementation where $T_e + T_i$ exceeds the deadline violates the timing requirement and has to be revised. Timing verification, for complexity reasons, introduces uncertainty and in consequence imprecision into both time components. Hence, even if $T_e + T_i$ would not exceed the deadline (the allocated time), the bounds $\hat{T}_e$, $\hat{T}_i$ for the two times may contain so much overestimation that the satisfaction of the deadline cannot be shown. $\hat{T}_e$ is assumed to be determined by a standard single-core timing analysis. The achievable precision and the necessary effort to determine $\hat{T}_i$ heavily depend on the underlying multi-core architecture, the bus protocols, and, in general, the way resources are shared.

To ease timing verification, CAST-32A proposes *robust partitioning*, which is said to be fulfilled if both *robust resource partitioning* and *robust time partitioning* are achieved.

Robust Resource Partitioning is, unfortunately, not precisely defined, but said to be achieved when

1. *Software partitions cannot contaminate the storage areas for the code, I/O or data of other partitions.*
2. *Software partitions cannot consume more than their allocations of shared resources.*
3. *Failures of hardware unique to a software partition cannot cause adverse effects on other software partitions.*

The first and the last requirements mainly concern safety and integrity aspects. The second leaves open what it means to *consume* a resource. This requirement seems to

[1]Quotes from the paper are given in italics.

make sense for *bandwidth resources*, where it would mean that an application should not need more bandwidth than it was allocated.

Robust Time Partitioning is achieved when, as a result of mitigating the time interference between partitions hosted on different cores, no software partition consumes more than its allocation of execution time on the core(s) on which it executes, irrespective of whether partitions are executing on none of the other active cores or on all of the other active cores.

This term is no more precisely defined as it allows the *mitigation* of the time interference between partitions hosted on different cores, where unfortunately the term mitigation remains undefined. One possible interpretation of *mitigation* is that remaining sources of time interference on the hardware are avoided by how the software uses the hardware. In Sect. 9.3, we give an overview of hardware and software approaches to achieve *temporal isolation*, where *temporal isolation* corresponds to our interpretation of what *robust time partitioning* is meant to achieve.

Only if robust partitioning is achieved, the worst-case execution times (WCETs) of applications can be determined separately: *Separate determination of the WCET of an application without any other applications executing is only valid if the applicant can demonstrate that they have an MCP Platform with Robust Partitioning or that time interference from other applications is avoided or mitigated for that application.* Unfortunately, no guidance is given as to which methods should be used to obtain WCETs. We assume static analysis to be the only viable option in safety-critical domains.

If robust partitioning cannot be achieved, *the WCET should be determined by analysis and confirmed by test on the target MCP with all software components executing in the intended final configuration.* In such a case, WCET analysis needs to correctly account for interference with shared resources. In Sect. 9.4, we discuss requirements for sound timing analysis in the presence of interference, i.e., if temporal isolation is not guaranteed.

9.3 Achieving Temporal Isolation

In this section, we discuss approaches to achieve temporal isolation between applications running on different cores of a multi-core processor. As in [4], we distinguish two types of resources:

Bandwidth resources: These may not serve multiple clients at the same time, and thus simultaneous accesses lead to delays for some of the accessing tasks. Buses are the prime example for bandwidth resources.

Storage resources: Prime examples for storage resources are all forms of memories, e.g., SRAM, DRAM, and caches. Shared storage resources pose additional challenges compared to bandwidth resources: (1) The latency of some storage resources, in particular caches and DRAM is history dependent. If accesses from different clients are interleaved they affect each other's future access latencies in ways that are hard to predict. (2) Internally, most storage resources are banked, i.e., they consist of multiple smaller memories, which operate more or less independently from each other. (3) As storage resources may be used for communication they may expose data races when software is migrated from single-core to multicore architectures.

9.3.1 Bandwidth Resources

When bandwidth resources, like buses, are shared, multiple clients may attempt to access the resource at the same time and thus cause a conflict. Arbitration mechanisms are used to resolve such conflicts, granting access to only a single client at a time. Arbitration mechanisms can roughly be divided into two classes:

1. *Time-driven arbitration* follows a predefined schedule, which assigns time slots of fixed size to particular clients. This is commonly referred to as *TDMA* (time-division multiple access).
2. *Event-driven arbitration* mechanisms decide at runtime, which client to grant access to the resource. Usually, such mechanisms base their decision on the access history of all clients. Examples are *round robin* and *FCFS* (first-come-first-serve) arbitration. *Priority-based* arbitration assigns priorities to clients and grants access to the highest priority client that is currently requesting the resource.

Time-driven arbitration naturally results in temporal isolation. However, it is not work conserving and thus generally exhibits lower average-case performance than work-conserving event-driven arbitration schemes. A challenge then is to find a good static configuration of time-driven arbitration to minimize the worst-case execution times of tasks [6]. Shah et al. [7] present an arbitration scheme in which unused time slots are redistributed at runtime among other clients, thus combining the worst-case performance guarantees afforded by time-driven arbitration, while still being work conserving. This results in the loss of temporal isolation, however, and thus makes timing verification more challenging.

By nature, event-driven arbitration does not result in temporal isolation. However, in the presence of event-driven arbitration, temporal isolation can still be achieved at a higher level of abstraction: Task and execution models, such as the Predictable Execution Model (PREM) [8] and the deterministic execution model [9], have been proposed in which accesses to shared resources are exposed at the scheduling level. Then offline scheduling can guarantee that no two tasks accessing shared resources may be running simultaneously and thus cause resource conflicts.

9.3.2 Storage Resources

In this section, we consider the first two properties of storage resources, which have been mentioned above:

1. History-dependent access latencies
2. Banked memories.

We refer to [10] for a discussion of how data races may be reduced or eliminated to facilitate a transition from single-core to multi-core architectures.

Both caches and DRAM-based memories feature history-dependent timing. The contents of a cache depend on its access history: usually, data that has been requested recently is more likely to be cached than data that has not been requested recently. Different interleavings of requests from different clients may result in different access histories and thus different cache contents, which may strongly affect execution times. For example, on an Intel Core 2 Quad processor, Radojkovic et al. [11] observe a $14\times$ slowdown, compared with its execution time in isolation, when an application is run in parallel with a cache-stressing microbenchmark. Similarly, DRAM latencies are highly variable due to the presence of row buffers, which serve as a small cache, buffering the contents of the last accessed row within each DRAM bank.

To achieve temporal isolation, caches can be partitioned *spatially*. Set-associative caches can be seen as two-dimensional structures: logically, they consist of a number of cache sets, each of which consists of a number of cache lines. Physically, set-associative caches are organized as a set of independent memories, called cache ways, each of which contains one of the cache lines of each cache set. Due to this organization, some caches have hardware support for *way-based* partitioning [12–14], i.e., partitioning the cache into multiple private caches of a smaller associativity, each accessible from one of the cores of a multi-core architecture.

If no hardware support for partitioning is available, caches can also be partitioned in software [15, 16]: If virtual memory is employed and the shared cache is physically indexed, then *set-based* cache partitioning can be realized in software by *page coloring* [17]: In page coloring, physical pages mapping to the same set of cache sets are assigned the same *color*. Then, the set of colors is partitioned among the different processes. Virtual pages belonging to a particular process are only mapped to physical pages of the colors assigned to that process.

Under way- and set-based partitioning, the hit/miss behavior of applications becomes independent of co-running applications. Given a powerful interconnect between the cores and the different cache ways, way-based partitioning also eliminates possible bandwidth contention, as the distinct cache ways may service requests simultaneously. This is not the case under set-based partitioning, where accesses from different applications may still attempt to access the same cache bank simultaneously, leading to delays, which have to be accounted for in timing analysis.

DRAM and some SRAM memories are banked, i.e., they consist of multiple independent memory banks. As in way-based cache partitioning, the banks of such memories can be partitioned among different clients to enable temporal isolation. For DRAM, this has been demonstrated in both hardware [18] and software [19] solutions. Similarly, the many-core architecture Kalray MPPA-256 [20] consists of 16 clusters, which each contain 16 cores. Each of the clusters also contains 16 SRAM banks, which can be flexibly allocated to the 16 cores of the respective cluster [21].

9.4 Multi-core Timing Analysis in the Presence of Interference

It will not always be possible to achieve full temporal isolation. Also, if the utilization of a particular resource is low, partitioning such a resource for temporal isolation may not be desirable for efficiency reasons. In such cases, the remaining interference and its effect on response times need to be safely bounded.

One can distinguish at least two approaches to timing analysis for multi-core architectures:

1. In a *fully integrated timing analysis* [22–24] tasks running on different cores are simultaneously analyzed, precisely capturing all possible interleavings of resource accesses from different cores. While this approach promises the highest possible precision, it appears to be practically infeasible for realistic systems due to the enormous number of system states.
2. *Compositional timing analysis* [25–27] can be seen as a natural extension of the classical two-step approach applied routinely on single-core architectures:

    a. Low-level analysis computes the "resource demand" of each task for each shared resource. In a system with a shared bus, low-level analysis would compute a bound on the task's number of bus accesses and on its execution time on the processor core assuming the task is run in isolation.
    b. Given such task characterizations, schedulability analysis [28] then determines, whether each task can be guaranteed to meet its deadlines, accounting for the *interference* it may experience on each of the shared resources.

We believe that compositional timing analysis is the most promising approach as it does not suffer from combinatorial explosion as a fully integrated timing analysis would.

However, for contemporary multi-core architectures it has recently been shown [29] that so-called *amplifying timing anomalies* [30] render naive compositional timing analysis unsound. The same paper also proposes an analysis approach that enables compositional analysis at the cost of a more expensive low-level analysis. The other option is to employ predictable multi-core architectures that are guaranteed to enable sound compositional timing analysis by construction [31].

9.5 Case Study: Configuring a Multi-core Architecture for Timing Analysis

The Infineon AURIX TC27x [32] (Fig. 9.1) is a multi-core processor widely used in the automotive domain. It consists of two TC1.6P (performance) cores and one TC1.6E (efficiency) core. One of the performance cores and the efficiency core have attached a checker core such that they can run in lockstep mode. This lockstep mode

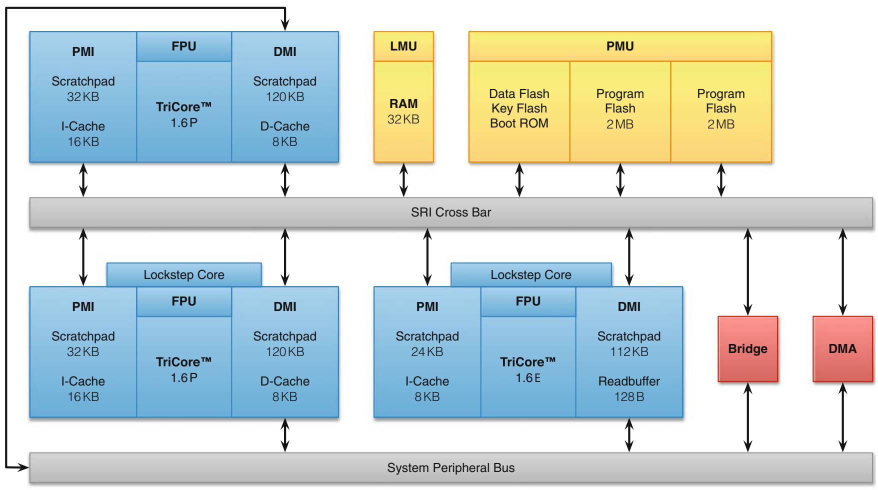


Fig. 9.1 Block diagram of the Infineon AURIX TC27x (Based on [32])

does not affect the timing behavior of the system (except in case of disagreement, then a failure is reported to the Safety Management Unit, which may trigger a handler).

We will briefly describe some of the hardware features of the AURIX TC27x that are relevant for the predictability of this architecture. Essentially these concern the avoidance or the reduction of the interference between the different cores on the shared resources. However, we will also indicate which setting of core-local features improve the precision of local WCET analysis.

9.5.1 Shared Resources

Memory and Caches

As can be seen in Fig. 9.1, each of the cores has a data scratchpad RAM and a program scratchpad RAM. These can be also accessed from the other cores.

The three cores of the AURIX TC27x are attached via the Program Memory Interface (PMI) to the SRI Cross Bar (SRI), and via the Data Memory Interface (DMI) to the SRI and the System Peripheral Bus (SPB). All peripherals are attached to the SPB except the On-Chip Debug Support (OCDS), which is connected to the SRI via the DMA interface. The Local Memory Unit (LMU) has a shared SRAM, and the Program Memory Unit (PMU) has two independent program flash memories and a data flash memory. They are attached to the cores through the SRI.

Each of the flash memories has its own port on the SRI.

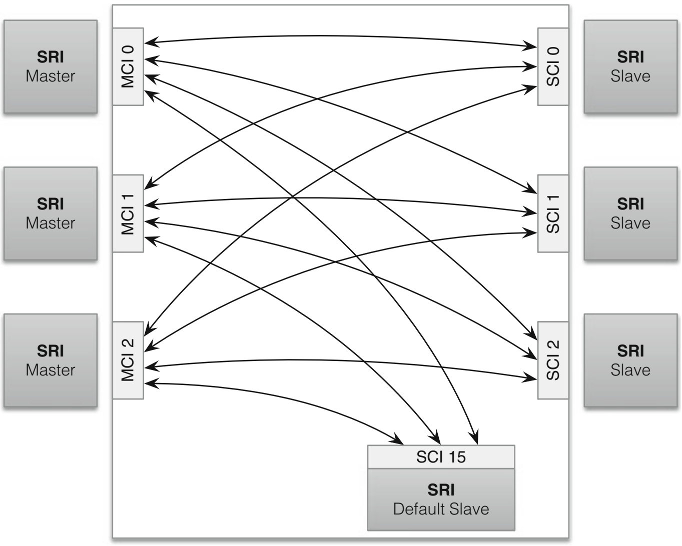


Fig. 9.2 Overview of the structure of the SRI cross bar (Based on [32])

SRI Cross Bar

The SRI (Shared Resource Interconnect) Cross Bar connects up to 16 bus masters with up to 15 slaves (and one additional default slave) via point-to-point connections (see Fig. 9.2). The default slave has two purposes. First, it handles the accesses to the SRI configuration registers. Second, the error handling concerning the SRI is done here. The three DMIs and three PMIs are master devices for the SRI, as is the DMA interface. A resource conflict happens when two or more master devices try to access the same slave device. Each slave has its own arbiter to handle these resource conflicts (see Fig. 9.3).

The arbitration rules are as follows: On the top level, the priorities of the master devices decide which request is handled first. The highest priority is 0, the lowest priority is 7. Only one master is allowed per priority, except for priorities 2 and 5. Here, an additional level of arbitration is performed. All masters with priority 2 form a round robin group, the masters with priority 5 form another one. Within these groups, round robin scheduling is done for arbitration.

Moreover, the arbiters contain a mechanism for starvation prevention. Starvation can happen if some high priority master continuously accesses the same slave such that a master with lower priority never gets its access granted. To prevent this, some kind of priority ceiling is performed when an access is not granted for a configurable timespan.

The SRI Cross Bar is well documented (about 70 pages) in the AURIX TC27x user manual [32]. It should be possible to derive the necessary formulas to predict the

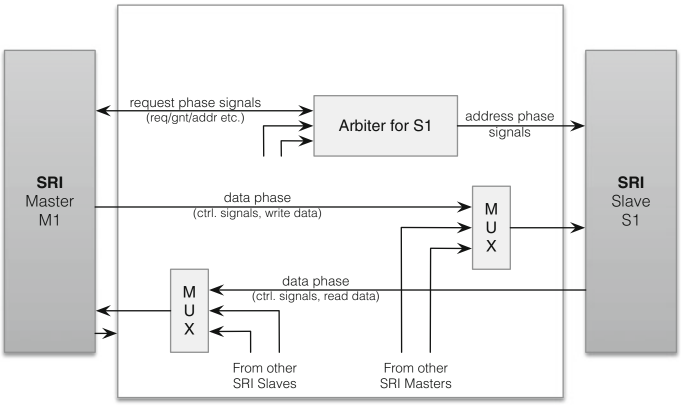


Fig. 9.3 Detailed view on one point-to-point connection in the SRI Cross Bar. Each slave has its own arbiter (Based on [32])

number of wait cycles depending on the number of conflicting accesses. However, the concrete derivation of these formulas remains future work.

9.5.2 Avoiding or Reducing Interference

In the following, we propose a configuration to minimize the amount of resource conflicts when used in a multi-core scenario:

- Let each of the performance cores use one dedicated program flash memory to avoid conflicting accesses. Let the efficiency core use the data flash if needed.
- Let the cores use the core-local data scratchpad instead of the shared RAM whenever possible to reduce conflicting accesses. In particular, place the stack into the core-local data scratchpad. If possible, have data preloaded from the shared RAM and data flash to the local scratchpad memories to control when accesses to the shared memory happens.
- Do not let the other cores access the core-local scratchpad memories.
- Let only one core access I/O channels (CAN, FlexRay, …). Assign each I/O channel in use to a specific core.

9.5.3 Single-Core WCET analysis

`aiT` supports the Infineon AURIX TC27x and can be used to compute single-core WCET bounds for each of the three cores. Only the write-back caches are not supported by `aiT`. Hence, the data caches need to be bypassed.

Moreover, the analysis model of aiT assumes that no other SRI master devices besides the analyzed core access the same slaves, i.e., that no resource conflicts happen in the SRI Cross Bar. The interference costs must be incorporated in a system-level analysis.

Branch Prediction

Both the TC1.6P and the TC1.6E cores use branch prediction mechanisms to improve the performance. The TC1.6E core uses a static and the TC1.6P core uses a dynamic branch prediction scheme. For the latter feature, a sound static analysis has to take both cases—correct prediction and misprediction—into account.

Given the fact that the TC1.6P pipeline can execute up to three instructions in one clock cycle, and its use of dynamic branch prediction, the existence of timing anomalies is likely [31]. However, we assume that the AURIXs is a compositional architecture with constant-bounded effects [33].

9.6 Conclusions

Multi-core processors with shared resources pose a severe problem for sound and precise WCET analysis. Integrated analyses of all concurrently executing, potentially interfering tasks on all cores will not scale. We argue that a compositional approach will be efficient and sufficiently precise. We also show how a popular multi-core processor can be configured to improve analyzability.

References

1. A.C. Shaw, Reasoning about time in higher-level language software. IEEE Trans. Softw. Eng. **15**(7), 875–889 (1989). https://doi.org/10.1109/32.29487
2. C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, R. Wilhelm. Reliable and precise WCET determination for a real-life processor, in *EMSOFT*, vol. 2211, LNCS 2001, pp. 469–485
3. R. Wilhelm, B. Wachter. Abstract interpretation with applications to timing validation, in *20th International Conference on Computer Aided Verification*, CAV 2008, Princeton, NJ, USA, 7–14 July 2008, Proceedings ed. by A. Gupta, S. Malik. Lecture Notes in Computer Science, vol. 5123 (Springer, Berlin, 2008), pp. 22–36. ISBN: 978-3-540-70543-7
4. A. Abel, F. Benz, J. Doerfert, B. Dörr, S. Hahn, F. Haupenthal, M. Jacobs, A.H. Moin, J. Reineke, B. Schommer, R.Wilhelm, Impact of resource sharing on performance and performance prediction: a survey, in *CONCUR 2013 - Concurrency Theory - 24th International Conference*, CONCUR 2013, Buenos Aires, Argentina, 27–30 August 2013, Proceedings. ed. by P.R. D'Argenio, H.C. Melgratti. Lecture Notes in Computer Science, vol. 8052 (Springer, Berlin, 2013), pp. 25–43, ISBN: 978-3-642-40183-1. https://doi.org/10.1007/978-3-642-40184-8
5. Certification Authorities Software Team (CAST). Position Paper CAST-32A Multi-core Processors, Nov 2016
6. J. Rosen, A. Andrei, P. Eles, Z. Peng, Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip, in *28th IEEE International Real-Time Systems Symposium* 2007, pp. 49–60

7. H. Shah, A. Raabe, A, Knoll. Priority division: A high-speed shared memory bus arbitration with bounded latency, in *Design, Automation and Test in Europe, DATE 2011*, Grenoble, France, 14–18 March 2011, pp. 1497–1500. https://doi.org/10.1109/DATE.2011.5763319
8. R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, R. Kegley, A predictable execution model for COTS-based embedded systems, in *Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium* (IEEE Computer Society, Washington, DC, USA 2011), pp. 269–279. ISBN: 978-0-7695-4344-4. https://doi.org/10.1109/RTAS.2011.33
9. F. Boniol, H. Cassé, E. Noulard, C. Pagetti, Deterministic execution model on COTS hardware, in *ARCS*, 2012, pp. 98–110
10. A. Hamann, D. Dasari, S. Kramer, M. Pressler, F. Wurst. Communication centric design in complex automotive embedded systems, in *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, ed. by M. Bertogna, vol. 76. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017, 10:1–10:20. ISBN: 978-3-95977-037-8. https://doi.org/10.4230/LIPIcs.ECRTS.2017.10. http://drops.dagstuhl.de/opus/volltexte/2017/7162
11. P. Radojković, S. Girbal, A. Grasset, E. Quiñones, S. Yehia, F.J. Cazorla, On the evaluation of the impact of shared resources in multithreaded COTS processors in time-critical environments, ACM Trans. Archit. Code Optim. (2012)
12. Y. Xie, G.H. Loh, PIPP: promotion/insertion pseudo-partitioning of multi-core shared caches, in *Proceedings of the 36th Annual International Symposium on Computer Architecture*. ISCA '09 (ACM, Austin, TX, USA, 2009), pp. 174–183, ISBN: 978-1-60558-526-0. https://doi.org/10.1145/1555754.1555778
13. K.J. Nesbit, J. Laudon, J.E. Smith, Virtual private caches, SIGARCH Comput. Archit. News **35**(2), 57–68 (2007), ISSN: 0163-5964. https://doi.org/10.1145/1273440.1250671
14. M.K. Qureshi, Y.N. Patt, Utility-based cache partitioning: a low- overhead, high-performance, runtime mechanism to partition shared caches, in *IEEE/ACM International Symposium on Micro Architecture* MICRO '06 (IEEE Computer Society, 2006), pp. 423–432
15. X. Zhang, S. Dwarkadas, K. Shen, Towards practical page coloring-based multicore cache management, in *Proceedings of the 4th ACM European Conference on Computer systems*. EuroSys '09. Nuremberg, (ACM, Germany, 2009), pp. 89–102, ISBN: 978-1-60558-482-9. https://doi.org/10.1145/1519065.1519076
16. V. Suhendra, T. Mitra, Exploring locking & partitioning for predictable shared caches on multicores, in *Proceedings of the 45th Annual Design Automation Conference*. DAC '08. Anaheim, (ACM, California 2008), pp. 300–303, ISBN: 978-1-60558-115-6. https://doi.org/10.1145/1391469.1391545
17. G. Taylor, P. Davies, M. Farmwald, The TLB slice - a low-cost highspeed address translation mechanism. SIGARCH Comput. Archit. News **18.3**a 355–363 (1990), ISSN: 0163-5964. https://doi.org/10.1145/325096.325161
18. J. Reineke, I. Liu, H.D. Patel, S. Kim, E.A. Lee, PRET DRAM controller: bank privatization for predictability and temporal isolation, in *Proceedings of the Seventh IEEE/ACM/IFIP International Conference On Hardware/ Software Co design And System Synthesis*. CODES+ISSS '11. (ACM, Taipei, Taiwan, 2011), pp. 99–108, ISBN: 978-1-4503-0715-4. https://doi.org/10.1145/2039370.2039388
19. H. Yun, R. Mancuso, Z.-P. Wu, R. Pellizzoni, PALLOC: DRAM bank- aware memory allocator for performance isolation on multicore platforms, in *Proceedings of Real-Time and Embedded Technology and Application Symp (RTAS)*, Berlin, Germany, Apr 2014
20. B.D. de Dinechin, D. van Amstel, M. Poulhi'es, G. Lager, Time-critical computing on a single-chip massively parallel processor, in *Design, Automation and Test in Europe Conference & Exhibition*, DATE 2014, Dresden, Germany, 24-28 March 2014. ed. by G. Fettweis, W. Nebel (European Design and Automation Association, 2014), pp. 1–6, ISBN: 978-3-9815370- 2-4. https://doi.org/10.7873/DATE.2014.110
21. H. Rihani, M. Moy, C. Maiza, R. I. Davis, S. Altmeyer, Response time analysis of synchronous data flow programs on a many-core processor, in *Proceedings of the 24th International Con-*

ference on Real-Time Networks and Systems, RTNS 2016, Brest, France, 19–21 Oct 2016, pp. 67–76. https://doi.org/10.1145/2997465.2997472

22. A. Gustavsson, A. Ermedahl, B. Lisper, P. Pettersson, Towards WCET analysis of multicore architectures using UPPAAL, in *WCET*, ed. By B. Lisper, vol. 15. Dagstuhl, Germany, 2010, pp. 101–112, ISBN: 978-3- 939897-21-7. https://doi.org/10.4230/OASIcs.WCET.2010.101
23. T. Kelter, P. Marwedel. Parallelism analysis: precise WCET values for complex multi-core systems, in *Formal Techniques for Safety-Critical Systems - Third International Workshop*, 2014, pp. 142–158
24. T. Kelter, WCET Analysis and Optimization for Multi-Core Real-Time Systems, PhD thesis, TU Dortmund University, 2015
25. S. Schliecker, R. Ernst, Real-time performance analysis of multiprocessor systems with shared memory, ACM Trans. Embed. Comput. Syst. **10**(2), 22:1–22:27 (2011)
26. S. Altmeyer, R.I. Davis, L.S. Indrusiak, C. Maiza, V. Nélis, J. Reineke, A generic and compositional framework for multicore response time analysis, in *RTNS*, pp. 129–138 (2015). https://doi.org/10.1145/2834848.2834862
27. W.-H. Huang, J.-J. Chen, J. Reineke, MIRROR: symmetric timing analysis for real-time tasks on multicore platforms with shared resources, in *DAC*, June 2016
28. G.C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, vol. 23, 2nd edn. (Springer, Real-Time Systems Series, 2004), ISBN 978-0-387-23137-2
29. S. Hahn, M. Jacobs, J. Reineke, Enabling compositionality for multicore timing analysis, in *Proceedings of the 24th International Conference on Real Time and Networks Systems* (2016). https://doi.org/10.1145/2997465.2997471, http://embedded.cs.uni-saarland.de/publications/EnablingCompositionalityRTNS2016.pdf
30. T. Lundqvist, P. Stenström, Timing anomalies in dynamically scheduled microprocessors, in *RTSS*, 1999, pp. 12–21. https://doi.org/10.1109/REAL.1999.818824
31. S. Hahn, J. Reineke, R. Wilhelm, Toward compact abstractions for processor pipelines, in *Correct System Design - Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday*, Proceedings. ed. by R. Meyer, A. Platzer, H. Wehrheim, Oldenburg, Germany, 8–9 September 2015, pp. 205–220
32. Infineon Technologies AG. AURIX TC27x D-Step 32-Bit Single-Chip Microcontroller User's Manual V2.2 2014–12 (2014)
33. R. Wilhelm, D. Grund, J. Reineke, M. Schlickling, M. Pister, C. Ferdinand. Memory hierarchies, pipelines, and buses for future architectures in time-critical embedded systems, IEEE Trans. CAD Integr. Circuits Syst. **28**(7) 966–978 (2009). https://doi.org/10.1109/TCAD.2009.2013287

Chapter 10
Aerospace Engineering Curricular Expansion in Information Systems

Ella M. Atkins

Abstract This chapter investigates specific approaches to evolve the Aerospace Engineering curricula to increase coverage of the fundamentals of computer science and deepen student experience in programming. First, existing K-12, Aerospace Engineering, and Computer Science and Engineering curricula are examined. Multidisciplinary programs including robotics and Cyber-Physical Systems (CPS) are reviewed to provide insight into potential directions in which an Aerospace-centric program might expand. Student, faculty, and industry interests offer insight into key Computer Science and Engineering (CSE) content to infuse into next-generation Aerospace curricula. The approach being taken at the University of Michigan, the author's home institution, is described, including plans to increase curricular flexibility and introduce a new course providing students background in key computer science concepts such as data structures and complexity, computational science with application to Aerospace analysis and design, and embedded data management and control. A discussion of potential future curricular extensions into human–machine systems and electromechanical devices concludes the chapter.

10.1 Introduction

Modern Aerospace systems integrate traditional physical vehicle structural and mechanical components with avionics, software, and people. Embedded sensors and microcontrollers have substantially reduced aircraft weight and increased reliability through local data processing, redundancy, and lightweight communication. High-bandwidth sensor data streams and advanced decision algorithms require capable onboard computers. Computers with increasingly complex software are now tasked with managing the payload, controlling vehicle motions, and supporting increasingly autonomous decision systems. Manned vehicles require software and devices to assure an onboard pilot remains situationally aware whereas unmanned vehicles must rely on communication links to remote ground or mission control stations.

E. M. Atkins (✉)
University of Michigan, Ann Arbor, MI, USA
e-mail: ematkins@umich.edu

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_10

Operator interface designs require careful consideration of real-time data stream and communication link properties as well as human factors. A twenty-first-century Aerospace system designer faces the difficult challenge of understanding both the traditional physics-based models underlying structure, aerodynamic, propulsion, and flight dynamics foundations as well as understanding the computer science foundations necessary to design, implement, and validate/verify the avionics and software supporting onboard and network-wide data-to-decision systems.

To face this challenge, the industry has staffed Aerospace systems design teams with diverse technical experts organized to collectively provide the breadth and depth of expertise needed to produce modern Aerospace systems. For example, to design a fighter or transport aircraft, teams of Aerospace engineers propose and optimize aerostructural and vehicle control designs; teams of electrical engineering propose and optimize avionics and power systems designs; teams of computer scientists propose and optimize onboard, network, and off-board software; and teams of human interface experts, including pilots, propose logical user interfaces. Engine manufacturers organize into similar subteams to produce efficient FADEC (Full Authority Digital Engine Control)-equipped products that can be hosted on a variety of airframe designs. Specialized subteams make sense so long as the subsystem designs can be developed independently, with consideration of design elements exclusively within that field of study. For example, an Aerospace engineer offering Computational Fluid Dynamics (CFD) expertise will still have sufficient background in rigid body vehicle dynamics and structures from their undergraduate studies to communicate effectively with vehicle control and structural experts. Electrical engineers will have sufficient background in analog, digital, and power systems to optimize over all avionics system elements. Computer scientists will similarly have sufficient background in logic, data structures, algorithms, and software engineering practices to work as a team despite specializations ranging from real-time embedded computing to artificial intelligence and cybersecurity. Human factors or cognitive engineering teams understand operators and ensure interfaces are informative, intuitive, and not distracting/misleading. Human-centered designs assure the "people in the system", pilots, remote operators, passengers, the overflown public, remain situationally aware and are comfortable rather than threatened.

These four subteams form the four pillars supporting the modern "Aerospace System" enterprise as shown in Fig. 10.1: (Traditional) Aerospace Engineering, Electrical Engineering, Computer Science, and Human Factors and Operations. With reference to the USA education system, Aerospace system designers enter higher education programs with a K-12 background and first-year university experience that is nearly common across the engineering fields. This foundation provides excellent coverage of traditional mathematics through Calculus and the traditional sciences, e.g., physics and chemistry. Computer science, circuits, and human–machine system exposures tend to occur through extracurricular activities in robotics or game development, requiring universities to fill knowledge gaps in these areas. Following a nearly common year 1 curriculum, university students select one of the "pillars" or majors and specialize in that area for the remainder of their degree. Project-based experiences can expose students to the challenges related to other "pillars", but a

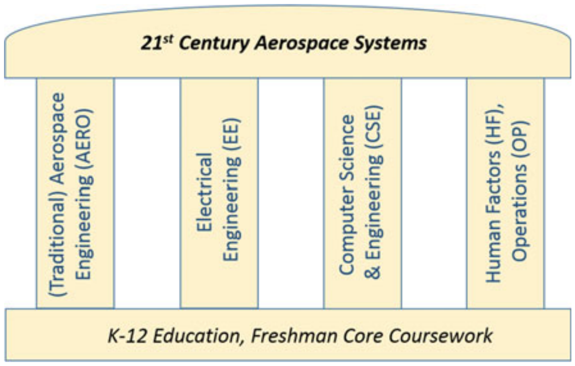

21st Century Aerospace Systems
(Traditional) Aerospace Engineering (AERO)
Electrical Engineering (EE)
Computer Science & Engineering (CSE)
Human Factors (HF), Operations (OP)
K-12 Education, Freshman Core Coursework

Fig. 10.1 Foundations of a Modern Aerospace System

minor or double major is required for a student to gain depth in any of these other pillar areas.

As Aerospace systems more tightly couple Fig. 10.1 "pillars" to achieve new levels of performance, it is clear new graduates will require a new level of crosscutting expertise. Today's entering college students increasingly recognize that future Aerospace Engineers will benefit from an improved foundation in computer science, while future Computer Scientists electing a career in Aerospace system design recognize they will benefit from improved foundation in vehicle aerostructural, electromechanical, and control systems. At the University of Michigan, we find evidence of the former in terms of the steadily increasing rates at which our Aerospace undergraduates pursue Computer Science minors or double majors. We find evidence of the latter in terms of the steadily increasing rates at which Computer Science undergraduates pursue robotics-centric coursework, research experiences, and student teams. In our growing robotics graduate program, entering students are eager to learn what they do not know. In particular, Aerospace and Mechanical Engineering graduates seek knowledge of software and autonomy, while Computer Science and Engineering graduates seek knowledge in electromechanical (robot) system design, build, and test. While robotics indeed provides multidisciplinary exposures, knowledge of the domain (e.g., Aerospace) tends to be limited, and graduates have statistically selected employment outside the Aerospace sector. It therefore remains critical for the Aerospace Systems community to work toward improved holistic education in Aerospace and Computer Science for the twenty-first-century workforce.

Researchers and educators have recognized the engineering/computer science curricular gap for more than a decade. The National Science Foundation (NSF) has established a robust Cyber-Physical Systems (CPS) program aimed at encouraging researchers in traditional engineering and computer science disciplines to closely collaborate on crosscutting research projects and to improve CPS education generally. The robotics community has also shared the vision of crosscutting expertise, placing importance on tight integration of sensors and mechanisms with capable perceive-decide-act logic and code. Independent programs dedicated to CPS and/or robotics, though still the exception rather than the rule, can offer guidance on what is possible in crosscutting curricular programs.

This paper proposes potential strategies for improving the exposure of Aerospace students to Computer Science foundations, and improving the exposure of Computer Science students to Aerospace/Robotics. Legacy curricula, degree credit limits, students' interests, and faculty expertise are all critical factors to examine in this context. Higher education curricula are already packed with required courses, so some sacrifice in traditional foundational background is required to maintain degree credit limits. Curricular additions must therefore be defined alongside methods to relax existing coursework requirements when needed. This paper is structured as follows. First, background in K-12 foundations, undergraduate Aerospace Engineering, and undergraduate Computer Science curricula is presented. Next, efforts to establish multidisciplinary programs in CPS and robotics are summarized, along with a review of open questions that must be addressed when considering options to infuse computer science into traditional engineering programs. Specific initiatives to improve computer science coverage in the University of Michigan's Aerospace Engineering curriculum are described in the context of student, industrial advisory board, and faculty input. The paper concludes with a summary of presented strategies and remaining challenges.

10.2 Current Curricula

This section first reviews typical student preparation for Aerospace Systems careers in precollege (K-12) curricular and real-world exposures. Next, traditional Aerospace Engineering (AERO) and Computer Science and Engineering (CSE) curricula are reviewed in the context of the author's current institution, the University of Michigan. The author recently published a survey of computational course requirements for Aerospace Engineering programs in the USA [1]; results indicate Michigan's current Aerospace curricula are representative as discussed further below.

10.2.1 K-12 Education

Today's youth have cell phones and access to laptop and/or desktop computers. Teens communicate fluently online and very quickly embrace new apps. Higher education courses can therefore assume a student will easily use a Windows environment, downloadable apps, and course web pages. Students will have significant experience with online search engine use and will likely find relevant YouTube and Wikipedia pages faster than faculty. Many students have had "technology" courses in which they learn to use graphics and office software packages. These courses teach navigation through menus, websites, and disk/cloud storage options. However, K-12 does not yet provide consistent coverage of computer science fundamentals, resulting in a significant gap between "usage" of an app and "understanding" of the code needed to realize that app. Because the K-12 curriculum is still struggling to establish a rigorous and consistent computer science curricular standard, K-12 educators often rely on extracurricular activities such as team-based robotics or gaming competitions to at least attract students willing to devote their spare time to such pursuits.

The current state-of-the-practice in K-12 education has the following results. First, most all K-12 graduates are comfortable with "apps" and can easily communicate and navigate using online resources. Some K-12 students have computer programming experience for activities such as robotics and gaming, but few K-12 students have knowledge of computer science foundations. In contrast, most all K-12 graduates entering a top engineering program will be well versed in math and physics, as evidenced by the fact that most undergraduates arrive at top-tier universities with Advanced Placement (AP) math and/or physics credit.

K-12 students with computer programming exposure certainly capitalize on this exposure in their improved conceptual understanding. However, because programming experiences are primarily extracurricular, no specific foundation can be presumed at a university. What language has a student used, e.g., C++, Java, Python, Basic, JavaScript? Is the student's exposure general or specialized to the particular extracurricular activity, e.g., JavaScript for Minecraft, Arduino C code loops for robots? Because even students with computer programming background in K-12 have such diverse exposures, entry-level college courses cannot assume proficiency in specific procedural or object-oriented code foundations, nor can such courses assume all students have even basic mastery of a common programming language. As a result, freshman programming courses, even within a top college of engineering, have to "start from scratch".

Researchers in K-12 Computer Science (CS) education have long studied the challenges of establishing a standard curriculum ensuring rigorous K-12 student CS preparation. Hubwieser et al. [2] provide a statistical overview of CS education programs internationally, revealing that the struggle to establish a consistent and rigorous K-12 education in CS is worldwide. This study [2] presented statistical results on CS educational terminology, curricular competencies and goals, learning content, assessment strategies, and teacher training requirements. Terminology was varied and somewhat challenging to precisely correlate, a product of the relative youth of CS in comparison to established subjects such as physics. Terminology variance led to large sets of goals and competencies, though aggregation led to "super-categories" such as "Representing, Understanding, Creating, and Testing Algorithms". In the final analysis, the following languages were taught in more than one country surveyed:

- Java,
- C++, C,
- Python,
- AppInventor,
- BASIC, VisualBasic,
- HTML,
- JavaScript, and
- Pascal.

The above list represents an impressively broad suite of languages with different features. Even when a student has had the opportunity to pursue K-12 CS coursework, this breadth in language choices illustrates a major challenge in student preparation inconsistency for entry-level engineering programming courses.

Perhaps more important than programming language is foundational concept coverage. Per Ref. [2] study, the following curricular content was covered in over half of the surveyed countries' curricula: algorithms, applications, computer and communication devices, networks, data structures, database systems, information and digitalization, CS mathematics, modeling, object-oriented concepts, operating systems, problem-solving, and programming. This is an extensive list that sets a high mark for K-12 CS educators; K-12 coverage of this set of competencies could revolutionize our ability to build toward advanced CS concepts in all engineering programs. That said, there remains a substantial gap between the ideals of [2] and the actual K-12 CS preparation observed in students entering US university engineering programs today.

To reflect how significantly the current deficiency in K-12 computer science education impacts a student's educational experience, consider for comparison the current K-12 curricula in traditional mathematics and physics. Any K-12 student aspiring to gain admission to a top-tier university engineering program is advised to take as many math and science courses as they can handle in their K-12 education, with emphasis on basic math skills, algebra, geometry, and calculus. Top students tend to complete all the math available in their high schools, with many adding courses from local community colleges in their senior year. Most high schools now offer some form of calculus, and the majority of students entering a top university earn AP exam credit to place out of at least the first-semester Calculus course. Top students in good schools also pursue AP science credit in physics, chemistry, and biology. This consistent, long-term series of K-12 exposures in math and science provides an undeniable foundation on which college coursework can directly build. For example, even a Calculus I course can assume all entering students understand basic math operations, algebra, geometry, and trigonometry. If a student is not prepared for Calculus I, it is expected the student will "catch up". This is in sharp contrast to the necessary "no student left behind" posture a first-year programming course must take, which ultimately results in a one-term programming course covering only basic conceptual material more akin to Algebra I than to Calculus. As engineering students enter their second year of an undergraduate program, most are still at a low proficiency level in computer programming. As described below, computer science programs incrementally build student knowledge as needed, whereas traditional engineering programs tend to "sweep this deficiency under the carpet" in courses emphasizing math/physics fundamentals. This deficiency can no longer be ignored in preparing the twenty-first-century student for an Aerospace Systems career, regardless of which Fig. 10.1 pillar a student selects.

10.2.2 Traditional Aerospace Engineering (AERO) Curriculum

The traditional Aerospace Engineering curriculum is built on a foundation of K-12 plus first-year engineering math, science, and humanities coursework. Over a century ago, academia recognized the need to split Aeronautics from Mechanical Engineering to enable focus on the atmosphere and flight vehicles in coursework. Researchers envisioned a time when space flight would also be possible, resulting in the creation of Aerospace as well as Aeronautics and Astronautics departments at major universities. Because of its heritage in mechanical engineering, Aerospace fundamentals were organized around topics with analogs in Mechanical Engineering: structures and materials, gas dynamics (a generalization of aerodynamics and propulsion), and vehicle flight dynamics and control. This "pillar structure" for traditional Aerospace Engineering is shown in Fig. 10.2.

Early aircraft designs focused on optimizing aerodynamics and propulsion systems. Structures have evolved from fabric and wood to metal and now to composite materials. Early aircraft hosted mechanical actuation and instrumentation systems. Aircraft and spacecraft have evolved to fly-by-wire vehicle designs, and Unmanned Aircraft Systems (UAS) are fully capitalizing on lightweight, low-cost avionics components. Increasingly-autonomous onboard data-to-decision systems are found on most mass-produced Aerospace vehicles. Still, Aerospace foundations focus on the traditional pillars, though laboratory and design project experiences have evolved to capitalize on modern hardware and software.

The current University of Michigan Aerospace Engineering curriculum has traditional content and is shown in Fig. 10.3. Note that general elective credits are not shown. The University requires the core engineering courses shown in the left column. The next category, Core Aerospace, illustrates the traditional "three-pillar" organization of lecture-based coursework. We currently require a sequence of three laboratory-based courses. Figure 10.3 groups an introductory Aerospace course, a sophomore seminar, and a senior system design course in a single category. Cur-

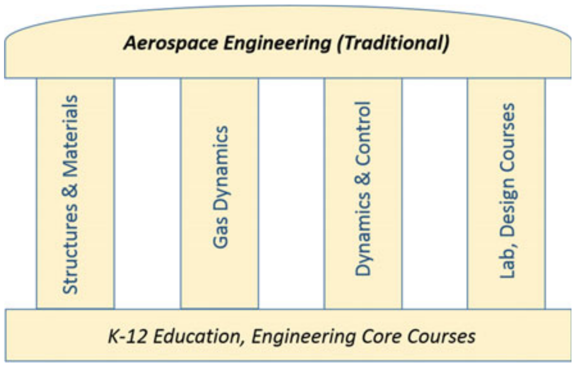

Aerospace Engineering (Traditional)
Structures & Materials
Gas Dynamics
Dynamics & Control
Lab, Design Courses
K-12 Education, Engineering Core Courses

Fig. 10.2 The traditional pillars of Aerospace Engineering

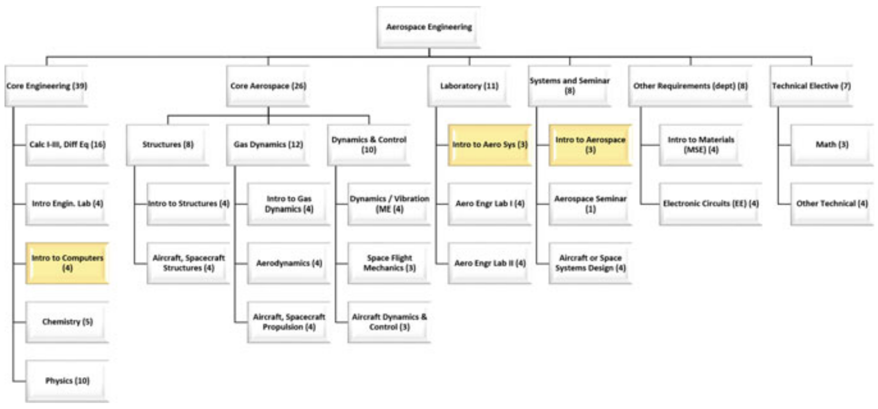


Fig. 10.3 University of Michigan Aerospace Engineering (AERO) curriculum (current)

rently, all Aerospace majors must take an introduction to materials class (taught in Materials Science and Engineering (MSE)) along with an electronic circuits class (taught by Electrical Engineering (EE)), but this is expected to evolve, as discussed later in this paper. The final Aerospace degree requirement, beyond general electives, is a Technical Elective distribution of 7 credits, 3 of which must be acquired in an advanced math course.

As shown in Fig. 10.3, there is very little required CS content in Michigan's current Aerospace Engineering degree program. Required courses containing some CS content are highlighted in the figure. A single freshman programming course introduces engineering students to CS concepts, but as discussed above, this course must cater to students across all K-12 preparation levels. Further, the course needs to cater to all engineering majors, resulting in emphasis on two languages (C++ and Matlab), which further limits coverage depth. This freshman course was never intended to cover CS theory given its main priority to ensure students have at least some programming proficiency for upper-level engineering coursework.

At the sophomore level, two required Aerospace courses require computer programming content. The "Introduction to Aerospace" course, currently taught by the author and two other Aerospace faculty interested in improving the Aerospace CS curricula, has evolved to require Matlab for every assignment and every exam. Early lectures assure students have understanding of the basic language mechanics, important especially for transfer students who have not seen Matlab previously. Assignments then progressively build on numerical methods and plotting capabilities to expose students to the use of a mathematical programming language for aerodynamic, steady flight, orbit, and launch analyses. The "Introduction to Aerospace Systems" laboratory course is structured as a custom hovercraft design, build, test experience in which students learn Computer-Aided Design (CAD), composite manufacturing, and apply basic principles of structural and aerodynamics in their iterative designs. Once students are able to successfully drive their hovercraft via radio-control link, they are asked to write specific Arduino C control code functionality. These two courses give students some sophomore-level exposure to analysis and embedded code development. These exposures are important, and they reinforce concepts and the languages covered in freshman programming. However, these exposures are not currently followed-through with any further CS content, resulting in limited maturity and confidence for students who choose not to take elective courses involving CS theory or practice.

The Aerospace Engineering graduate curriculum currently follows the same pillar structure organization depicted in Fig. 10.2. While traditional structures, gas dynamics, and control theory courses lay the traditional foundation for students pursuing degrees in this area, specific courses have been added to address computational science and real-time decision-making areas crosscutting with computer science. In gas dynamics, courses in numerical methods and Computational Fluid Dynamics (CFD) expose students to numerical methods and also computational strategies for managing high-performance computing resources needed to solve most important Aerospace flow field problems. In flight dynamics and control, courses in flight software systems and Aerospace information systems have been developed to help students gain further competence in embedded coding for autonomous control, as well as exposing students to CS theoretical underpinning they have not previously seen. The information systems course, in particular, covers concepts from data structures and complexity through finite state automata and discrete search. Our experience is that students find the theoretical underpinnings fairly easy to grasp but find coding projects much more difficult. This finding is not surprising given that students have substantial practice with pencil-and-paper problems in almost every undergraduate Aerospace course, but students have mostly used short Matlab codes to assist with Aerospace problem-solving thus are not well equipped to scale their coding experience to larger projects implemented in a lower level language such as C or C++.

10.2.3 Computer Science and Engineering (CSE) Curriculum

Computer Science and Engineering (CSE) is a relatively new discipline in comparison to Aerospace Engineering. Many universities including the University of Michigan offer two forms of computer science degrees: one within a humanities umbrella (e.g., literature, science, and arts) and another in engineering. New degrees in data and information technology have also emerged in recent decades. Because Aerospace Engineering and CSE have the most natural overlap, CSE is the CS-related degree of reference for this paper.

Figure 10.4 shows required coursework for the University of Michigan CSE program. The left column of core engineering coursework is the same as core coursework for Aerospace except that CSE students are required to take linear algebra in lieu of either Calculus III or Differential Equations. The 24-credit core com-

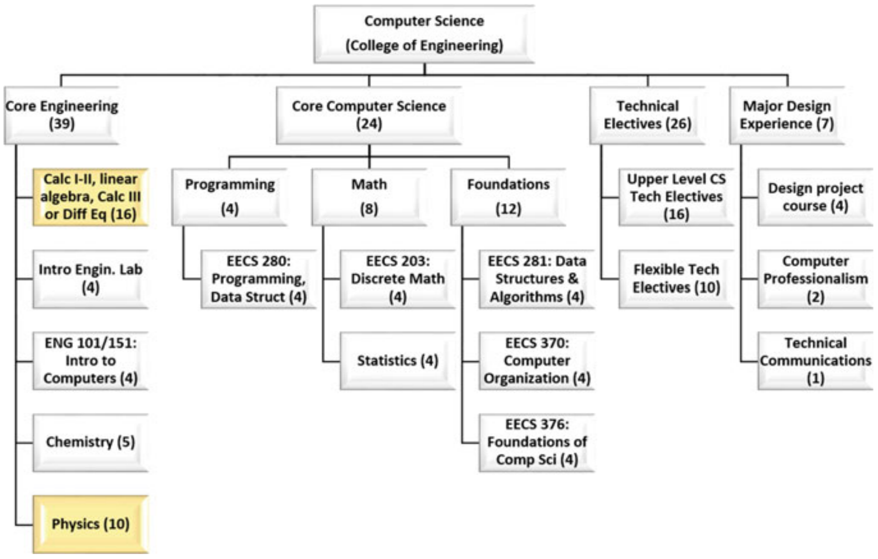

Computer Science (College of Engineering)
Core Engineering (39)
Calc I-II, linear algebra, Calc III or Diff Eq (16)
Intro Engin. Lab (4)
ENG 101/151: Intro to Computers (4)
Chemistry (5)
Physics (10)
Core Computer Science (24)
Programming (4)
EECS 280: Programming, Data Struct (4)
Math (8)
EECS 203: Discrete Math (4)
Statistics (4)
Foundations (12)
EECS 281: Data Structures & Algorithms (4)
EECS 370: Computer Organization (4)
EECS 376: Foundations of Comp Sci (4)
Technical Electives (26)
Upper Level CS Tech Electives (16)
Flexible Tech Electives (10)
Major Design Experience (7)
Design project course (4)
Computer Professionalism (2)
Technical Communications (1)

Fig. 10.4 University of Michigan Computer Science and Engineering (CSE) curriculum

puter science required for all students includes fundamental content in discrete math, object-oriented programming and data structures, algorithms, computer organization (introduction to computer architectures), and theoretical foundations. Students learn complexity theory, foundational data structures, and algorithms, and are introduced to grammars, languages, finite state automata, and Turing Machines. CSE students have substantial flexibility in selection of junior and senior level technical electives, and are required to complete a major design experience that typically includes a team-based software development effort along with a formal project reporting (technical communication) requirement.

Figure 10.5 illustrates CSE course prerequisites as a dependency graph.[1] Required sophomore and junior courses are highlighted. As shown, Discrete Math (EECS 203) and the sophomore-level programming course (EECS 280) precede the three remaining required courses (EECS 281, EECS 370, and EECS 376).[2] As shown, there are numerous technical electives available to CSE undergraduates, providing undergraduates substantial specialization opportunities. CSE has experienced remarkable growth in the past decade, so even elective classes tend to have most every classroom seat filled.

Typically, students take EECS 281 (Data Structures and Algorithms) prior to taking either junior-level course. EECS 280 and 281 in particular have lengthy coding project requirements. From the author's experience with undergraduate students,

[1]This chart is reproduced from https://www.eecs.umich.edu/eecs/undergraduate/computer-science/17_18_cs_eng.pdf.

[2]EECS refers to Electrical Engineering and Computer Science.

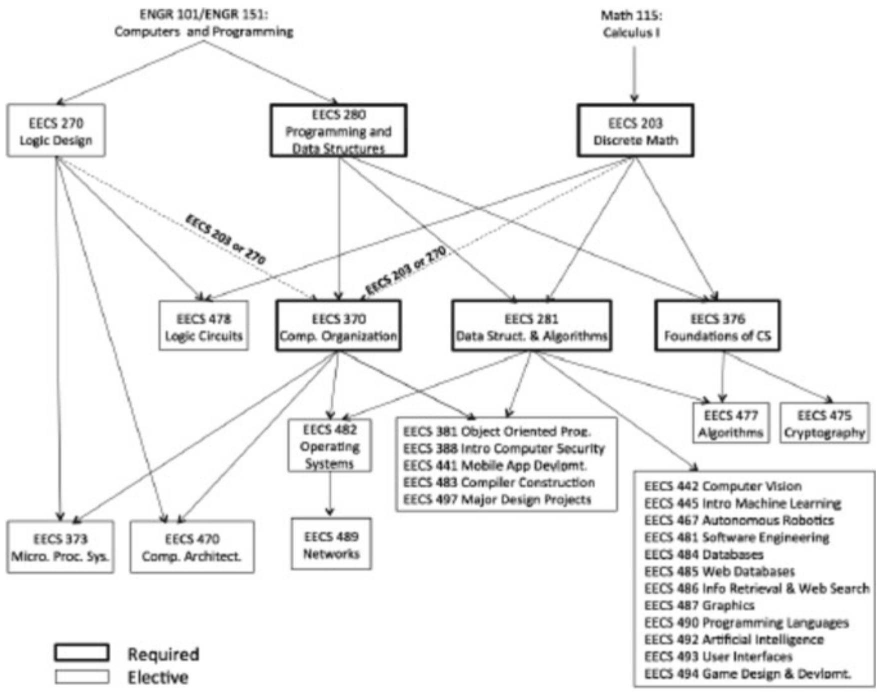

ENGR 101/ENGR 151: Computers and Programming
Math 115: Calculus I
EECS 270 Logic Design
EECS 280 Programming and Data Structures
EECS 203 Discrete Math
EECS 203 or 270
EECS 203 or 270
EECS 478 Logic Circuits
EECS 370 Comp. Organization
EECS 281 Data Struct. & Algorithms
EECS 376 Foundations of CS
EECS 482 Operating Systems
EECS 381 Object Oriented Prog.
EECS 388 Intro Computer Security
EECS 441 Mobile App Devlpmt.
EECS 483 Compiler Construction
EECS 497 Major Design Projects
EECS 477 Algorithms
EECS 475 Cryptography
EECS 442 Computer Vision
EECS 445 Intro Machine Learning
EECS 467 Autonomous Robotics
EECS 481 Software Engineering
EECS 484 Databases
EECS 485 Web Databases
EECS 486 Info Retrieval & Web Search
EECS 487 Graphics
EECS 490 Programming Languages
EECS 492 Artificial Intelligence
EECS 493 User Interfaces
EECS 494 Game Design & Devlpmt.
EECS 373 Micro. Proc. Sys.
EECS 470 Comp. Architect.
EECS 489 Networks
Required
Elective

Fig. 10.5 University of Michigan Computer Science and Engineering (CSE) prerequisite map with lists of CSE technical electives

completion of EECS 281 is the "minimum bar" for a student to have reliably achieved a strong foundation in data structures and programming. This implies that non-CSE students seeking competence in CSE will need to take a minimum of 12 credit hours of CSE coursework, EECS 203, EECS 280, and EECS 281, to gain CSE competence.

The Aerospace Engineering department currently recommends students interested in CSE pursue a CSE minor to supplement their Aerospace major. Students pursuing this minor must complete EECS 203, EECS 280, EECS 281, and one of the listed upper-level technical electives. Common choices by Aerospace students include computer vision, artificial intelligence, and autonomous robotics. The CSE minor is a good option for an undergraduate student who either can afford tuition for an extra semester or who arrives with substantial Advanced Placement (AP) credit. However, a CSE minor is not practical for most students following a standard curricular track, suggesting the need for further Aerospace curricular revision.[3]

[3]The University of Michigan Aerospace Engineering Department does not currently offer a minor degree option; if available a student selecting a CSE major and Aerospace minor would face similar course credit challenges.

10.3 Cross-Disciplinary Programs

Two multidisciplinary fields have emerged that have made progress in integrating Aerospace (AERO) and CSE foundations to some extent: Cyber-Physical Systems (CPS) and robotics. Background in each is provided below, although the author acknowledges that in practice neither have targeted Aerospace-specific systems as a focus of attention. Certainly, CPS and robotics program graduates have competence in physics, math, and CSE. However, the Aerospace industry has had trouble attracting the most talented CPS and robotics students due to the extremely favorable market for current CPS and robotics graduates. The high demand for CSE, CPS, and robotics graduates is yet another motivation for augmenting traditional Aerospace degree programs.

10.3.1 Cyber-Physical Systems (CPS)

The CPS focuses on modeling and control of sensing, mobility, information, and networking systems operating in and making decisions about a complex environment [3–5]. The nature of data, translation, and abstraction of this data, and subsequent control decisions made in CPS, depend substantially on both the application and the perspective taken by the researcher studying CPS. CPS research tends to be customized to the domain of interest, from an autonomous vehicle or cooperative vehicle team to collaborative human–robot systems. CPS decisions range from coordination and fault-tolerant control of physical actuators to real-time data management and secure networking. CPS are inherently multidisciplinary, challenging educators and practitioners to build knowledge of fundamental and application concepts that cut across, at a minimum, the fields of dynamics and control (control theory) and computer science (real-time computing).

CPS graduate degree programs have been proposed with some success. In [6], an embedded systems focus is achieved with curricular content in control, communication, distributed systems, machine learning, sensors, and security. Reference [7] emphasizes balance in theoretical and applied CPS coursework and hands-on experience to support “ready to engineer” graduates. In this work, the goal is to graduate CPS engineers with knowledge of physics, software, and systems. Specific education and experience in collaboration across multidisciplinary teams is encouraged. Impact of CPS to society should be recognized in the context of economics, human–machine interfaces, and social/legal contexts. Textbooks in CPS are beginning to be published [8]. Authors have attempted to undertake the challenge of simultaneously providing depth in CS, particularly real-time embedded systems, along with depth in mathematics and control theory for sensor data fusion and robust control.

CPS hold potential to bridge the specific Aerospace—CSE gaps related to computer-based data management and control for complex physical systems. However, most CPS research does not extend consideration to the other aspects of the

Aerospace System, e.g., aerostructural and propulsive systems as well as mission-specific challenges, particularly for space applications.

10.3.2 Robotics

Robotics is a multidisciplinary area of study that integrates principles of mechanical engineering, electrical engineering, and computer science to design, build, and deploy physical systems that assist humans or independently complete physical tasks. Coursework may be characterized in three contexts: sensing, reasoning, and acting.[4] Although plagued by comparisons to science fiction, robotic systems first experienced widespread use decades ago in factory assembly environments. Robotics has experienced rapid growth in the past decade due to affordable microelectronics, capable processors, and high-density energy storage systems. It is now possible to rapidly assemble and test gadgets that drive, fly, swim, and grasp at reasonable monetary and time investment. Robotics programs therefore can introduce students to foundational mathematics and real robotic systems through hands-on manipulator, driving, and flying systems equipped with onboard sensors, including cameras and lidar, augmented by off-board motion capture systems.

The goal of "first courses" in "mathematics for roboticists" and "robotic systems" is to prepare highly capable graduate students with diverse backgrounds for the spectrum of follow-on robotics courses they select. Today, popular robotics subjects include machine learning, machine vision, artificial intelligence and planning, linear and nonlinear control systems, signal processing, mechatronics, and navigation and mapping. New courses in nontraditional areas such as "robot ethics" are also being introduced as issues in policy, law, and culture. While robotics programs currently tend to place primary focus on manipulation, self-driving cars, and human–robot interaction, UAS or "drones" provide rich experiences for students with interest in flight, and robotics principles such as task and path planning and navigation through obstacle fields can also be directly applied to flight vehicles and planetary exploration systems. A number of high-quality robotics textbooks have been published; early texts focused on robotic manipulator kinematics, dynamics, and control. Specialized texts, online courseware, multimedia instructional materials, and open-access databases have been developed for use in robotics education and practice. Of particular relevance are open-source community-developed toolchains such as ROS, the Robot Operating System (http://www.ros.org), and cloud databases such as Open Street Map (OSM) (http://www.openstreetmap.org).

[4]These areas are used to loosely organize the robotics graduate curriculum at the University of Michigan.

10.4 Evolving the Aerospace Engineering Curriculum

This section explores opportunities to evolve the Aerospace Engineering curricula given topics in CSE, CPS, and Robotics that could improve students' expertise in twenty-first-century Aerospace Systems. Certainly, no one, including the author, would like to see key topics in the current curriculum sacrificed, but future Aerospace graduate could benefit from more than the tenuous required CS content included now.

At the core of this challenge are the following key questions:

- What aspects of computer science theory and practice are essential for the Aerospace Engineering student?
- Given credit limits, what do we remove from the required Aerospace Engineering curriculum to "make space" for new content?

The first question is beginning to be addressed through analysis of critical foundations for both Aerospace research and projects in industry. The second question has been more problematic to address, as faculty need to acknowledge that some existing required course content must become elective. Another barrier to evolution is accreditation, with organizations such as ABET (http://www.abet.org) insisting that Aerospace meet the standards required decades ago. Standards must evolve along with curricula to assure relevance in twenty-first-century Aerospace curricula.

This is not the first paper to address extension of the "traditional Aerospace pillars". In 2004, Long [9] proposed a five-pillar structure consisting of (1) fluid dynamics and thermophysics; (2) propulsion and power; (3) structural mechanics and materials; (4) guidance, control, and dynamics; and (5) computing, information, and communication. This publication appears in the inaugural issue of the AIAA Journal of Aerospace Computing, Information, and Communication, now the Journal of Aerospace Information Systems (JAIS). The first four pillars map to the three-pillar structure presented in Fig. 10.2 with fluid dynamics and propulsion both included within gas dynamics. The final or fifth pillar represented what Long considered the essential next step toward a more holistic Aerospace Engineering–CS education. In later publications, Long described a successful Aerospace software engineering course introduced at Penn State University [10], with follow-up publications such as Ref. [11]. The author feels Long's pain, in that she has spent her career pleading for Aerospace curricular reform only to face the curriculum shown in Fig. 10.2 despite over a decade of attempted reform.

Finally, over the past year, steps toward curricular reform have been seriously taken. Though small steps, there is hope at the University of Michigan, a traditional department, which means there is also hope for evolution in other traditional Aerospace programs.[5] Ultimately, the pressure to evolve has not come from tradi-

[5]The Massachusetts Institute of Technology (MIT) has embraced information systems long-term, with some growing pains as traditional faculty felt fundamentals were being sacrificed. Stanford University recently introduced an undergraduate Aerospace degree, which offered them a clean slate in which they were able to include CS content. Other universities have not yet responded; however, it is likely they are feeling similar pressures to the University of Michigan's Aerospace Department.

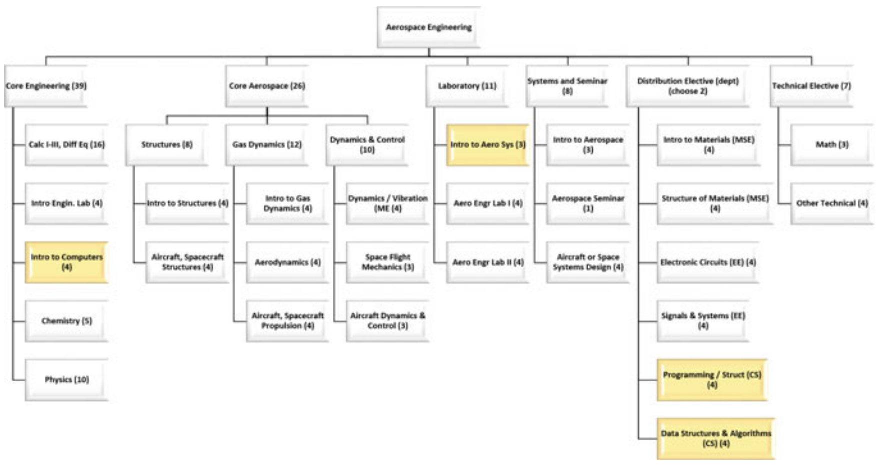

Aerospace Engineering
Core Engineering (39)
Calc I-III, Diff Eq (16)
Intro Engin. Lab (4)
Intro to Computers (4)
Chemistry (5)
Physics (10)
Core Aerospace (26)
Structures (8)
Intro to Structures (4)
Aircraft, Spacecraft Structures (4)
Gas Dynamics (12)
Intro to Gas Dynamics (4)
Aerodynamics (4)
Aircraft, Spacecraft Propulsion (4)
Dynamics & Control (10)
Dynamics / Vibration (ME (4)
Space Flight Mechanics (3)
Aircraft Dynamics & Control (3)
Laboratory (11)
Intro to Aero Sys (3)
Aero Engr Lab I (4)
Aero Engr Lab II (4)
Systems and Seminar (8)
Intro to Aerospace (3)
Aerospace Seminar (1)
Aircraft or Space Systems Design (4)
Distribution Elective (dept) (choose 2)
Intro to Materials (MSE) (4)
Structure of Materials (MSE) (4)
Electronic Circuits (EE) (4)
Signals & Systems (EE) (4)
Programming / Struct (CS) (4)
Data Structures & Algorithms (CS) (4)
Technical Elective (7)
Math (3)
Other Technical (4)

Fig. 10.6 Proposed distribution elective for the University of Michigan's Aerospace Engineering program

tional Aerospace faculty, nor has it come from the Aerospace industry. It instead has come from students, who have increasingly sought CS minors and who have begun to, in some cases, abandon Aerospace as a major due to the challenge of fitting both AERO and CS into a 4-year program of study. A committee of six faculties covering Aerospace autonomy, validation and verification, space systems, and computational fluid dynamics has engaged in proposing specific computing curriculum reforms. To date, there has been little appetite for adding Aerospace courses that fully mirror CS courses, but the faculty has approved a curriculum change that will increase student flexibility, and a new course is being developed to address what are viewed as key gaps in Aerospace student CS background.

Figure 10.6 illustrates a welcome curricular change recently approved by the faculty. Specifically, the two required extra-departmental courses, one in materials and one in electrical circuits, have morphed into a "Distribution Elective" to increase student flexibility. In the revised curriculum, students can select any two of the six listed courses, two from Materials Science and Engineering (MSE), two from Electrical Engineering (EE), and two from CSE. This change will make it easier for Aerospace students to elect a CSE minor in that at least one CSE minor courses, EECS 280 or EECS 281, can also count toward the Aerospace degree. Note that students opting for EECS 281 will have to complete prerequisite EECS 203, but this one additional course will not prevent most students from selecting the CSE distribution elective given interest.

With the modified curriculum, Aerospace students will have the opportunity to deepen their exposures in one or more out-of-department course sequences. However, students who do not elect the EECS 280-281 sequence will graduate with no better CS background than they obtain today. To address this problem, a group of Aerospace faculty are developing a new 4-credit course with a computer laboratory component that assures one-on-one oversight. The course will be comprised of three CS-centric modules with acknowledged connection to Aerospace: CS fundamentals including data structures, iterative and recursive algorithms, and computational complexity, Computational Science including key numerical methods and efficient memory and computational approaches, and embedded real-time data management and control with a hardware-based final project (quadcopter operating indoors and outdoors in a netted facility). As a junior course, students will continue to build on their Matlab analysis experiences and C/C++ (Arduino) experiences from sophomore year. The course will offer a combination of new materials and repeat exposures to key CSE concepts students might not remember long-term without this new required exposure. We envision a team of two faculties, one with expertise in embedded systems and one with expertise in computational science, to teach the course until clear notes and baseline projects are established. One in place, students will have required course-work exposures to CSE topics at freshman, sophomore, and junior levels. Senior electives in numerical methods and flight software systems will offer students the option of continuing their CSE studies in Aerospace-centric analysis and embedded control.

10.5 Conclusion and Discussion

This chapter has summarized state-of-the-practice in K-12, Aerospace Engineering, and CSE education. Emerging CPS and robotics disciplines were reviewed to offer ideas for extending the traditional Aerospace curriculum to better incorporate the breadth of Aerospace Systems topics, particular computer CSE. CSE augmentations to the University of Michigan's Aerospace program are proposed and discussed.

Exposure of Aerospace students to CSE concepts now appears within reach; however, there are two remaining pillars underlying the twenty-first-century "Aerospace System": electrical engineering and human factors/operations. While consideration of fundamentals for these areas is beyond the scope of this paper, it will be important for Aerospace and CSE programs to continue evolving. Otherwise the gap between "operators" and "engineers" will continue to widen, and Aerospace Engineers might eventually be as unaware of how analog and digital circuitry work as they are of computer science foundations today.

References

1. E.M. Atkins, Education in the crosscutting sciences of aerospace and computing, J. Aerosp. Inf. Syst. **11**(10), 726–737 (2014)
2. P. Hubwieser, M.N. Giannakos, M. Berges, T. Brinda, I. Diethelm, J. Magenheim, Y. Pal, J. Jackova, E. Jasute, A global snapshot of computer science education in K-12 schools, in *Proceedings of the 2015 ITiCSE on Working Group Reports*, ACM, 2015, pp. 65–83

3. R.R. Rajkumar, I. Lee, L. Sha, J. Stankovic, Cyber-physical systems: the next computing revolution, in *Proceedings of the 47th Design Automation Conference*, ACM, 2010, pp. 731–736
4. E.M. Atkins, J.M. Bradley, Aerospace cyber-physical systems education, in *Infotech@ Aerospace (I@ A) Conference*. (American Institute of Aeronautics and Astronautics, USA, 2013), p. 4809
5. E.M. Atkins, Education in the crosscutting sciences of aerospace and computing. J. Aerosp. Inf. Syst. (2014)
6. F. Kurdahi, M.A.A. Faruque, D. Gajski, A. Eltawil, A case study to develop a graduate-level degree program in embedded and cyber-physical systems, ACM SIGBED Rev. **14**(1), 16–21 (2017)
7. M. Törngren, M.E. Grimheden, J. Gustafsson, W. Birk, Strategies and considerations in shaping cyber-physical systems education, ACM SIGBED Rev. **14**(1), 53–60 (2017)
8. E.A. Lee, S.A. Seshia, *Introduction to Embedded Systems: A Cyberphysical Systems Approach* (MIT Press, USA, 2016)
9. L.N. Long, Computing, information, and communication: the fifth pillar of aerospace engineering, J. Aerosp. Comput. Inf. Commun. **1**(1), 1–4 (2004)
10. L.N. Long, O. Janrathitikarn. A new software engineering course for undergraduate and graduate students, in *AIAA Infotech@ Aerospace Conference* 2010, pp. 20–22
11. L.N. Long, On the need for significant reform in university education, especially in aerospace engineering, in *Aerospace Conference* (IEEE, New York, 2015), pp. 1–7

Index