


## Article

# A Modeling Method of Agents and SOA in Advanced Avionics System Based on AADL

Pingyu Deng <sup>1,2</sup>, Qing Zhou <sup>2</sup>, Dong An <sup>3,\*</sup>, Shihai Wang <sup>3</sup>  and Kui Li <sup>2</sup><sup>1</sup> School of Aeronautics and Astronautics, Shanghai Jiao Tong University, Shanghai 200240, China<sup>2</sup> Science and Technology on Avionics Integration Laboratory, Shanghai 200233, China<sup>3</sup> School of Reliability and Systems Engineering, Beihang University, Beijing 100083, China

\* Correspondence: andong@buaa.edu.cn; Tel.: +86-10-82313598

**Abstract:** The modeling method of agents and service-oriented architecture (SOA) in avionics systems describes agents and SOA in avionics systems with models. To our knowledge, however, the current modeling methods cannot describe the behavior of agents and SOA accurately and do not fit well with the existing avionics system models. This paper addresses the above problems by presenting a modeling method based on architecture analysis and design language (AADL). In this method, the working states of agents are described by the mode components, with the working process being triggered by the input of agents; and the services are described by the process component. The application of the software system is described by the system components that contain several process components. Moreover, different modes of the system are used to describe different applications, and the transitions of application are triggered by specific application requests. Software architecture of an avionics system is modeled by the proposed method. This case demonstrates that the proposed method can accurately describe how agents and SOA work in a new way and fit well with the existing avionics system models.

**Keywords:** agent; service-oriented architecture (SOA); architecture analysis and design language (AADL); avionics system; modeling method



**Citation:** Deng, P.; Zhou, Q.; An, D.; Wang, S.; Li, K. A Modeling Method of Agents and SOA in Advanced Avionics System Based on AADL. *Appl. Sci.* **2022**, *12*, 8157. <https://doi.org/10.3390/app12168157>

Academic Editors: Erika Ottaviano, Jose Machado, Katarzyna Antosz, Dariusz Mazurkiewicz, Yi Ren, Pierluigi Rea, Rochdi El Abdi, Marina Ranga, Vijaya Kumar Manupati and Emilia Villani

Received: 8 June 2022

Accepted: 12 August 2022

Published: 15 August 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

With the advancement of aviation technology, artificial intelligence and information technology have become the focus of aviation research. Yang Wei, academician of Chinese Academy of Sciences, and Sun Cong, academician of Chinese Academy of Engineering, highlighted the importance of information and intelligence technologies in their discussions on the development of future fighter jets [1,2]. As an indispensable part of informatization, software architecture is the important foundation of avionics systems and artificial intelligence technology in avionics systems. The application of artificial intelligence technology and service-oriented architecture (SOA) in avionics systems has become the focus of avionics architecture. Expert systems that contain intelligent agents have been applied to improve the reliability and maintainability of avionics systems [3]. A new generation of airborne human-computer interaction systems has been developed based on multi-agent systems to enhance situation awareness and decision-making through the pilot cockpit [4]. Furthermore, recent studies have revealed that the SOA can help an avionics system not only improve its flexibility but also reduce the cost and human-computer interaction of updating features [5–7]. Likewise, the SOA has been applied to avionics systems [8–12].

However, with the increase in the scale of avionics systems, system design becomes increasingly complex; thus, modeling has become the focus of design work. Improvement of the modeling method of agents and SOA is the focus of current research. This paper proposed a novel modeling method of agents and SOA in avionics systems.

Agents in avionics systems can interact with the external environment autonomously. When the input changes, the working state of the agent will also change, and many attributes of the agent will also change under these states. In addition, SOA encapsulates functional units as services, and deploys and reuses them, which brings higher flexibility of software architecture and convenience of development. However, when SOA executes different applications, the difference of services invoked will lead to the change in system attributes. Such changes in the attributes of agents and SOA will bring difficulties in describing many of their characteristics, such as the task time delay, task reliability, processor scheduling policy, and so on. Traditional avionics system modeling methods lack accurate description of these changes and characteristics of agents and SOA; such inaccuracies in modeling will bring great difficulties to the analysis and design of the system, which may lead to software system failures and even serious accidents and dangers.

As an integrated system of software and hardware, the modeling method of avionics system needs to consider the modeling of integrated systems of software and hardware. Architecture analysis and design language (AADL), a representation method, is developed for designing and analyzing software and hardware architecture of an embedded real time system and its functional and non-functional properties [13–15]. This modeling language mainly focuses on the binding relationship and interaction between system components, and adequately describes the complex real time embedded system architecture.

However, there are some issues to consider in the modeling work. First, the working state of the agent changes with the input and SOA is characterized by architectural flexibility and not only the encapsulation but also reuse of functions. These require a modeling method to effectively support the dynamic behavior of the system. Secondly, AADL language is suitable for modeling avionics systems, but the model of software architecture in existing avionics system modeling methods lacks dynamic behavior description. The mode component in AADL is suitable for describing dynamic behavior. Therefore, the combination of a traditional software architecture modeling method and a dynamic behavior description method, based on pattern components, provides a more accurate description of the dynamic behavior of the agent and SOA. At the same time, this model has very good compatibility with the existing avionics models based on AADL, so that the same model can describe the common software architecture and the agent and SOA at the same time, and accurately describe these complex features of the agent and SOA.

Based on the above content, this paper proposes a novel modeling method of agent and SOA based on AADL, aiming to improve the ability of model to describe agent and SOA by describing the dynamic behavior and attributes of agents and SOA in different states. Firstly, based on the change in the agent working state, the environmental input feedback and multi working states modeling method of the agent are proposed, aiming to describe the transition mechanism of the working state of the agent with external input and the attributes of the agent in different working states. In addition, considering the service-oriented encapsulation of SOA and the invocation of services by applications, an application and service modeling method of SOA is proposed, aiming to describe the encapsulation and reuse of services in SOA and the flexibility of software architecture, when executing different applications. Moreover, this modeling method is used to model an avionics system, including SOA and agents. Based on this model, the delay time of different agent working states and SOA execution of different applications are analyzed. The analysis results show that the system has different delay times under different inputs and different application execution, which accurately describes the attributes of agents in different states and SOA execution of different applications and proves the effectiveness of this method in practice.

In the remainder of this paper, Section 2 briefly reviews the agents, SOA, and AADL, and Section 3 presents the agents and SOA modeling method based on AADL. In Section 4, a case study is introduced and the last section provides conclusions and future work.

## 2. Background

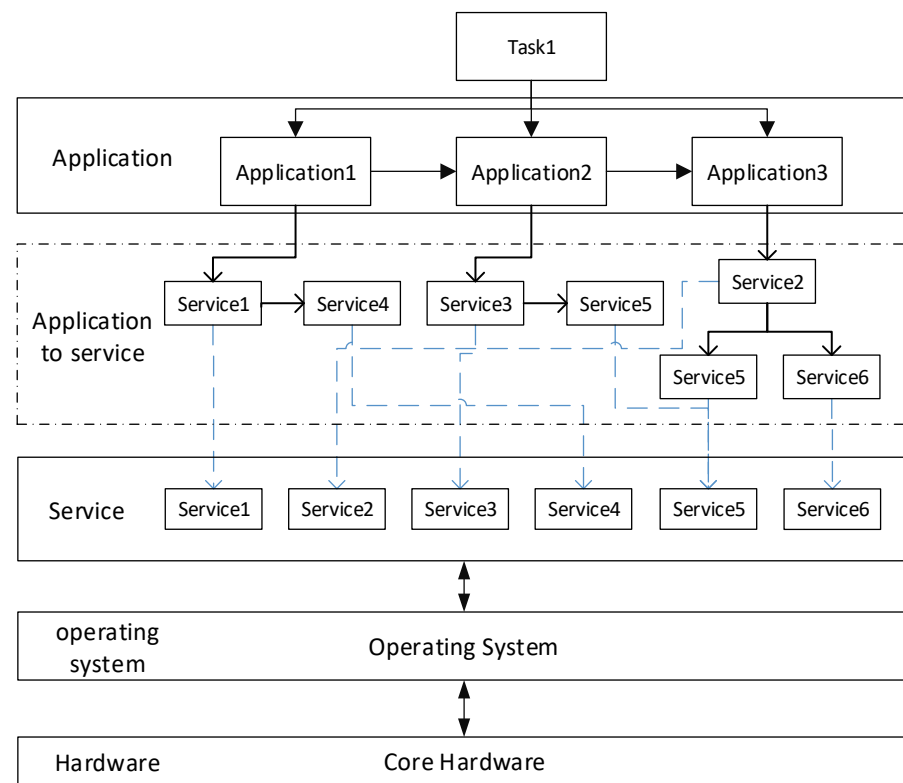
This section briefly reviews the research and modeling methods for agents and SOA, and an introduction to AADL. Moreover, disadvantages with the current modeling methods for avionics systems are discussed.

### 2.1. Research and Modeling Methods for Agents

Maes defined an agent as a software entity that can carry out a group of operations on behalf of users or plans with certain independence or autonomy, and feel and automatically perform tasks in the computing system environment of complex dynamic environments [16]. Jennings pointed out that an agent is a computer (hardware or software) system that meets specific design needs, and it is located in a specific environment with a high degree of flexibility and autonomy [17,18]. For example, the agents in avionics systems can autonomously identify targets and plan routes. At present, agents have been used in avionics systems, and there are many studies on modeling methods for agents. Agents are divided into typical agents according to functions and typical states based on the discrete event system (DEVS) [19,20]. An agent can also be defined as a mapping, which maps the local environment perceived by an agent into general behavior attempts rather than specific actions [21]. The adaptive avionics system software based on agents has been applied in avionics fault diagnosis and air traffic systems [22,23]. Adaptive software in avionics systems can be modeled by using multi-agent systems based on the belief, desire, and intention (BDI) model or hybrid automata [24–26]. In addition, the formal verification of agents based on the BDI model also has some achievements [27]. The abstract state machine (ASM) is used to model the agents in avionics systems, and based on this method, an intelligent landing gear system was modeled [28]. Agents have been applied to model and simulate pilot behavior, which means that agents have great potential in the aviation field [29].

### 2.2. Research and Modeling Methods for SOA

Up to now, there is still not a definition of SOA that is universally accepted, but some are popular in the industry. According to Erl, SOA represents an open, agile, extensible, federated, and composable architecture, consisting of self-contained, high-quality, interoperable, discoverable, and potentially reusable services [30]. In our previous work, the concept of service-oriented advanced avionics architecture was put forward, and service components and management framework of SOA in avionics systems were defined [31]. SOA working logic in avionics is shown in Figure 1. The bottom two layers are the operating system and hardware of the avionics system, which provide hardware resources and basic system functions for the whole system. The next layer is the service layer, which is oriented to user application requirements and encapsulates business logic into services with explicit functions, which are provided to applications and other services for invocation. Above the service layer is the application layer, which is the business logic entity that invokes various services for different task requirements. Between the application layer and the service layer is the invocation relationship from the application to the service and between the services. These invocation relationships represent how the application invokes the service to achieve the task requirements. Above the application layer is the task layer. The tasks accomplish the system goals through the invocation of applications, the invocation of applications to services, and the management and operation of system resources by services.

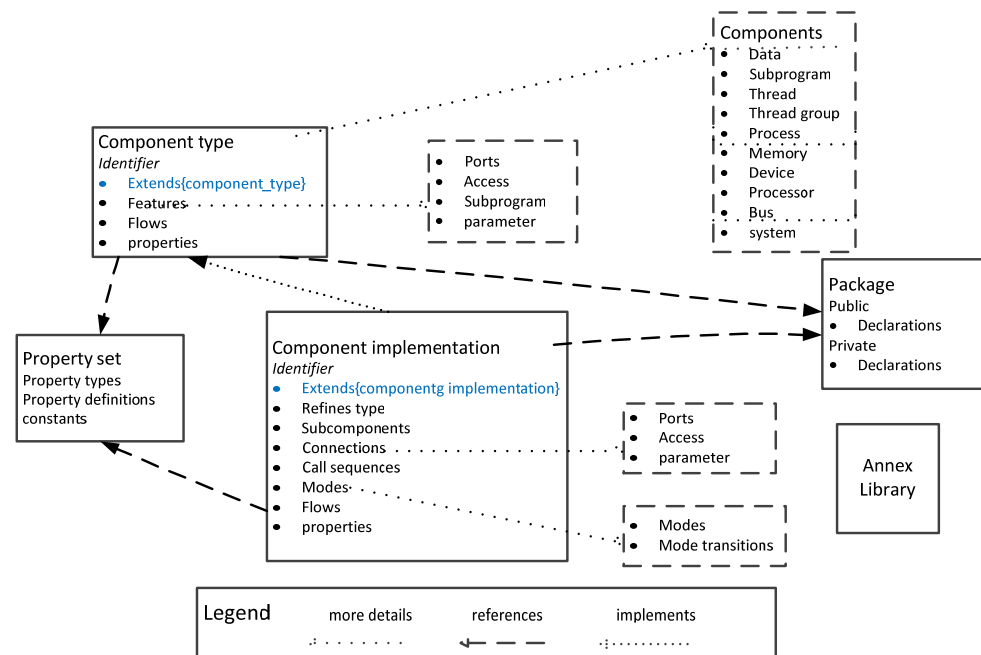


**Figure 1.** SOA working logic in avionics system.

In the current SOA research and project practice, some methods, such as object-oriented analysis and design method (OOAD), and business process modeling (BPM) methodology have been applied to SOA modeling. In SOA modeling, service choreography and other contents should be paid attention to on the basis of software architecture, and many design principles in OOAD are also carried out by SOA inheritance [32], but OOAD applied in SOA modeling mainly abstracts the system from the class and object instance levels, and lacks the description of the overall business architecture, when confined in a single application. In the process of modeling SOA with BPM, the core is the management of the process, and the essence is to take the process as new abstract data [33]. Zdun et al., proposed a process-driven modeling approach that focuses on SOA modeling in the context of business processes [34]. In this method, although the process management of the work unit is realized, the description of the business implementation and architecture in the system is missing.

### 2.3. Introduction of AADL

AADL is a design language defined by the standard AS5506 of the Society of Automotive Engineers. It is highly effective for model-based analysis and specification of complex real time embedded systems. The application software includes a thread, thread group, process, and subprogram. The execution platform includes a processor, memory, device and bus [15]. AADL describes the software and hardware architecture of the system through the concepts of component and connection, and models with components. In the AADL model, the component is the core element and the basis of modeling. The elements of the AADL architecture are shown in Figure 2 [15], and the language elements of AADL are divided into different parts around the components, such as component type, components, component implementation, etc. On this basis, the detailed information of the elements is further described by the relationship between the elements.



**Figure 2.** AADL language elements.

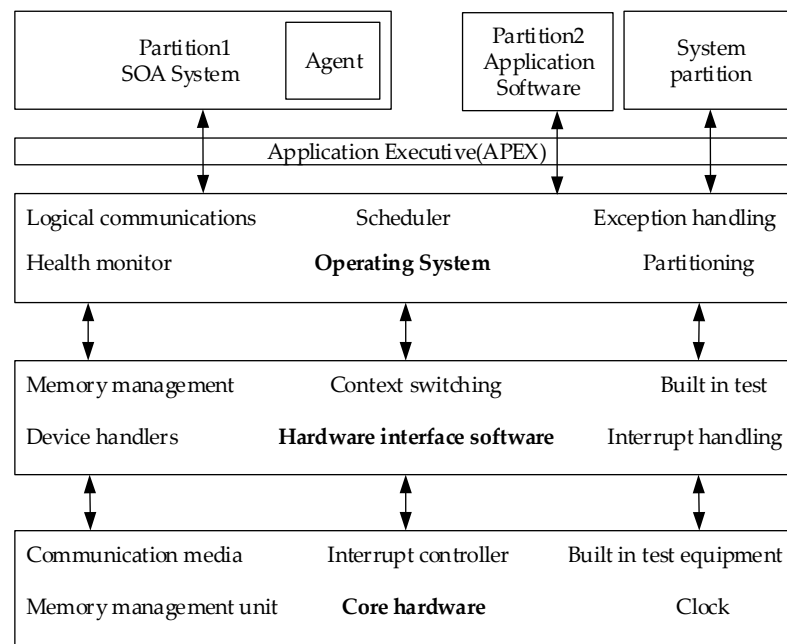
Bertrand proposes that modes can be used to represent various states of a system or component in AADL and can also represent component configurations and connections in the system. Furthermore, a system or component can transit from one mode to another when a set condition is triggered [35]. The property of a system or component can have different values in different modes, and mode transitions can be triggered by events, data, or messages [15,35]. Based on these characteristics, it can be observed that the mode is suitable for describing the working state of a system or component. In different modes, different values of the same attribute can express the characteristics of the system or component in different working states; a set of events triggers a mode transition that describes a change in a working state.

#### 2.4. Disadvantages of Current Avionics System Modeling Method

Traditional software modeling methods that are currently applied for avionics systems lack specificity to the agent and SOA. Although AADL describes traditional software and architecture in avionics system well, as it includes data, subroutines, threads, thread groups, processes and other components, there is still difficulty in describing the dynamic behavior and attribute change in the agent and SOA in avionics systems. The existing software modeling methods of avionics systems have the following disadvantages:

- (1) They cannot accurately describe the transition mechanism of the working state of the agent with external input or the attributes of the agent in different working states;
- (2) They cannot accurately describe the encapsulation and reuse of services in SOA or the flexibility of software architecture when executing different applications.

Figure 3 shows the avionics architecture with agents and SOA. The avionics system architecture is divided into several parts from top to bottom. The lowest layer is the core hardware layer, which is the core hardware equipment of the avionics system, such as communication media and interrupt controller. The upper layer is the hardware interface software layer, which is the interface software of hardware devices, such as memory management and context switching. Next, there is the operating system layer, which is the most important basic system software and provides various basic functions, such as scheduler and health monitor. At the top are the various software and system partitions that run on the operating system, and SOA and agents are also part of this.



**Figure 3.** Avionics architecture with agents and SOA.

### 3. Agents and SOA Modeling Method Based on AADL

This section introduces the characteristics of agents and SOA, and proposes a modeling method suitable for describing these characteristics based on AADL.

#### 3.1. Modeling Method for Agents

Compared with traditional software, the agent can perceive the environment, solve problems by reasoning and respond to the external environment through actions. The working state of the agents will change autonomously according to the external input, leading to possible changes to the characteristics of the agents, such as software reliability, execution time and other attributes. Recent studies show that the component attributes have a significant influence on the quantitative analysis of the avionics system; as a result, the following section will stress the working state of the agents to describe and model the agents better.

An agent can be defined as

$$A = (X_A, Y_A, S_A, U_A), \quad (1)$$

where  $X_A$  is the input set and  $Y_A$  is the output set of the agent;

$$S_A = \{s_{A1}, s_{A2}, s_{A3}, \dots, s_{An}\}, \quad (2)$$

where  $S_A$  is the working state set of agent,  $s_{An}$  denotes nth working state.

$$U_A = \{u_{A12}, u_{A13}, u_{A31}, \dots, u_{Aij}\}, \quad (3)$$

where  $U_A$  is the set of trigger conditions of the working state transition;  $u_{Aij}$  denotes the trigger condition for transitioning from the ith working state to the jth working state.

In the process of modeling the agent, the appropriate components need to be selected to describe the agent first. The agent in the avionics system is the software with specific functions, so the modeling method of software is used to describe the agent. Therefore, the agent is represented by a process component, which receives input or output through interfaces on the process component. The input elements in set  $X_A$  and output elements in set  $Y_A$  can be events, data or messages. Next, due to the multiple working states of an agent, the process component cannot describe its properties in different states, so modes are used

to represent the working states in set  $S_A$  at the component level. In different modes, the properties of the process component can describe the detailed characteristics of the agent through different attribute values. In the trigger condition of mode transition, specified events, data or messages, which are in set  $U_A$ , can be set according to the characteristics of the agent to trigger the transition of the working mode of the agent. In Figure 4, the following three states of agent p1 are set through the mode component: mode1, mode 2 and mode 3, and three conditions, trigger1, trigger2 and trigger3, are set to trigger the transition of the agent state.

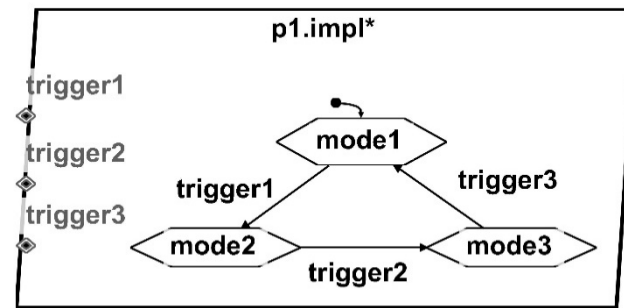


Figure 4. AADL model of an agent.

### 3.2. Modeling Method for SOA

In the service-oriented architecture of avionics systems, various functional units are encapsulated as services through specific interfaces and protocols, and services have specific functions, which can be provided to other services or applications for use. SOA enables distributed deployment, composition, and reuse of services, eliminating differences in hardware platforms, operating systems, and programming languages. This loosely coupled and architecturally flexible architecture is difficult to describe with traditional static architecture modeling methods. If this process is described using a static architecture modeling approach, all services and applications need to be modeled and coexist with the overall system architecture, and repeated services need to be modeled repeatedly. This static architectural model cannot accurately describe the actual process by which SOA executes application.

An SOA can be defined as

$$B = (X_B, Y_B, B_B, S_B, K_B, U_B), \quad (4)$$

where  $X_B$  is the input set and  $Y_B$  is the output set of the SOA;

$$B_B = \{B_{B1}, B_{B2}, B_{B3}, \dots, B_{Bn}\}, \quad (5)$$

where  $B_B$  is the application set of SOA;  $B_{Bn}$  denotes nth application.

$$S_B = \{s_{B1}, s_{B2}, s_{B3}, \dots, s_{Bn}\}, \quad (6)$$

where  $S_B$  is the service set of SOA;  $s_{Bn}$  denotes nth service.

$$K_B = \{k_{B1}, k_{B2}, k_{B3}, \dots, k_{Bn}\}, \quad (7)$$

where  $K_B$  is the set of invocation relationships between applications and services;  $k_{Bn}$  denotes the invocation relationship between nth application and corresponding services.

$$U_B = \{u_{B1}, u_{B2}, u_{B3}, \dots, u_{Bn}\}, \quad (8)$$

where  $U_B$  is the set of application requests;  $u_{Bn}$  denotes the request to nth application.

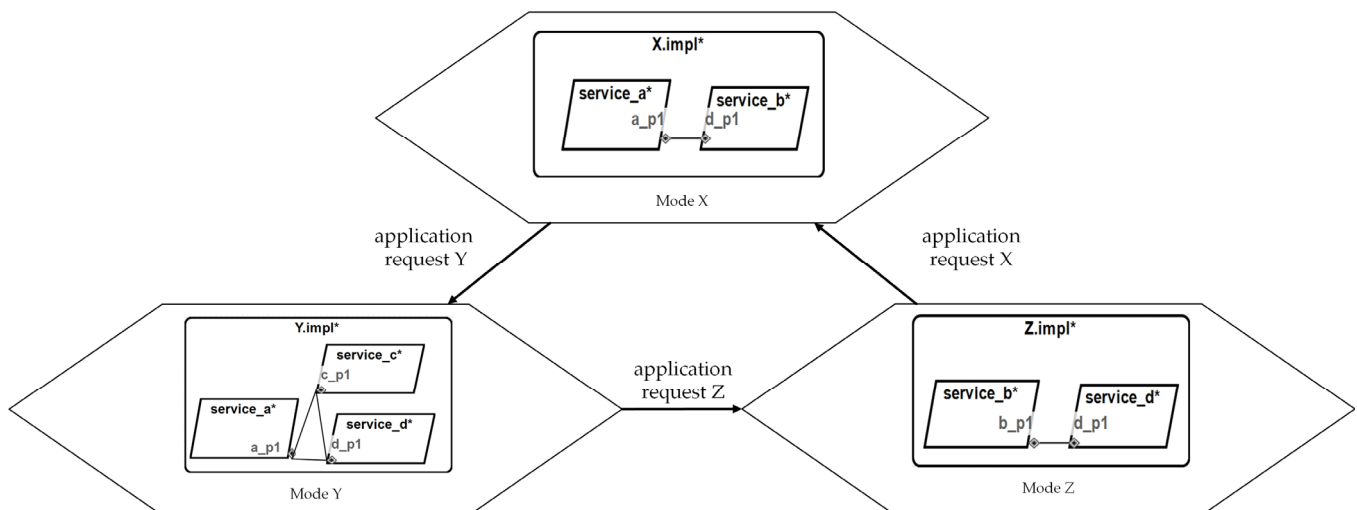
Representing component configurations and connections in the system, the modes in AADL can be triggered by set events, data, or messages to switch from source mode to



target mode. These characteristics of modes can model SOA better than traditional static architecture models.

In the modeling process of SOA, the service in set  $S_B$  is first represented by process components. Since application will invoke one or more services, it is represented by the system component. System components can contain multiple process sub-components, which are used to represent the relationship between application and service. Next, the modes are used at the system level, and different modes of system components are used to represent different applications in set  $B_B$ . Because the modes of system components can represent the configuration and connection of components in the system, the components and configuration relationships contained in different modes of system components are also different. System-level modes can be used to represent invocation relationships in set  $K_B$  between applications and services. Based on this, the application request in  $U_B$  is represented with a trigger condition for the mode transition, such as an event, data, or message, and the SOA system invokes the services and completes the application according to the application request. The trigger condition can enable the transition of the system component from the source mode to the target mode, that is, from the original application to the target application.

For example, in an SOA system, there is application X with two services A and B, application Y with three services A, C and D, and application Z with three services B and D. After service X is executed, the system receives service request Y and executes it. After service Y is executed, the system receives service request Z and executes it. To model this process using the proposed method, the service architecture in these applications should be described first, that is, the process component configuration in the system component. Then, the mode component is used to describe the target services executed by the system after receiving the service request. That is, the service request triggers the mode transition. As shown in Figure 5, three different modes are used to describe the invocation of services by applications in the system. For example, when application Y is executed after receiving application request Y, the system application will transition from X.impl to Y.impl and call the internal services of Y.impl.



**Figure 5.** An example of SOA system modeling.

Similar to the modeling method for agents, during SOA modeling, other software components such as the thread component, thread group component, data component and corresponding attribute values of each component, can also be added to the process component to describe services in SOA in detail.

In addition, for the agent modeling method in the previous section, if the agent is regarded as an independent agent service in SOA, when the agent service is included in the application, it can be added to the system component, representing the application as a



process component. The component-level modes used in the agent model to describe the state of work can work together without conflict with the system-level modes used in the SOA model to describe the application.

#### 4. Case Study

Based on a case study of some functions in an avionics system, modeling of advanced avionics architecture, including agents and SOA, is completed.

The model is divided into the following three levels: front-end radar module, signal processing module and application module. With the hierarchical distribution, the modeling method of “signal-data-knowledge” is adopted. In the working process, the front-end radar module detects the target signal, and the signal processing module processes the radar signal to obtain data. The application module analyzes and processes the data to obtain knowledge, which is used in the follow-up work, such as path planning, communication and equipment monitoring.

The front-end radar module contains two radar modules. The radar module contains the radar detection equipment, FPGA module and switch. The device component represents the radar detection equipment and switch, and the processor component represents the FPGA module. Signal processing module includes the signal processor, FPGA module, power supply and data storage module. The device component represents the power supply, processor component FPGA module and signal processor, and memory component data storage module. The application module contains multiple software components, such as the Data Distribution Service (DDS), Remote Procedure Call (RPC) component and SOA System. The SOA system includes intelligent path planning, communication and equipment health status monitoring and related services. The SOA system is represented by the system component and the DDS and RPC component are represented by the process component. The graphical model of the three-level architecture of the avionics system is shown in Figure 6. Moreover, in Figure 6, the green part represents the signal level, the red part represents the data level and the yellow part represents the knowledge level. These three parts together form the three-level architecture.

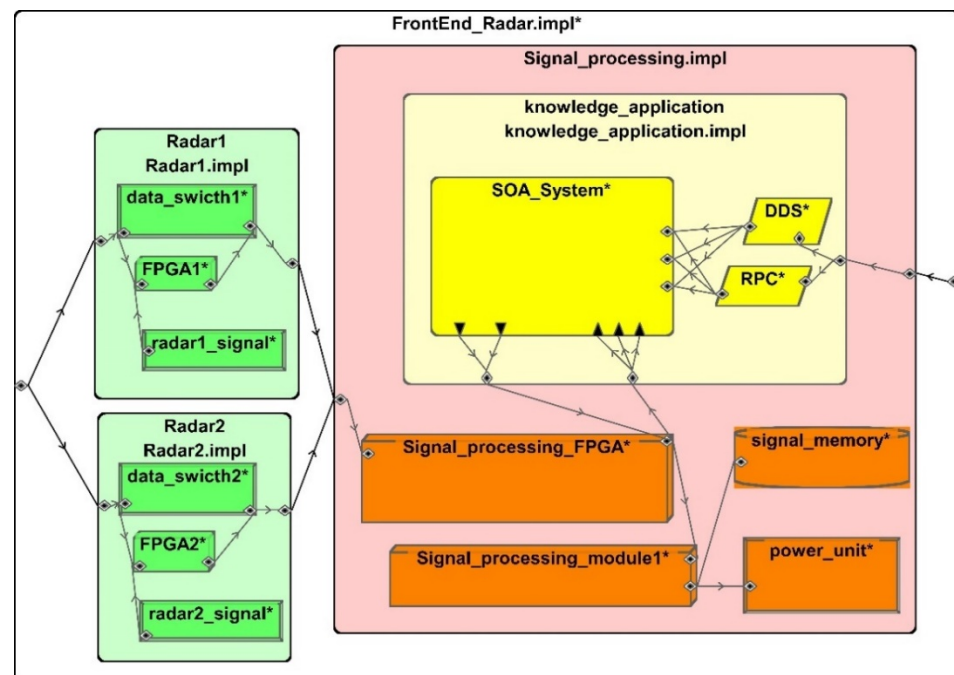


Figure 6. AADL model of avionics three-level architecture.

In the SOA system, there are three applications, including intelligent path planning, communication and equipment health monitoring, and five services, including the path

planning service, communication management service, signal reception service, signal transmission service, equipment signal acquisition service and equipment status analysis service. The mapping between application and service is as follows: intelligent path planning application includes the intelligent path planning service, signal reception service, and signal transmission service. Communication application includes the communication management service, signal reception service and signal transmission service. Equipment health monitoring application includes the equipment signal collection service and equipment status analysis service.

The path planning service includes an agent, and the agent modeling method is used to model the service. According to the characteristics of the agents, its state is divided into the following three states: target recognition, obstacle warning and path planning. The agent service has different execution time attributes in the three states, and its state transition is triggered by different image data input.

This system is modeled using the SOA modeling method in this article. The system component is used to describe the intelligent path planning, communication and device health monitoring in the system, and the process component is used to describe the services contained in the application. The application transition of the system is triggered by task request events. For intelligent path planning services that contain agents, process components are used to describe them. A graphical model of the SOA system is shown in Figure 7. In Figure 7, the colors red, yellow and green are used to represent three different applications, including intelligent path planning, communication and equipment health monitoring. In the traditional software system modeling method, these three different applications need to be deployed independently in the system with fixed architecture simultaneously. However, the SOA executes the application and invokes the corresponding service upon request, so the modeling method presented in this paper better describes how SOA works and its flexibility.

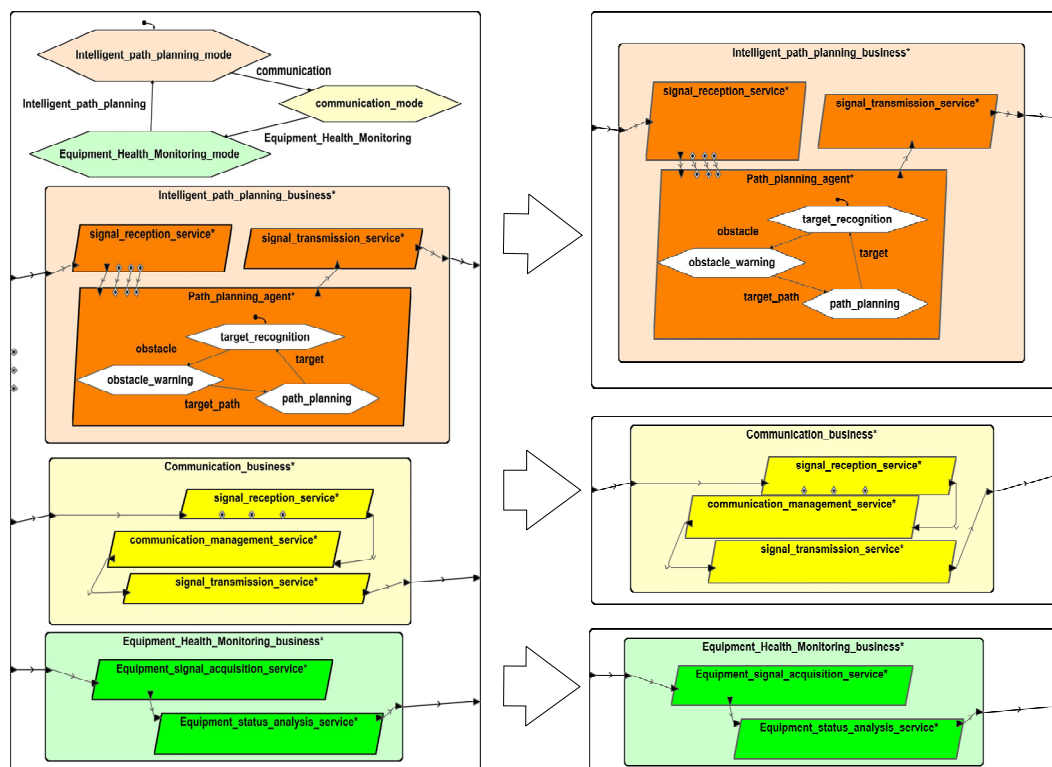


Figure 7. AADL model of SOA system with agents.

In this case, services with the same name in different applications refer to repeated invocation of the same service. As can be observed from the AADL model code snippets

below, the signal reception service and signal transmission service in the intelligent path planning application and communication application are invocations to the same service rather than repeated implementation of the same function. This modeling method reflects the loose coupling and architectural flexibility of SOA. Assuming that new applications added to the model in the future contain the same service, the existing service can be invoked in the same way.

Based on this model, this case will show how this modeling method works in practice by analyzing the delay time of a task path. In this path, radar1\_signal in Radar1 first obtains the signal and sends it to FPGA1 for preliminary signal processing, Then, the processed data is transferred from data\_switch1 to Signal\_processing\_FPGA for secondary data processing. The processed data is then transmitted to the SOA\_System, which executes applications based on system tasks, such as path planning or equipment health monitoring. The path information is as follows: radar1\_signal-FPGA1-data\_Switch1-Signal\_Processing\_FPGA-SOA\_System. The SOA\_System receives control instructions from DDS and RPC to execute a specific application. For intelligent target analysis applications that include agents, the image data will trigger the agent working state transition. The task path and the components on the path model are highlighted in red, yellow, and green in Figure 8.

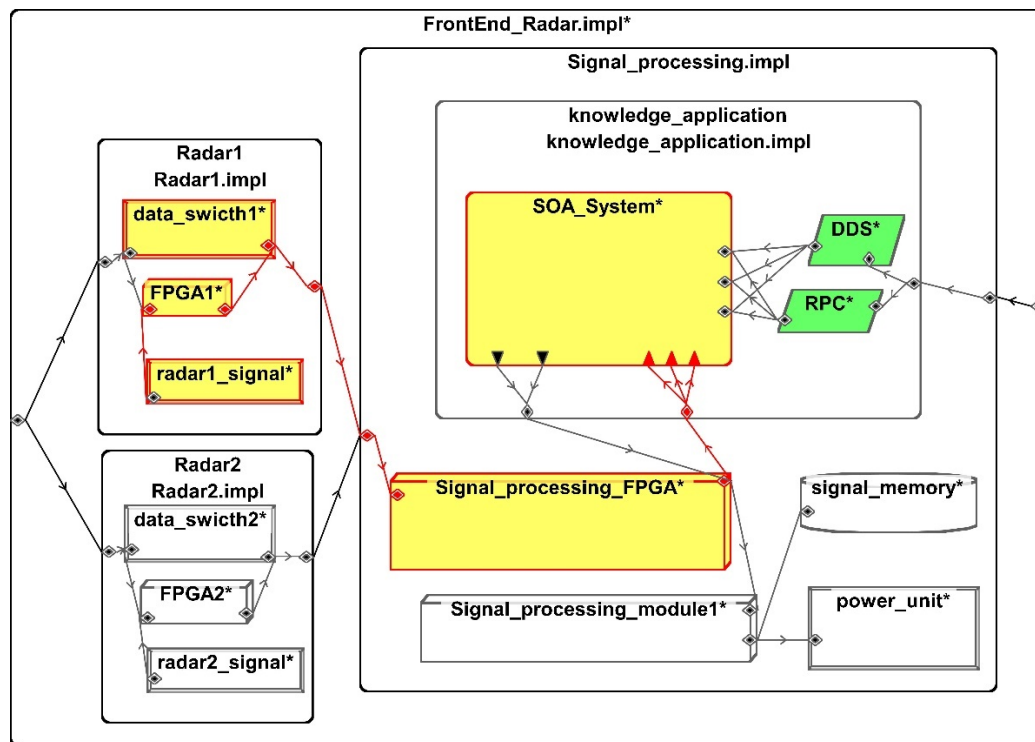


Figure 8. The task path in AADL model.

The delay time of the hardware components in the task path are shown in Table 1. There are four hardware components in the task path.

Table 1. The delay time of hardware components in the task path.

Component Name	Value (ms)
radar1_signal	5
FPGA1	10
data_switch1	4
Signal_processing_FPGA	15

The delay time of the SOA\_System and services inside are shown in Table 2. There are seven services in the SOA\_System.

**Table 2.** The delay time of services in SOA\_System.

Service Name	Value (ms)
SOA_System	5
Signal_reception_service	5
Signal_transmission_service	5
Communication_management_service	5
Signal_transmission_service	2
Signal_reception_service	5
Equipment_signal_acquisition_service	3
Equipment_status_analysis_service	10

For the target analysis service with agents, there are three working states, target recognition, obstacle warning, path planning. In the different working states, the delay time of the target analysis service is shown in Table 3.

**Table 3.** The delay time of target analysis service.

Working State	Value (ms)
Object_identification	20
Obstacle warning	27
Path_planning	35

Based on the above information, the delay time of this task path in three different application execution cases is analyzed. In the first case, the avionics system sends an application request through RPC to enable the SOA system to execute the equipment health monitoring application, and Equation (9) shows the calculation process of system delay time  $T_1$  in this case.

$$\begin{aligned}
 T_1 &= T_{\text{radar1\_signal}} + T_{\text{FPGA1}} + T_{\text{data\_switch1}} + T_{\text{Signal\_processing\_FPGA}} + T_{\text{SOA\_System}} \\
 &\quad + T_{\text{Equipment\_signal\_acquisition\_service}} + T_{\text{Equipment\_status\_analysis\_service}} \quad (9) \\
 &= 5 + 10 + 4 + 15 + 5 + 3 + 10 = 52 \text{ ms},
 \end{aligned}$$

In the second case, the avionics system sends an application request through RPC to make the SOA system execute intelligent target analysis application. At this time, the image information from the radar triggers the agent to transition to the obstacle warning state, and Equation (10) shows the calculation process of system delay time  $T_2$  in this case.

$$\begin{aligned}
 T_2 &= T_{\text{radar1\_signal}} + T_{\text{FPGA1}} + T_{\text{data\_switch1}} + T_{\text{Signal\_processing\_FPGA}} + T_{\text{SOA\_System}} + T_{\text{signal\_reception\_service}} \\
 &\quad + T_{\text{obstacle warning}} + T_{\text{signal\_transmission\_service}} = 5 + 10 + 4 + 15 + 5 + 5 + 5 + 27 \quad (10) \\
 &= 76 \text{ ms},
 \end{aligned}$$

In the third case, the avionics system sends an application request through RPC to make the SOA system execute intelligent target analysis application. At this time, the image information from the radar triggers the agent to transition to the path planning state, and Equation (11) shows the calculation process of system delay time  $T_3$  in this case.

$$\begin{aligned}
 T_3 &= T_{\text{radar1\_signal}} + T_{\text{FPGA1}} + T_{\text{data\_switch1}} + T_{\text{Signal\_processing\_FPGA}} + T_{\text{SOA\_System}} + T_{\text{signal\_reception\_service}} \\
 &\quad + T_{\text{path\_planning}} + T_{\text{signal\_transmission\_service}} = 5 + 10 + 4 + 15 + 5 + 5 + 5 + 35 \quad (11) \\
 &= 84 \text{ ms},
 \end{aligned}$$

Through the comparison of the system delay time in Table 4, the same task path in the system has different results, and the SOA executes different applications and the external input triggers different agent working states. It can be observed that the modeling method proposed in this paper can more accurately describe the working state of the agent, the SOA working process and the system attributes in the execution of different applications, and in practice, can improve the accuracy and effectiveness of the analysis.

**Table 4.** The system delay time in different cases.

Case	Value (ms)
T <sub>1</sub>	52
T <sub>2</sub>	76
T <sub>3</sub>	84

## 5. Conclusions and Future Work

By aiming to solve the problems of the existing modeling methods in describing the dynamic behavior and attribute changes in the agent and SOA in avionics systems, this study proposes a novel modeling method of agents and SOA in an avionics system based on AADL and modes. Firstly, according to the change in the working state of the agent, the environmental input feedback and multi working states modeling method of the agent are proposed based on the AADL, and the transition mechanism of the agent working state with the external input is described. On this basis, the attributes of the agent under different working states are described. Secondly, considering the service-oriented encapsulation and reuse of functions by SOA and the invocation of services by applications, an application and service modeling method is proposed based on AADL, aiming to describe the application, service and software systems in SOA, and describe the system's attribute parameters based on this. Compared with the current methods and technologies, the method proposed in this paper mainly reflects the advantages of two aspects. On the one hand, by modeling the agent dynamic behavior, the relationship between the agent working state change and the input is described. On the other hand, through the modeling of SOA services and applications, the architecture characteristics and attributes of SOA are described more accurately. In addition, this method has good compatibility with the existing AADL model of avionics systems, and has obvious effects in practice, which is of great significance to the design and analysis of avionics systems.

In the subsequent research, the modeling method will be refined and improved, and based on this, the system model analysis method, including agents and SOA, will be systematically studied.

**Author Contributions:** Conceptualization, P.D.; methodology, P.D.; investigation, Q.Z.; validation, D.A.; resources, D.A.; writing—original draft preparation, S.W.; writing—review and editing, K.L.; visualization, K.L.; supervision, Q.Z.; project administration, P.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Yang, W. Development of future fighters. *Acta Aeronaut. Astronaut. Sin.* **2020**, *41*, 8–19.
2. Sun, C. Development trend of future fighter: A review of evolution of winning mechanism in air combat. *Acta Aeronaut. Astronaut. Sin.* **2021**, *42*, 8–20.
3. Haider, K.; Tweedale, J.; Jain, L. An Intelligent Decision Support System Using Expert Systems in a MAS. In Proceedings of the 1st KES International Symposium on Intelligent Decision Technologies, Univ. Hyogo, Himeji, Japan, 23–25 April 2009.
4. Canino-Rodriguez, J.M.; Garcia-Herrero, J.; Besada-Portas, J.; Ravelo-Garcia, A.G.; Travieso-Gonzalez, C.; Alonso-Hernandez, J.B. Human Computer Interactions in Next-Generation of Aircraft Smart Navigation Management Systems: Task Analysis and Architecture under an Agent-Oriented Methodological Approach. *Sensors* **2015**, *15*, 5228–5250. [[CrossRef](#)] [[PubMed](#)]



5. Gaska, T. An Affordable Ima Bridge for Refreshing Deployed Avionics Systems. In Proceedings of the 2014 IEEE/Aiaa 33rd Digital Avionics Systems Conference (DASC), Colorado Springs, CO, USA, 5–9 October 2014.
6. Farcas, C.; Farcas, E.; Krueger, I.H.; Menarini, M. Addressing the Integration Challenge for Avionics and Automotive Systems-From Components to Rich Services. *Proc. IEEE* **2010**, *98*, 562–583. [\[CrossRef\]](#)
7. Ahmadi, R.; Marquardt, O.; Riedlinger, M.; Reichel, R. An Adaptive Software Architecture for Future CMS. *SAE Int. J. Aerosp.* **2015**, *8*, 260–272. [\[CrossRef\]](#)
8. Qi, Y.; Kang, M.; Xue, B.; Wang, S. Research on Service-Oriented UAV Route Planning Component Design. In Proceedings of the 2019 5th International Conference on Mechanical and Aeronautical Engineering (ICMAE 2019), Sanya, China, 12–15 December 2019.
9. Rodrigues, D.; Zaniolo, R.R.; Branco, K.R.L.J.C. Knowledge-Based Framework: Its Specification and New Related Discussions. In Proceedings of the 4th International Conference on Mathematical Modeling in Physical Sciences (ic-Msquare2015), Mykonos, Greece, 5–8 June 2015.
10. Rodrigues, D.; Pires, R.d.M.; Estrella, J.C.; Vieira, M.; Correa, M.; Camargo Junior, J.B.; Jaquie Castelo Branco, K.R.L.; Trindade Junior, O. Application of SOA in Safety-Critical Embedded Systems. In Proceedings of the 5th International Conference on Convergence and Hybrid Information Technology (ICHIT), Daejeon, Korea, 22–24 September 2011.
11. Tambe, S.; Balasubramanian, J.; Gokhale, A.; Damiano, T. MDDPro: Model-Driven Dependability Provisioning in Enterprise Distributed Real-Time and Embedded Systems. In Proceedings of the 4th International Service Availability Symposium, Durham, NH, USA, 21–22 May 2007.
12. Ren, R.; Zhu, L.; Wang, Z.; Xi, Q. A Design and Research of Digital Operation Support Framework for Civil Aircrafts. In Proceedings of the 2017 International Conference on Sensing Diagnostics, Prognostics and Control (SDPC), Shanghai, China, 16–18 August 2017.
13. Surhone, L.M.; Tennoe, M.T.; Henssonow, S.F. *Architecture Analysis and Design Language*; Betascript Publishing: Beau Bassin, Mauritius, 2010.
14. Zhang, Q.; Wang, S.; Liu, B. Approach for Integrated Modular Avionics Reconfiguration Modelling and Reliability Analysis Based on AADL. *Iet Softw.* **2016**, *10*, 18–25. [\[CrossRef\]](#)
15. Feiler, P.H.; Gluch, D.P. *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language*; Addison-Wesley Professional: Upper Saddle River, NJ, USA, 2013.
16. Maes, P. Agents That Reduce Work and Infomation Overload. *Communications ACM* **1994**, *37*, 30–40. [\[CrossRef\]](#)
17. Jennings, N.R.; Faratin, P.; Johnson, M.J.; Norman, T.J.; Wiegand, M.E. Agent-Based Business Process Management. *Int. J. Coop. Inf. Syst.* **1996**, *5*, 105–130. [\[CrossRef\]](#)
18. Wooldridge, M.; Jennings, N.R. Intelligent Agents: Theory and Practice. *Knowl. Eng. Rev.* **1995**, *10*, 115–152. [\[CrossRef\]](#)
19. Kim, D.S.; Chang, S.K.; Rim, K.W. Modeling and Design of Intelligent Agent System. *Int. J. Control. Autom. Syst.* **2003**, *1*, 257–261.
20. Min, F.; Yang, M. A Prototype for DEVS-Based Agent Model. *Syst. Simul. Technol. Appli-Cation* **2006**, *8*, 262–265.
21. Wang, J.; Tian, S.; Ye, W. The Study on Modeling Method of Agent. *Syst. Simul. Technol-Ogy Appl.* **2011**, *13*, 228–230.
22. Xu, J.; Zhong, Z.; Xu, L. ISHM-Oriented Adaptive Fault Diagnostics for Avionics Based on a Distributed Intelligent Agent System. *Int. J. Syst. Sci.* **2015**, *46*, 2287–2302. [\[CrossRef\]](#)
23. Kashi, R.N.; D'Souza, M. Mitigating Byzantine Failures in Multi-Agent Based Dependable and Adaptable Avionics Software. In Proceedings of the 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 20–22 February 2019.
24. D'Souza, M.; Kashi, R.N. Avionics Self-Adaptive Software: Towards Formal Verification and Validation. In Proceedings of the Distributed Computing and Internet Technology, Bhubaneswar, India, 10–13 January 2019.
25. Nagaraj, R.K.; D'Souza, M. A Verifiable Multi-Agent Framework for Dependable and Adaptable Avionics. *Sadhana-Acad. Proc. Eng. Sci.* **2021**, *46*, 27. [\[CrossRef\]](#)
26. Marc, F.; Degirmenciyan-Cartault, I.; El Fallah-Seghrouchni, A. Multi-Agent Planning as a Coordination Model for Self-Organized Systems. In Proceedings of the IEEE/Wic International Conference on Intelligent Agent Technology, Halifax, NS, Canada, 13–17 October 2003.
27. Kashi, R.N.; D'Souza, M.; Baghel, S.K.; Kulkarni, N. Formal Verification of Avionics Self Adaptive Software: A Case Study. In Proceedings of the 9th India Software Engineering Conference, BITS Pilani, Goa, India, 18–20 February 2016.
28. Elkholy, W.; El-Menshawy, M.; Bentahar, J.; Elqortobi, M.; Laarej, A.; Dssouli, R. Model Checking Intelligent Avionics Systems for Test Cases Generation Using Multi-Agent Systems. *Expert Syst. Appl.* **2020**, *156*, 113458. [\[CrossRef\]](#)
29. Wu, H.; Kang, F.; Huang, W.; Lu, Y. Modeling and Simulation of Pilot Behavior Interfaced with Avionics System. In Proceedings of the Asia Simulation Conference/International Conference on System Simulation and Scientific Computing (AsiaSim and ICSC 2012), Shanghai, China, 27–30 October 2012.
30. Erl, T. *Service-Oriented Architecture: Concepts, Technology, and Design*; Prentice Hall: Upper Saddle River, NJ, USA, 2005.
31. Zhou, Q.; Deng, P.; Liu, Q.; Hong, R. Research on Service-oriented Advanced Avionics Ar-chitectur. *Avion. Technol.* **2018**, *49*, 1–7.
32. Zimmermann, O.; Kroghdahl, P.; Gee, C. Elements of Service-Oriented Analysis and Design. Technical article, IBM (2 June 2004). Available online: <http://www.ibm.com/developerworks/webservices/library/ws-soad1/> (accessed on 7 June 2022).
33. Pant, K.; Juric, M.B. *Business Process Driven SOA Using BPMN and BPEL*; Packt Publishing: Birmingham, UK, 2008.

- 
34. Zdun, U.; Hentrich, C.; Dustdar, S. Modeling Process-Driven and Service-Oriented Architectures Using Patterns and Pattern Primitives. *ACM Trans. Web* **2007**, *1*, 14-es. [[CrossRef](#)]
  35. Bertrand, D.; Déplanche, A.M.; Faucou, S.; Roux, O.H. A Study of the AADL Mode Change Protocol. In Proceedings of the IEEE International Conference on on Engineering of Complex Computer Systems, Belfast, UK, 31 March–3 April 2008.