



**DYNAMIC POSITIONING CONFERENCE**  
October 11-12, 2016

COMPENTENCY/DESIGN SESSION

---

## Application of AADL for Marine Control Systems

By Aman Batra<sup>1</sup>, Sofien Kerkeni<sup>1</sup>, Pierre Dissaux<sup>2</sup>, Frank Singhoff<sup>3</sup>

D-ICE ENGINEERING,  
ELLIDISS TECHNOLOGIES,  
Université de Bretagne Occidentale, Lab STICC UMR CNRS 6285

## I. Abstract

For decades the critical importance of functions offered and ensured by electrical or software based systems has led to the development of new modeling languages, verification techniques, methods and tools for mastering their realization and their maintenance. This is particularly true for space and avionics domains. The application of these techniques has allowed people to respect the certification constraints that are required for embedded systems. The increasing complexity of these software and systems require to work on architecture models and proceed to the verification activities at the earliest stages of the development life-cycle.

In this paper, we show how modeling tools and verification techniques that have been initially developed for aerospace can be applied on a DP system. DPS are affected by failures originating from various components. These failures must to be identified in the earlier stages of the design and development process; otherwise the cost incurred to make the necessary changes may be huge. Nowadays DPS are aimed to be secured, reliable and available at all time. Therefore it is essential to point out these failures as early as possible.

First, DPS modelling is proposed using AADL (Architecture Analysis and Description Language) [1]. AADL is an international standard issued by the SAE (Society of Automotive Engineering). This model is then used for RAMS Analysis (Reliability, Availability, Maintainability, Safety).

Thanks to an extension to the language, the Error Model Annex, AADL demonstrates the very interesting ability to describe formally failures of components and to study the propagation of errors. This language, these analyses and methods could be considered as the next step in terms of system analysis and assessment, system optimization and FMEA (Failure Modes and Effects Analysis), etc.

## II. Abbreviation / Definition

<b>A</b>	
AADL .....	5
<b>D</b>	
DGPS .....	18
DPS.....	5
<b>E</b>	
EMV .....	10
<b>F</b>	
FHA .....	8, 12
FTA.....	8, 12, 15
<b>H</b>	
HPR .....	12
<b>I</b>	
IJS.....	11
<b>L</b>	
LTW.....	18
<b>M</b>	
MRU.....	12
MTBF.....	19
<b>N</b>	
NIST.....	3
NRPD.....	18
<b>O</b>	
OSATE .....	5
<b>P</b>	
PMS.....	5
<b>R</b>	
RBD .....	17
<b>S</b>	
SAE .....	5
SEI .....	8
<b>U</b>	
UML .....	5
UPS.....	11

### III. Introduction

System safety analysis methods have evolved for many years. Today they are mature and used in many diverse fields. They incorporate risk management, hazard identification and analysis techniques necessary to support systems development processes. Safety analysis methods detect, evaluate and can therefore optimize system safety.

Traditionally the analysis methods such as fault trees and failure modes and effects analysis were created manually and rely significantly on the expertise of the analyst. As the complexity of the system increases, the possible amount of failures also increases. Therefore by automating the safety analysis methods the accuracy and the quality of the analysis can be maintained [2] [3].

The impact of software and hardware architecture is crucial for safety critical systems and its realization is therefore extremely significant. Along with the growing complexity of the systems, the software architecture is becoming more and more complex. Hence, the chances of introducing faults at different stages also increase. According to the National Institute of Standards and Technology (NIST) 2002 study and illustrated on Figure 1, 70% of the faults are added in the initial phases while 80% of these faults are not detected until the last phase. Consequently the cost for rectifying the systems are enormous and the relevance of fault free software architecture is paramount [3] [4].

The following figure depicts the cost escalation problem encountered during the standard V life cycle of software development

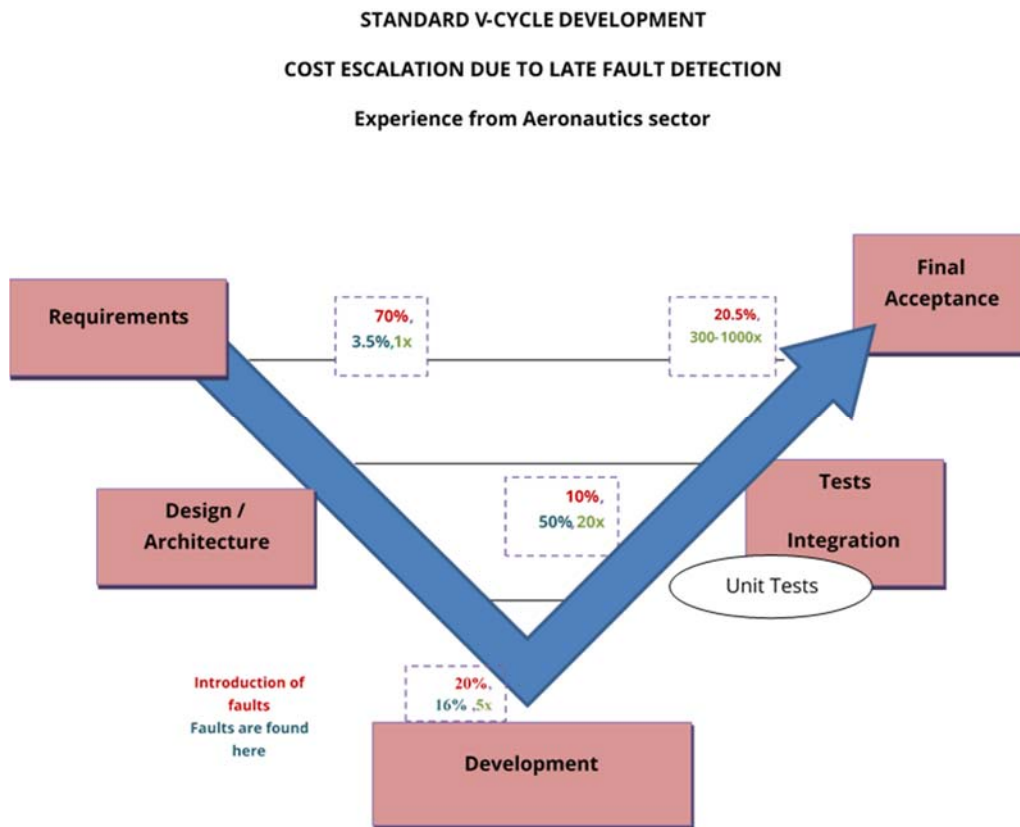


Figure 1: Cost escalation due to late fault detection from [5].

Model based development can be defined as a paradigm for system development. Model based development usually relies on domain specific languages which provide textual and graphical representations of the pertinent entities of the system. Modeling is a mandatory activity of the system development process. Along with modeling, simulation and analysis are also crucial activities. Model based development has numerous benefits. It can be used to investigate complex systems which are difficult to study. Moreover, it can be used to examine the effect of changes to the system without producing an actual prototype [2] [4].

Modeling and simulation are disciplines which create an understanding of the relation amongst system components and system altogether. Engineers usually apply domain specific softwares to create and simulate their models in order to perform analysis. As an example, computer hardware engineers employ Very High Speed Integrated Circuits Hardware Description Language (VHDL) for modeling and simulation operations. Whereas control engineers use Simulink for creating systematic representation of their control models. These evolving models are created at different levels of development process and with various considerations. Therefore due to inconsistency between these models, the system failures endure till the final stages of the development [2] [4]. Figure 2 illustrates such issue.

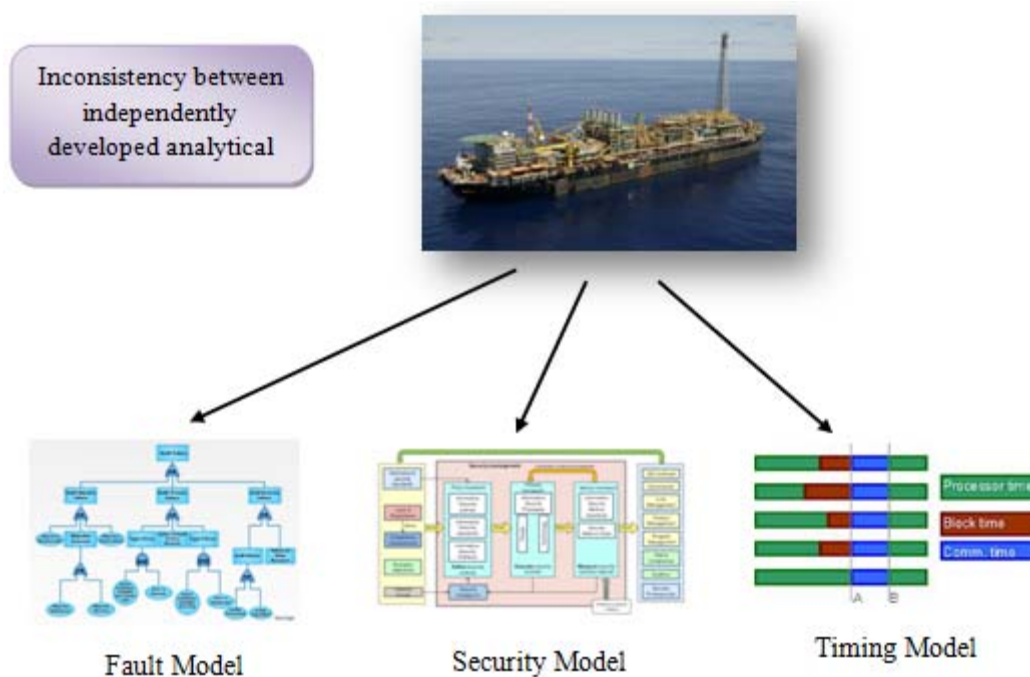


Figure 2: Inconsistent Analysis between different models.

Hence, model based development is quite significant as it employs parallel based design methodology instead of the conventional series methodology. In the typical software development process, the phases occur consecutively with end of each phase marking the beginning of other. Whereas in the model based development methodology, design and implementation phases occur in parallel. Figures 3 and 4 throw light on this ideology.

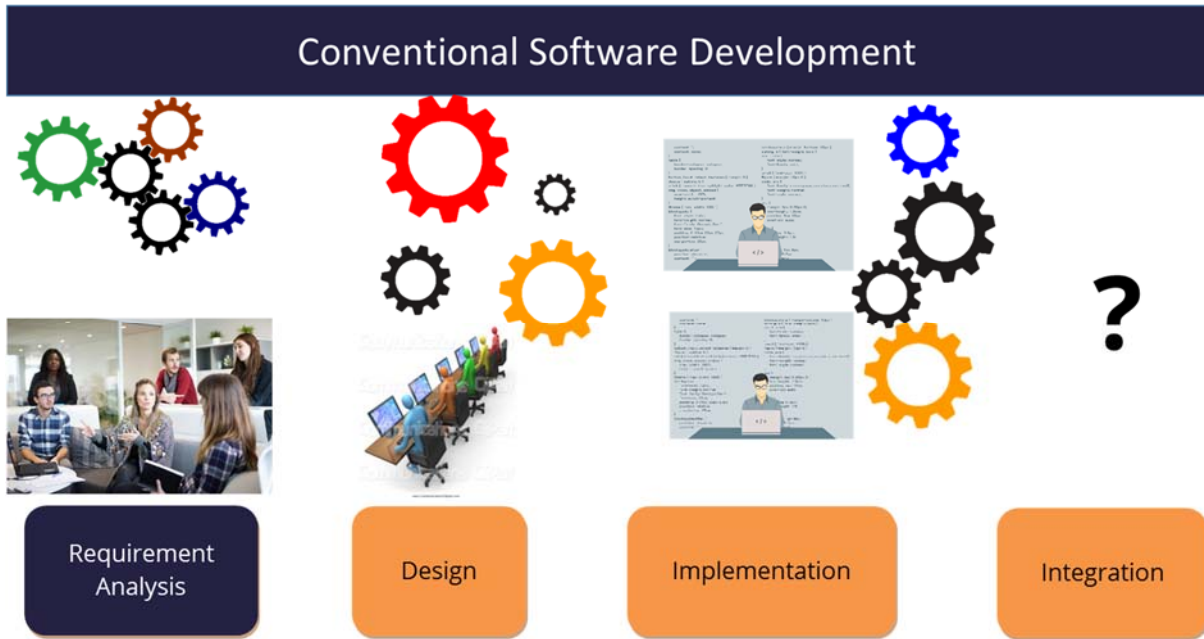


Figure 3: Description of typical software development process [6].

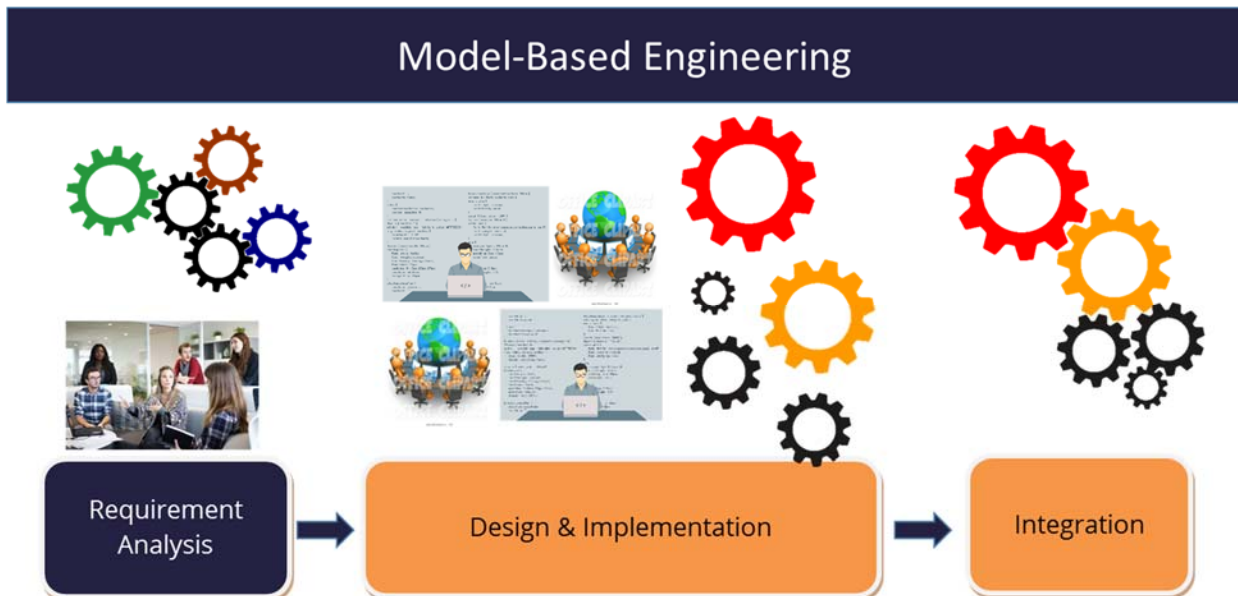


Figure 4: Description of software development using model-based process development [6].

System failure can also occur due to a single fault in one of the components which propagates to other components. A basic fault in a component can lead to a big failure in the system. Such situations have also been reported in the marine industry. The main reason of these incidents being the inadequate inspection of the components before a marine operation. The next given paragraph highlights the above notion.

In July 2002 in the United Kingdom Continental Shelf (UKCS) region, aboard a vessel there was an incident of dynamic positioning system failure. At around 02:56 hrs, a complete blackout happened, which caused all the vessel systems powered from vessel’s power management system (PMS) to be lost due to power

failure. It took around 15 minutes for all the systems to be restored and to start working normally. This occurred due to the inadequate connection of the terminal resistors on printed circuit boards of the Programmable Logic Controller (PLC), which caused the erroneous signal to be transferred to the PLC [7].

Another incident regarding fault in power management system took place in July 2002 aboard another ship. The failure of a timer in a bow thruster has led to an over current in the bus bar. This has caused the failure of one of a card of the control system board due to their close location. Due to this fault, two stern azimuth thrusters have failed and the DP system changed from automatic mode to manual mode. As a result, the vessel has drifted off by over 40 metres [8].

In this paper we will initially provide some contemporary history and introduction of AADL. We will also present some benefits of AADL, which makes it preferable over other languages. Afterwards we introduce the various categories of AADL components and the Error Model Annex. Next, we will discuss a possible description of the dynamic positioning system and provide an AADL-centric safety analysis approach. Subsequently, we will examine and assess the achievements to justify the fact we selected AADL.

#### IV. AADL: Presentation & History

The SAE AADL standardization committee was established in 1999. In 2000, representatives from 10 Aerospace companies issued requirements to initiate the AADL standard [9] [10]. In 2001, a first draft was created. In 2002, major organizations such as the European Space Agency and Airbus have identified AADL as a strong candidate for their system and software architecture needs [11]. The first public version of OSATE (Open Source Architecture Tool Environment), the AADL reference environment was released in 2004 [12].

The initial standard was expanded by adding the AADL Meta model and XML Metadata Interchange (XMI) format, graphical AADL symbols, programming language interface and Error Model Annex in 2006 [12]. In 2009 SAE has incorporated in the standard more enhancements based on the experience of AADL with the industry. In 2011, the AADL standard was further augmented by adding the Behaviour Annex, Data Modelling Annex and ARINC653 Annex [12].

Large software application involves intensive design and production phases. During these phases, the cost incurred by the implementation errors has to be minimized and the effectiveness and accuracy has to be maximized. The National Institute of Standards and Technology (NIST) has reported in 2002 that the software errors costs incurred by the U.S. economy is \$59.5 billion per year [13]. Therefore, there was a clear need to rely on architecture modeling languages such as UML (Unified Modeling Language) and AADL. This will indeed ensure the mitigation of almost all of these errors.

These diagnostics initially done in aerospace or automotive sectors could easily be applied to marine control systems and especially for DP systems [14]. The intricacies and the complexity of the DPS make crucial to identify, evaluate and analyse the errors and failures originating in the DPS. Consequently, the benefits of AADL make it a strong candidate for application in the DPS.

#### V. AADL: Introduction

The Architecture and Design Language (AADL) [1] is a language used for the dynamic architecture of the system. It is utilized for describing software and hardware components, specifying their nominal and

erroneous behaviors and formalize their interactions with the external environment. It can be used to check and monitor redundancy of system through distinct analysis methods [3]. This includes finding root causes of all failures occurring in system either by a single component or several.

A real time system can be defined as the one which has to process information and give response within a defined timeframe or else it should face the severe consequences such as a failure [15]. An embedded system is a combination of computer hardware and software, containing a fixed capability or variable (i.e. programmable) [16]. SAE AADL is utilised for creating the predictable model-based systems for real-time and embedded systems [17].

The AADL model can be used to assess for purposes such as checking the consistency of the system faults, the accuracy of the system architecture and to run various analysis methods. It brings well defined semantics of the component-based model. These semantics are beneficial for the construction and analysis of the structured model [4].

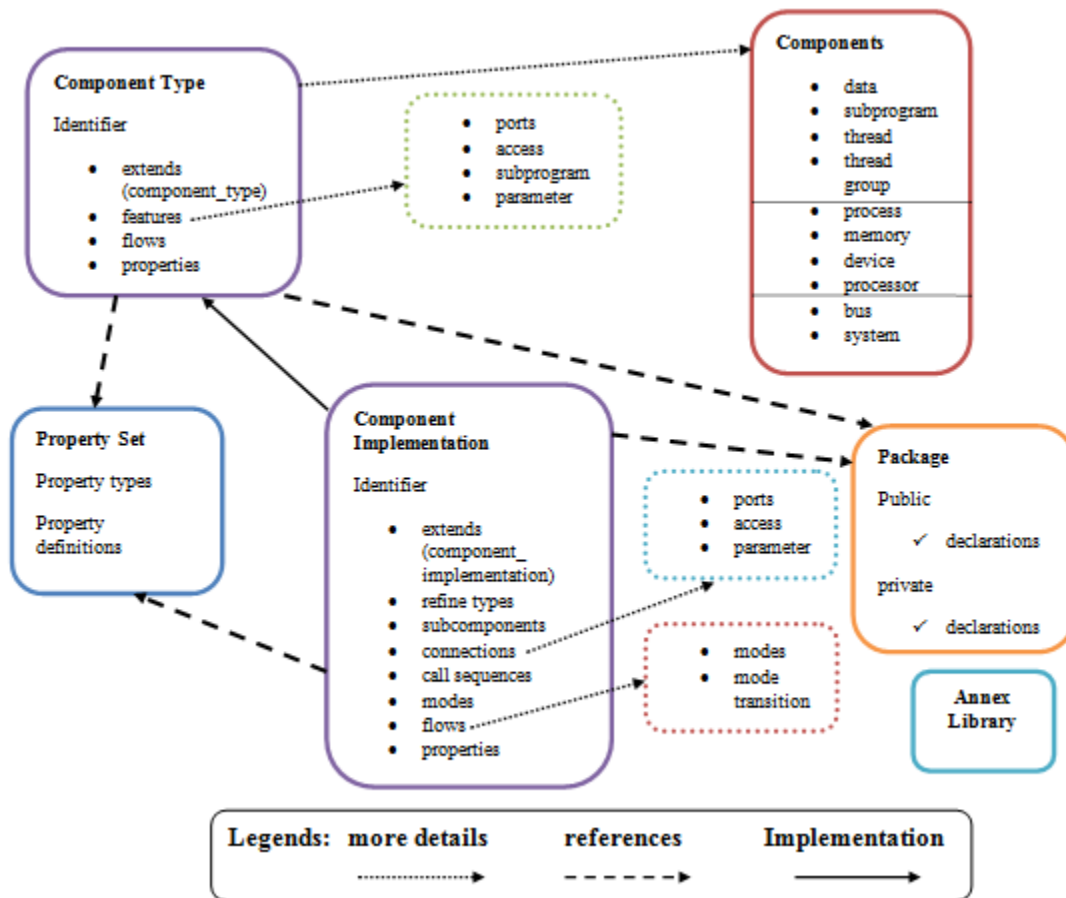


Figure 5: Summary of the AADL elements [18].

The above figure shows the various entities of AADL and their interactions. The AADL declarations of component type and component implementation, the various categories of components (software, hardware and composite), the features like the ports and parameters, the mode transitions and the Error Model Annex library are illustrated in this figure. The component type defines the interface of the component and encompasses flows, features and properties, while the component implementation describes its internal



definition by enclosing the properties such as subcomponents, connections, modes, etc. The software category of components consists of data, subprogram, thread, process and thread group. Whereas device, processor, memory and bus are the hardware components. A system entity models composite component. The purpose of ports is to model input and output flow of data to and from the components. The Error Model Annex library allows the specification of information related to error such as the errors types and the properties.

```

package Library::Sensor
public
with EMV2;
with Library::Errors;
with arp4761;
device Sensor
features
  signal : out data port;
annex EMV2 {**
  use types Library::Errors;
  use behavior Library::Errors::simple;
  error propagations
    signal : out propagation {BadValue};
  flows
    ef : error source signal{BadValue};
  end propagations;
  properties
    emv2::hazards =>
      ([ crossreference => "N/A";
        failure => "BadValue";
        phases => ("all");
        severity => ARP4761::Minor;
        likelihood => ARP4761::Probable;
        description => "Bad value from the sensor";
        comment => "Alarm would be initiated but no
          immediate effect on position keeping capabilities
          because of presence of alternative sensors.";
        ])
    applies to ef.BadValue;
**};
end Sensor;
end Library::Sensor;

```

Figure 6: AADL syntax of component 'Sensor'.

The figure 6 depicts a simple AADL syntax of the 'Sensor' component. The model starts with the declaration of the package and the libraries under the 'public' domain. The component 'device' is employed for the 'Sensor'. The next part declares the ports which are categorized as 'in port' or 'out port' as per their function. The ports model the interaction between a component and its external environment. The next

section consists of Error Model Annex. It is a collection of error and reliability declarations. It provides information about error propagations and transitions, error flows, properties, etc.

The AADL language also supports modelling and analysis in several views and information about the varying effects of faults on the system. The execution platform consisting of the hardware and software components which are associated with the timing and performance analysis of the system [19].

The AADL, similarly to its predecessor MetaH (which was developed in 1990 by Vestal Honeywell Technology Centre), is a modeling language that not only defines the textual and the graphical representation for the architecture but also consists of a well-defined syntax and semantics that allows the system to adequately depict the real time properties of the systems and its functions [13].

The models in the AADL can be created with the level of obligations required. Partly created models can also be analyzed and worked upon [20]. Promptly defined semantics of the AADL aids in the utilization of diverse analysis methods. These methods provide qualitative as well as quantitative results. The chosen architectural preferences can be appraised and affirmed [21]. The development process can be improved using the elucidated architectural model developed by AADL.

OSATE (Open Source Architecture Tool Environment) is an open source toolset platform to support AADL which was developed by the Software Engineering Institute (SEI), Carnegie Mellon University. It supports many features which are significant for simulating, prototyping and analyzing the quality of the system at every phase of abstraction [22].

The OSATE contains a compiler for textual AADL, a graphical editor for AADL, an instance model generator and supports the XML-based XMI interchange format for AADL established on its Meta model specifications [23]. The analysis tools that are available in its library are FHA (Functional Hazard Analysis), FTA, Consistency Checks, Unhandled Faults Analysis and Reliability Block Diagram.

The FHA (Functional Hazard Assessment) is a technique that scrutinizes the effects of functional failures on the components of a system [24]. The FTA (Fault Tree Analysis) is a logic block diagram that presents the state of a system (failures) as a concoction of the states of its components [25]. Consistency Checks are the obligatory checks carried out on a system in order to review the rationality of the system [26]. Unhandled Fault Analysis is used to check that whether all the faults present in the system are managed [27]. A RBD (Reliability Block Diagram) does the system reliability analysis on the intrinsic systems by exploiting the relationships between the components [28].

There are several benefits of AADL which gives it an edge over other modelling languages. AADL defines component centric interaction semantics. This enables the modeller to create an intricate model and to bring it as closer to the actual real-life model specifications. The AADL offers a broad variety of viewpoints, for example the modeller can choose the type of AADL defined components to be used for the system [29].

The major difference between the AADL and UML (Unified Modelling Language), which is another well-known modelling language is given as follows:

- AADL is textual and graphical in nature while UML is only graphical in nature,
- AADL contains declarative model instances whereas UML contains declarative model only,
- AADL has precise semantics on a limited area while UML semantics are lower on a wider area,
- AADL consists of standardized extensions whereas UML consists of generic extensions
- and AADL consists of components while UML consists of hierarchial graphic classes.

AADL comprehensively seizes the execution nature of the software as well as hardware components. The execution nature is generally responsible for the intrinsic properties such as reliability, safety and performance of a system [29]. AADL is supported by a variety of academic and commercial tools. The most famous one is OSATE, developed by the SEI. Others are Ocarina (AADL compiler and code generator), developed by Telecom ParisTech, ISAE and ESA; MASIW, developed by the Academy of Sciences of Moscow; Stood and AADL Inspector, developed by Ellidiss Technologies.

## VI. AADL: Components

System modelling in AADL is supported through numerous categories of components. On the basis of their functions components can be divided into software, hardware, and composite groupings [18] [17] [30].

The thread, process, data, thread group and subprogram constitute the software abstractions. A thread correlates to a synchronously executing component. A scheduler is utilized for the execution of threads. Threads also contain dispatch protocol property value. While a process is the component that encloses its constituents into a protected address space and comprises of special partitions in terms of virtual address spaces. Threads are typically contained in a process.

A data component serves as a data type in source text. It is shared by components, ports and subprogram parameters. The function of a thread group is organizing threads and data as a solitary constituent that are always present in a process. The thread group necessitate access to the subcomponents such that its constituents can interact with the surroundings. Whereas a subprogram is a perceptibly executable code. It is callable from threads and other subprograms.

Hardware components consist of the processor, memory, device and bus. The objective of a processor component is to schedule and execute threads. It also involves functionalities of an operating system. It may also contain memory components and is connected to buses. While the memory component accumulates code and data. It is used for the modelling of RAM or ROM memories. It can also contain nested memory components and are often connected to buses.

The device component refers to the external components which communicate with the surrounding environmental components. They give physical significance to a system component. They can be connected to the software components and also to buses. The bus forms connections between the hardware components such as processors, memory, and devices. They are the communications mediums and are used for the exchange of data. They can also be connected to other buses.

The system is the solitary composite component. A system is a consolidation of software, hardware components as well as other systems. It allows the software and hardware components to be arranged in explicit hierarchical arrangement with well-defined semantics. A system may be connected to other system via a data or a bus component.

AADL consists of well-defined declarations in the form of component types and component implementations. A component type defines the functional interface between the components. It consists of flow properties and features. While a component implementation encompasses the properties of subcomponents, connections between the subcomponents, error properties such as transitions and propagations. It complements the component type definition to build the system hierarchy.

The correlation between a component and its external environment occurs due to ports. A port can be classified as an 'in' port and an 'out' port according to the information transferring through the component or into data port and event as per the characteristic of the signal being conveyed.

The distinct components are opted as per the requisite system architecture description. The components are the imperative constituents of the system definition and ports signifies the apparent relations amongst the components.

## VII. AADL: Error Model Annex

The SAE Error Model EMV2 Annex Standard is an extension to SAE AADL standard which is defined for providing effective safety analysis of the well-structured architectural model of the system. It allows the user to elucidate the architectural model with the failure models, fault propagation, failure effects, hazard analysis as well as the component and compositional error behaviours. The fault propagation doctrine of the Error Model Annex is exemplified in three levels of abstraction such as the fault propagation, the component failure behaviour and the composite failure behaviour [2] [3] [31].

The error model annex is defined for the architectural specific redundancy management and risk mitigation methods and the reliability, safety, integrity as well as the maintainability of the architectural system and its qualitative and quantitative evaluation. The annex model is used to define the varying error models in the error annex library and adjoin them with the corresponding architectural definitions. These error models are effectively utilised in the core AADL declaration [2] [3] [31].

The three levels of abstractions defined earlier can be detailed as follows [2] [3] [31]:

**The error propagation:** It is the linkage between a component and its external environment. This is designated by the error propagation paths which can be incoming as well as outgoing. Each error path influences the system in an exclusive manner. The propagation is further resolved as the error source of the propagation, error sink of the propagation or error path of the propagation passage through the component [2] [3] [31].

**The error component behaviour:** These are component specific behaviours. It consists of the error events, which can be a self-failure event and repair event. In conformity with the component error properties, the errors can be transformed into different forms. The errors can also be altered in the form that they are masked (i.e. the component is a sink for the error) or passed on in a different form (i.e. different error type) or in the same form. This ideology includes the following types of errors [2] [3] [31]:

**Commission and Omission errors:** These error types (Service Omission and Service Commission) are used to refer to the loss of command, loss of power or the sensor reading, etc. These are also referred by the terms NoValue, NoPower, etc. The terms utilized are user-centric and depend upon the definition of the error.

**Value Errors:** This particular error types represent the individual errors such as the Out of Range, Out of Bounds, Bounded Value Change, etc. These terms are typically recognized by the term InvalidValue errors. The errors also assign the characterization and description of the respective error.

**The error compositional behaviour:** These types of error behaviours are described for the entire system. The errors for the constituent components are defined and their events are specified. This behaviour is stated in the global syntax, which encompasses the interactions of the components and their properties. This is crucial for the development of the Reliability Lock Diagram and Fault Tree Analysis [2] [3] [31].

The characteristic error definitions of a system can be structured with the adequate collaboration of three given abstractions of the Error Annex Model. The distinctive error types of the components are employed as per their descriptions.

## VIII. AADL Dynamic Positioning System

We will model a generic DPS class II for a standard supply vessel quipped with

- 2 stern azimuths thrusters
- 2 bow tunnel thrusters

As schematically represented on the figure7, the DPS is connected to several sensors and other devices.

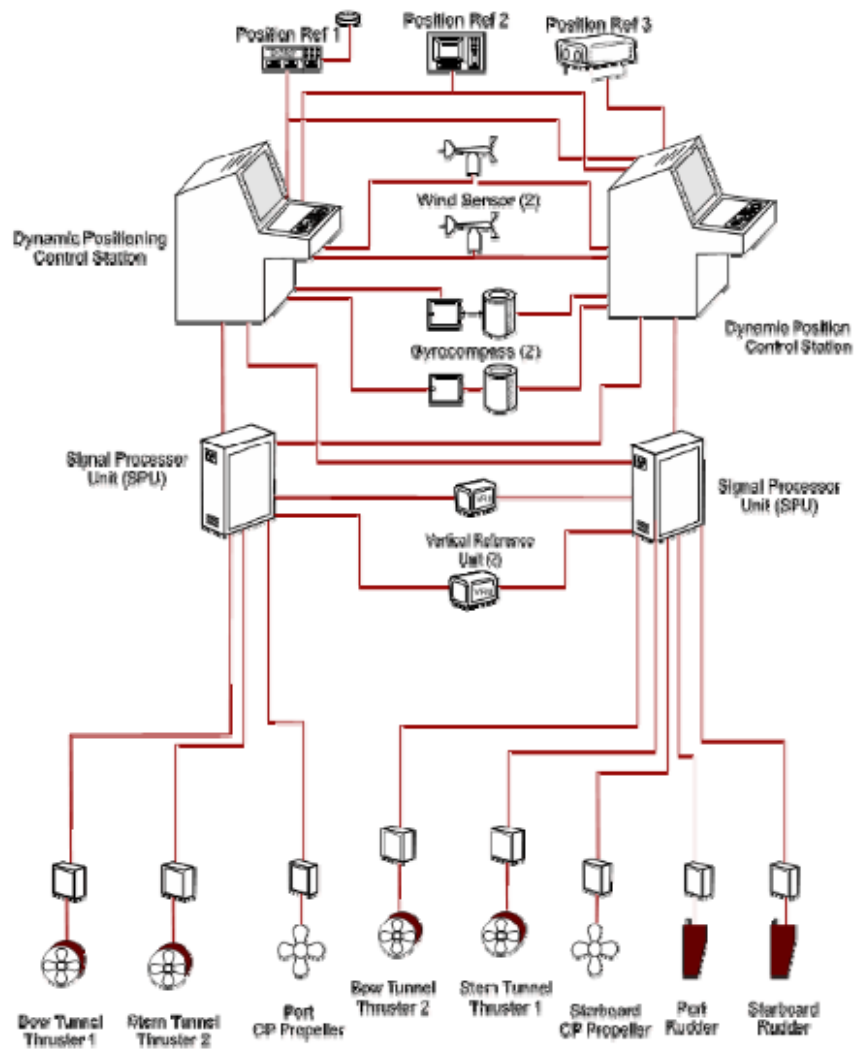


Figure 7: Generic configuration of a DP-2 Vessel [32].

Components of the DP control system are explained as follows:

**DP controller:** The DP controller runs the real time operating system in order to compute the control signals required by the vessel.

**DP HMI:** The HMI (Human Machine Interface) is used to give input to the controllers. It displays the results found by the controller.

**DP IJS (Independent Joystick):** This is an independent system from the DP controller. It receives information from only limited number of components as compared to the DP controller.

**DP UPS (Uninterrupted Power supply):** The DP system is provided with 3 UPS components in order to supply continuous power. The UPS\_IJS supplies power to the IJS system. All the modules are equipped with dual power supply, which makes sure that a failure will not result in loss of equipments.

Different types of sensors are used in the DP system. These are explained below:

**DGPS (Differential Global Positioning System) or DGNSS:** Differential Global Navigation Satellite systems are today very accurate reliable and widely used in DP operations.

**Laser Position Reference sensor (LPR):** This sensor utilizes the time and distance travelled by the laser beam from a sensor on the vessel to the reflector on the target to calculate position information of the vessel.

**Light taut wires (LTW):** Light weight taut wires are used to calculate the position of the vessel by utilizing the measurement of the angle of the wire. The wire is under strain due to a clump weight connected to the sea bed.

**Hydro-acoustic position reference (HPR):** The HPR sensor calculates the position of the vessel with respect to a target by measuring the range and heading from a transceiver mounted on vessel to a transponder connected to the target.

**Gyrocompasses:** Gyrocompass makes use of an electrically powered, fast spinning gyroscopic wheel as well as gravity and Earth's rotation to find the true north, which is used to find the vessel heading.

**Radar Sensor:** This sensor is a microwave based relative positioning reference system.

**Wind sensors:** It gives the information about wind speed and direction to the DP system.

**Motion Reference Units (MRUs):** Motion reference unit is a combination of accelerometers, gyros and magnetic sensors integrated with microprocessors to give the pitch and roll information to the vessel.

The DP is connected to the Power Management System (PMS).

The dynamic positioning system was modeled in the AADL by employing the customary syntax definitions. The components were detailed by using the system definitions in independent files and the correlations between the singular files were specified in an integration file.

The different types of analysis (FTA, FHA, Fault Impact, Consistency Checks, Unhandled Faults Analysis and Reliability Block Diagram) were accomplished. The subsequent sections provide a comprehensive explanation of the analysis methods and their results.

## IX. AADL: Analysis Methods

### IX.1 Functional Hazard Assessment (FHA):

FHA is an analytical tool which is employed in the conceptual phase to distinguish the system level safety analysis with functional hazards, and in the preliminary phase to categorize the subsystem level safety analyses with functional hazards [33]. It is a top-down method which scrutinizes system functions to recognize all potential failure conditions and classify the associated hazards.

FHA can also be exemplified as a standardized and extensive examination of functions to identify and classify failure conditions of those functions in accordance with their severity and likelihood [34]. It divides the failures according to their severity of the impact, their likelihood of occurrence, their description, the specific types of failures, etc.

The classification done by the MIL-STD-882 [35] and ARP 4761 standards are utilized in order to categorize the failures as per their severity and likelihood.

The MIL-STD-882 [35] is the recognized U.S Department of Defense military standard that provides a rational way to assess risks and maintains a standard practice for managing system safety. Risks are classified, evaluated and mitigated to a level that is satisfactory for the relevant authority.

ARP 4761 [36] is the distinguish Aerospace Recommended Standard from SAE International. It defines guidelines and methods of implementing the safety assessment for the civil aircrafts certification.

The Severity and Likelihood classification for MIL-STD-882 [9] [10] are given below:

- Severity : Catastrophic, Critical, Marginal, Negligible
- Likelihood : Frequent, Probable, Occasional, Remote, Improbable

The Severity and Likelihood classification for ARP 4761 [9] [10] are as follows:

- Severity : Catastrophic, Hazardous, Major, Minor, No Effect
- Likelihood : Probable, Remote, Extremely Remote, Extremely Improbable

The FHA is an iterative process. Therefore, it is conducted in extensive categories with the resolution enhancing as the analysis becomes finer. The premier step in FHA is to enlist all the failures and their properties. Thenceforth, the failures are categorized according to the pre-defined groups. This division is supplemented to the AADL-OSATE language and analysis is accomplished [37].

The figure 8 is a snapshot of the initial nine results obtained after the FHA analysis.

1	Component	Error	Hazard Description	Functional Failure	Severity	Likelihood	Comment
2	first_gyro_sensor	"BadValue on efa"	"Bad value from gyro sensor"	"BadValue"	Minor	Probable	"Alarm would be initiated but no immediate effect on position keeping capabilities"
3	first_gyro_sensor	"NoValue on efa"	"No heading information from gyro sensor. Th"	"NoValue"	Minor	Probable	"This can be due to the loss of gyrocompass; sensor telegram timeout or sensor rej"
4	first_gyro_sensor	"NoPower on efa"	"No power supply to gyro sensor."	"NoPower"	Minor	Remote	"Alarm would be initiated but no immediate effect on position keeping capabilities"
5	first_gyro_sensor	"HardwareFailure on efa"	"Failure of gyro sensor"	"HardwareFailure"	Minor	Probable	"Alarm would be initiated but no immediate effect on position keeping capabilities"
6	second_gyro_sensor	"BadValue on efa"	"Bad value from gyro sensor"	"BadValue"	Minor	Probable	"Alarm would be initiated but no immediate effect on position keeping capabilities"
7	second_gyro_sensor	"NoValue on efa"	"No heading information from gyro sensor. Th"	"NoValue"	Minor	Probable	"This can be due to the loss of gyrocompass; sensor telegram timeout or sensor rej"
8	second_gyro_sensor	"NoPower on efa"	"No power supply to gyro sensor."	"NoPower"	Minor	Remote	"Alarm would be initiated but no immediate effect on position keeping capabilities"
9	second_gyro_sensor	"HardwareFailure on efa"	"Failure of gyro sensor"	"HardwareFailure"	Minor	Probable	"Alarm would be initiated but no immediate effect on position keeping capabilities"
10	third_gyro_sensor	"BadValue on efa"	"Bad value from gyro sensor"	"BadValue"	Minor	Probable	"Alarm would be initiated but no immediate effect on position keeping capabilities"

Figure 8: Snapshot of initial nine results obtained after the Functional Hazard Analysis.

From figure 8, it can be observed that the first component ‘gyro sensor’ fails with the Bad Value Failure with severity ‘Minor’ and the likelihood ‘Probable’. The hazard description which gives the impact of the failure is ‘Bad Value from gyro sensor’ and the cause ‘Alarm would be initiated but there is no effect on the position keeping’ is described in the comment section.

The results obtained by the FHA are the recognition of failure properties such as hazards, modes and their description in detail. They facilitate the understanding the failures and the properties. The FHA results are considered to be outset of the safety hazard assessment methodology for a system [34].

### IX.2 Fault Impact Analysis:

Fault Impact Analysis is the analysis technique that is utilized to detect the path between the source of failure and the affected component. The immediate components that are affected by the failure path are also deduced [38].

The failure follows the connections between the components. The failure description, their conversion to the new failure and path followed by the failure is described in the syntax. Subsequently, the implementation is instantiated and fault impact analysis is executed. The result of the analysis is an excel file which contains details of the path followed by the failures.

The figure 9 is a snapshot of the initial nine results obtained after the Fault Impact analysis.

1	Fault Impact of System Internal Error Sources				
2	Component	Initial Failure Mode	1st Level Effect	Failure Mode	
3				second Level Effect	
4	first_gyro_sensor	{NoValue}	{NoValue} information -> DP_controller:first_gyro_input	DP_controller {NoValue}	{NoValue} second_ctrl_signal_to_azimuth -> second_azimuth_thruster
5	first_gyro_sensor	{NoValue}	{NoValue} information -> DP_controller:first_gyro_input	DP_controller {NoValue}	{NoValue} first_ctrl_signal_to_bow -> first_bow_thruster:thrust_input
6	first_gyro_sensor	{NoValue}	{NoValue} information -> DP_controller:first_gyro_input	DP_controller {NoValue}	{NoValue} second_ctrl_signal_to_bow -> second_bow_thruster:thrust_i
7	first_gyro_sensor	{NoValue}	{NoValue} information -> DP_controller:first_gyro_input	DP_controller {NoValue}	{NoValue} first_ctrl_signal_to_azimuth -> first_azimuth_thruster:thrust
8	first_gyro_sensor	{NoValue}	{NoValue} information -> DP_US:first_gyro_input	DP_US {NoValue}	{NoValue} second_ctrl_signal_to_azimuth -> second_azimuth_thruster
9	first_gyro_sensor	{NoValue}	{NoValue} information -> DP_US:first_gyro_input	DP_US {NoValue}	{NoValue} first_ctrl_signal_to_bow -> first_bow_thruster:thrust_input
10	first_gyro_sensor	{NoValue}	{NoValue} information -> DP_US:first_gyro_input	DP_US {NoValue}	{NoValue} second_ctrl_signal_to_bow -> second_bow_thruster:thrust_i
11	first_gyro_sensor	{NoValue}	{NoValue} information -> DP_US:first_gyro_input	DP_US {NoValue}	{NoValue} first_ctrl_signal_to_azimuth -> first_azimuth_thruster:thrust
12	first_gyro_sensor	{BadValue}	{BadValue} information -> DP_controller:first_gyro_input	DP_controller {BadValue}	{NoValue} second_ctrl_signal_to_azimuth -> second_azimuth_thruster

Figure 9: Snapshot of initial nine results obtained after the Fault Impact Analysis.





There is possibility of reusing the error description also. The inconsistencies in the error propagation may also lead to unhandled faults. Therefore, the importance of unhandled faults is paramount.

## IX.5 Fault Tree Analysis (FTA):

Fault tree analysis is a widely used method in the safety analysis and system reliability fields. It is the representation of the design functions which are utilized to recognize the path followed by a system hazard and to identify its feasible causes [41].

FTA is defined as an approach in which the aspects that lead to a certain objectionable event are recognized and classified in a plausible way [42]. It is also a very potent deductive tool which is employed to identify repulsive events and trace the path to their causes. It is one of the most widely used analysis method. It is highly recognized in the field of safety and reliability engineering. Substantial amount of information can be obtained from the fault tree analysis.

As the FTA is a deductive approach, the initial step is the identification of an undesired event. The next step is to recognize the failure path originating from this event. The path may pass through many components. The terminating point of the path is the top unacceptable event. This procedure creates a fault tree, which is a graphical interpretation of the failure path. The branches of the tree represent the contribution of that event to the top event [43].

The FTA provides the necessary information which is used to identify the potential contributors to the unacceptable top event failure. The dominance and contribution of any failure branches towards the top event can be deduced. The possible solutions of the failure results obtained from an FTA can be the adequate selection of the resources so as to curtail the chances of failure and failure probability of the top event. This type of analysis is very important as it is easier to detect the errors affecting the system through its tree structure [43].

The FTA employs the Boolean logic so as to give the description about the propagation of faults throughout the system. The composite error behavior is a critical property for this case. It expresses the error states of the components of the system and the Boolean relation amongst the components [27] [3].

The fault tree uses the 'or' and 'and' logic gates in order to find the culminating branch. The below given figure depicts the logic gate inferences.



Description	Picture	Truth table		
		Input A	Input B	Output
The "and" gate indicates the output occurs if all the input events are present.		T	T	T
		T	F	F
		F	T	F
		F	F	F
The "or" gate indicates the output occurs if at least one of the input events is present.		T	T	T
		T	F	T
		F	T	T
		F	F	F

Figure 11: Logic Gates used in fault tree analysis.

The figure given below is a snapshot of two branches of a fault tree analysis obtained.

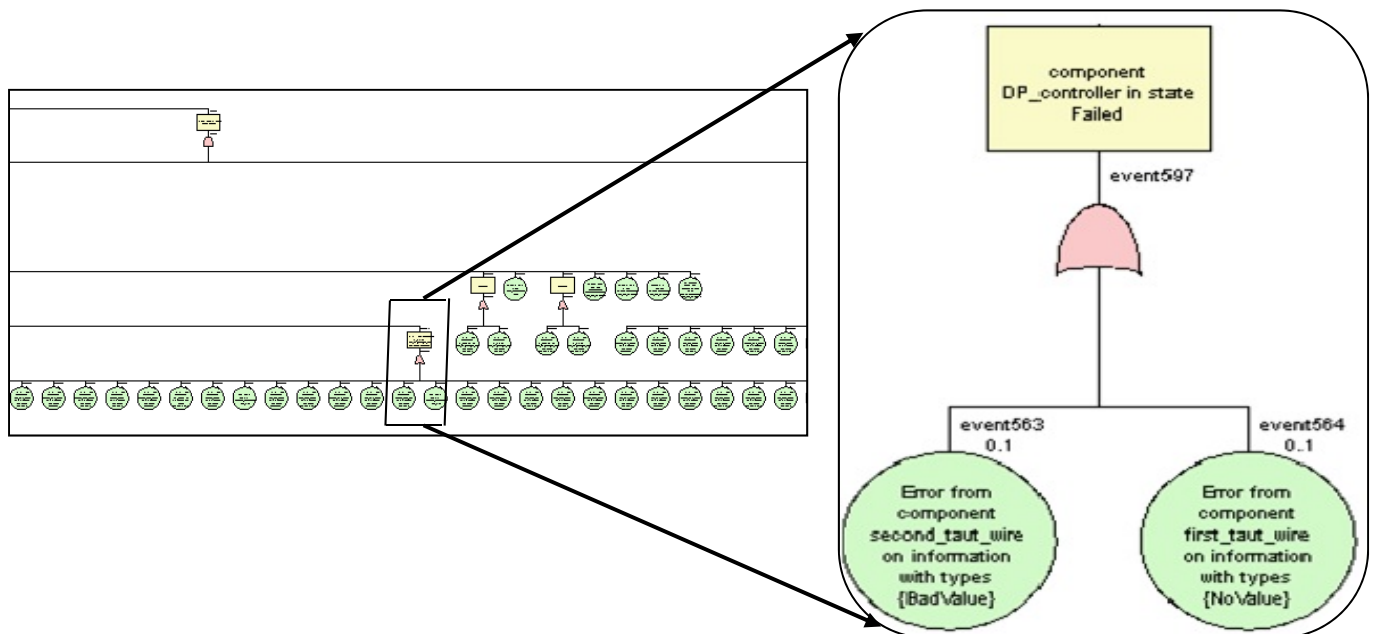


Figure 12: Fault tree analysis of the dynamic positioning system.

The above given fault tree gives the description of error flow from one branch to another. The failures 'BadValue' and 'NoValue' are from the two 'Taut Wires' to the component 'DP controller'.

The fault tree analysis is one of the most widely used analysis method. It is highly acknowledged in the field of safety and reliability engineering. Considerable amount of information can be gathered from the fault tree analysis.

### IX.6 Reliability Block Diagram (RBD):

A Reliability Block Diagram (RBD) provides the reliability/safety-related information about a system. It is a method which is used to infer that how failures of some components contribute to the combined system failure. The reliability block diagrams are used to study the reliability and dependability of the system components [44].

The RBD also depicts all the required functions that are paramount for the functioning of the system. The goal of the RBD is to show the relation between the constituent components of the system and their reliabilities. It is a quantitative method which makes it distinct from all the other analysis methods [45].

The Reliability Block Diagram analysis in AADL is used to calculate the overall failure probability of the components in the system. The failure probability of the individual components is mentioned in the composite error behavior of the system. The overall probability is calculated according to the connections between the components.

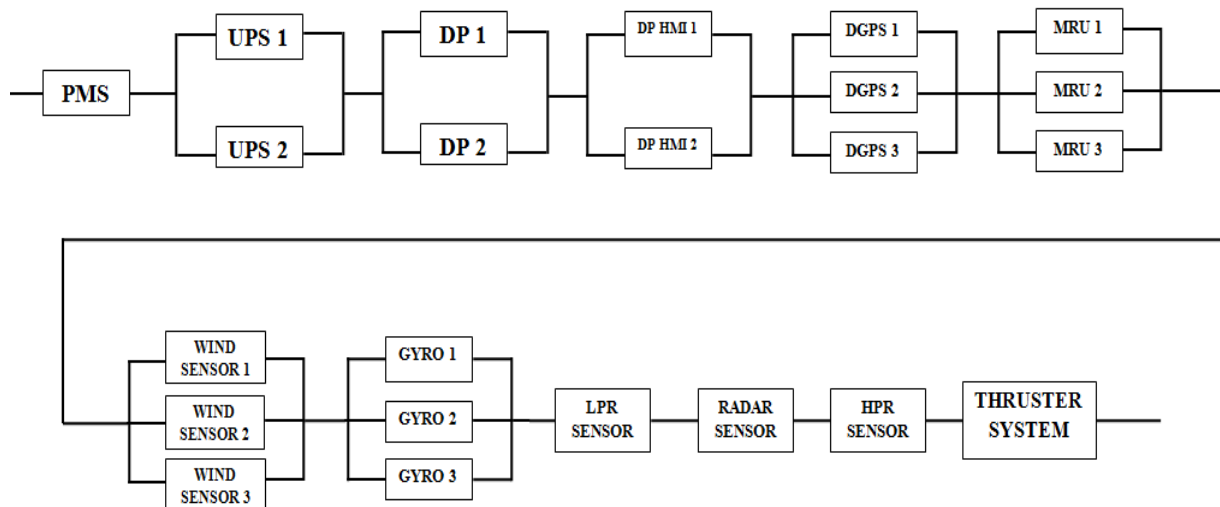


Figure 13: Reliability Block Diagram of the DP-2 system.

The above given figure describes the reliability block diagram of the DP-2 system. The components are arranged in series combination so as to assign equal weightage to their failures. Whereas the similar components are arranged in parallel arrangement in order to provide redundancy to the DP system. The thruster system comprise of series connection of bow and azimuth thrusters. The utilized DP- 2 system is a simple dynamic positioning system consisting of the most common components. The primary purpose behind this is to show the effectiveness of AADL analysis methods in investigating and scrutinizing the failures affecting a dynamic positioning system.

The following table gives the value of failure probability for the components of dynamic positioning system:

Component	Failure Probability
Gyro sensor	0.01
DGPS sensor	0.02
MRU sensor	0.01
Laser Position Reference Sensor	0.025
HPR Sensor	0.015
Wind Sensor	0.01
Radar sensor	0.02
DP control module	0.005
DP operating system	0.01
UPS	0.02
PMS	0.01
Bow Thruster	0.04
Azimuth Thruster	0.04

Table 1: Failure probabilities of Dynamic Positioning components.

The above given failure probability values in the table are just pure assumptions without formally proven figures.

#### X.6.1 Converting failure rate in E6 units to failure per hour from Non-Electronic Parts Reliability Data (NPRD) [46]:

$$\text{Failure rate with '1' E6 units} = 1 \times 10^{-6} \text{ per hours} = 0.000001 \text{ per hours}$$

#### X.6.2 Calculating number of failures per year [47]:

$$\text{Number of failures per year} = \text{Number of failures per hour} \times \text{Number of hours in a year}$$

$$\text{Number of failures per year} = \text{Number of failures per hour} \times (365\text{d} \times 24\text{h})$$

$$\text{Number of failures per year} = \text{Number of failures per hour} \times 8760\text{h}$$

### X.6.3 Converting failures per year (failure rate) into probability [48]:

$$P(t) = 1 - e^{(-\lambda t)}$$

Here:  $P(t)$  is the probability of failure at time  $t$ .  
 $\lambda$  is the failure rate (failures per year).

### X.6.4 Converting Mean time between failure (MTBF) to failure rate [49]:

$$\text{Mean time between failures (MTBF)} = \frac{1}{\text{Failure rate } (\lambda)}$$

### X.6.5 Converting failure probability to MTBF (in hrs):

Derivation:

$$P(t) = 1 - e^{(-\lambda t)} \quad [48]$$

$$e^{(-\lambda t)} = 1 - P(t)$$

$$-\lambda t \log e = \log(1 - P(t)) \quad (t \text{ is 1 year})$$

$$\lambda(\text{per year}) = \frac{-\log(1 - P(t))}{0.4342944819}$$

$$\lambda(\text{per hr}) = \frac{\lambda(\text{per year})}{365d \times 24h}$$

$$\text{MTBF(in hrs)} = \frac{1}{\lambda(\text{per hr})}$$

Here:  $\lambda$  is the failure rate.  $P(t)$  is the failure probability.  $t$  is time (one year).  $\log e = 0.4342944819$ .  
 The following are the documentation which can be used to obtain the failure probability and MTBF values of the electronic as well as non electronic devices.

Documentation	Short Description	Links for download
Telecordia documentation	This documentation contains the Failure Rates and MTBF information about common hardware electronic components in the communication industry. The failure rate can be converted into the failure probability by utilizing the formulae given before. The documentation can be accessed from the given link.	[50]
MIL-HDBK-217 handbook	MIL-HDBK-217 is a reliability prediction handbook published by the U.S Department of Defense. It consists of the failure rate models of the components used in electronic systems. The failure rate can be changed into the failure probability or MTBF by utilizing the formulae. The 1991 edition of the documentation can be downloaded from the given link.	[51]

Non-electronic parts reliability database (NPRD) documentation	The NPRD documentation published by U.S Navy provides the failure rates for a wide variety of electrical assemblies and electromechanical/mechanical parts and assemblies. The failure rate can be converted into the failure probability or MTBF by using the formulae. The 1991 edition of the documentation can be downloaded from the given link.	[52]
--	---	------

Table 2: Summary of documentation available to find Failure probability and MTBF values.

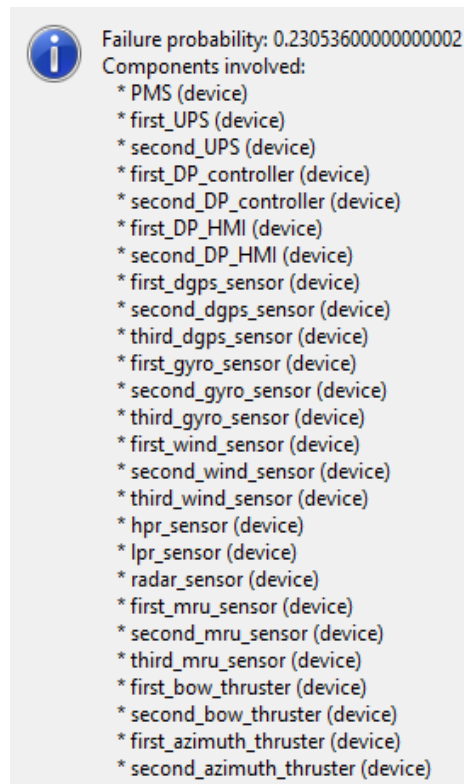


Figure 14: Snapshot of the failure probability calculation of Dynamic Positioning System.

The figure 14 depicts the failure probability value of the dynamic positioning system computed automatically using the AADL. The MTBF value found by converting the failure probability with the help of previously given formulae is 33433 hours or 3.8 years.

The importance of the reliability block diagram is that the comprehensive failure probability can be attained even if it is a a simpler model or a complex model. As the complexity of the system increases it becomes difficult to determine the failure probability manually but it can be conveniently obtained using the RBD.

## XI. Conclusion

In this paper, we have discussed about system safety analysis methods. These methods have advanced a lot and reached a higher stage of development. These are employed in numerous sectors. Risk management, hazard identification and analysis techniques are the vital constituents of the safety analysis methods. These methods are primarily responsible to retain and enhance system safety. System safety is paramount for the effective and adequate functioning of the system.

As the complexity of the systems increases, it is becoming difficult to detect failures and early phase of development. Therefore it is essential to detect these failures as the cost incurred at a later stage is enormous. There is also a need of transition from the manually worked upon analysis methods to the automatic analysis methods as the manual methods are becoming less efficient.

Along with the above given concerns, the drawback of the fault in a single component escalating to the entire system is also looked upon. Therefore, the SAE AADL is proposed as an appropriate solution for creating architecture model of the system and performing the various types of analysis methods. AADL contains well defined semantics which help in efficient architectural description as well as in the safety analysis methods. The chosen architectural abstractions can be evaluated and affirmed. This aids in enhancing the development process. The variety of components available with AADL gives an exemplary choice to the modeller to choose from. The execution nature of AADL is also beneficial in paying attention towards the inherent properties such as reliability, safety and performance of a system. AADL also consists of the exceptional tool support OSATE which provides excellent tools for the safety analysis.

AADL modeling was used for a generic dynamic positioning system and the safety analysis methods was performed on it. Different fault analysis methods provide varying types of analysis results. These are used to examine the faults from diverse point of views. The cause and effect of the faults can be traced to their roots. The graphical view obtained helps in enhancing the understanding of the system. All the possible types of faults can be discovered earlier. The analysis gives qualitative as well as quantitative results. Effective solutions can be deduced using this analysis.

This work is a part of the collaborative project MADNESS project – Modeling, Analysis & Description of Marine Embedded Systems lead by D-ICE ENGINEERING, ELLIDISS TECHNOLOGIES and the laboratory UBO/STICC/UMR/CNRS/6285.

## XII. Acknowledgements

The authors acknowledge their affiliated companies, D-ICE ENGINEERING, ELLIDISS TECHNOLOGIES and the laboratory UBO/STICC/UMR/CNRS/6285 for their allowance to publish the paper.

The work has been influenced by the excellent advices and comments of members of the SAE AADL online forum and the mailing lists.

## XIII. References

- [1] A. International, "AS5506 - Architecture Analysis and Design Language (AADL)," 2012.
- [2] A., Vestal, S., & Binns, P. Joshi, "Automatic generation of static fault trees from AADL models. In Workshop on Architecting Dependable Systems of The 37th Annual IEEE/IFIP Int. Conference on Dependable Systems and Networks," Edinburgh, UK, June 2007.



- [3] J., & Feiler, P. Delange, "Architecture fault modeling with the aadl error-model annex. In 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications (pp. 361-368). , " *IEEE*, August 2014.
- [4] P. H. Feiler, "Challenges in validating safety-critical embedded systems. *SAE International Journal of Aerospace*, 3(2009-01-3284), 109-116.," 2009.
- [5] G. Tassef, "The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology, RTI Project, 7007(011).," 2002.
- [6] Delange J, "Architecture Analysis with AADL: The Speed Regulation Case-Study," SEI-Carnegie Mellon University, 2014.
- [7] D. incident. [Online]. <http://www.imca-int.com/media/50897/imcasf06-02.pdf>
- [8] A. D. Incident. [Online]. <http://www.imca-int.com/media/50900/imcasf07-02.pdf>
- [9] S. A. E. Aerospace, "SAE Architecture Analysis and Design Language (AADL) Annex Volume 1: Annex A: Graphical AADL Notation. Annex C: AADL Meta-Model and Interchange Formats, Annex D: Language Compliance and Application Program Interface Annex E: Error Model Annex, AS5506/1.," 2011.
- [10] S. A. E. Aerospace, "SAE Architecture Analysis and Design Language (AADL) Annex Volume 2: Annex B: Data Modeling Annex Annex D: Behavior Model Annex Annex F: ARINC653 Annex. AS5506/2.," 2011.
- [11] A. History. [Online]. [https://wiki.sei.cmu.edu/aadl/index.php/The\\_Story\\_of\\_AADL](https://wiki.sei.cmu.edu/aadl/index.php/The_Story_of_AADL)
- [12] Seibel J., Wrage L. Feiler P, "What's New in V2 of the Architecture Analysis and Design Language Standard?," 2012.
- [13] D. De Niz, "Diagrams and languages for model-based software engineering of embedded systems: UML and AADL. White Paper, www. sei. cmu. edu/ library.," 2007.
- [14] D. System. [Online]. <http://www.imca-int.com/marine-division/dynamic-positioning.aspx>
- [15] [Online]. <http://www.le.ac.uk/eg/fss1/real%20time.htm>
- [16] E. System. [Online]. <http://internetofthingsagenda.techtarget.com/definition/embedded-system>
- [17] D. Gardner, "Architecture Analysis and Design Language: An Overview.," 2011.
- [18] Gluch, D. P., Hudak J. J. Feiler P., "The architecture analysis & design language (AADL): An introduction (No. CMU/SEI-2006-TN-011).," Software Engineering Inst., Carnegie-Mellon Univ Pittsburgh PA, 2006.
- [19] Feiler P. Hudak J., "Developing aadl models for control systems: A practitioner's guide.," 2007.
- [20] More AADL Advantages. [Online]. [https://books.google.fr/books?id=T\\_nOltBoX64C&pg=PT60&lpg=PT60&dq=Advantages+of+AADL&source=bl&ots=pgxl0nBMpG&sig=vj3JPRjyn-qZqZjTSicJ5f53P1k&hl=fr&sa=X&ved=0ahUKEwiF85fM4dfOAhXFfRoKHfwjBXg4ChDoAQgiMAE#v=onepage&q=Advantages%20of%20AADL&f=false](https://books.google.fr/books?id=T_nOltBoX64C&pg=PT60&lpg=PT60&dq=Advantages+of+AADL&source=bl&ots=pgxl0nBMpG&sig=vj3JPRjyn-qZqZjTSicJ5f53P1k&hl=fr&sa=X&ved=0ahUKEwiF85fM4dfOAhXFfRoKHfwjBXg4ChDoAQgiMAE#v=onepage&q=Advantages%20of%20AADL&f=false)
- [21] AADL Advantages. [Online]. <http://www.sei.cmu.edu/architecture/research/model-based-engineering/aadl.cfm>
- [22] Grant. E.S. Reza H, "Toward Extending AADL-OSATE Toolset with Color Petri Nets (CPNs)," *IEEE*, 2009.
- [23] OSATE details. [Online]. [https://books.google.fr/books?id=T\\_nOltBoX64C&pg=PT449&lpg=PT449&dq=AADL+osate&source=bl&ots=pgxlIIEQmH&sig=TvevlsaVxuBl0VUikDpILjQv2bs&hl=fr&sa=X&ved=0ahUKEwiG2e2m1s3OAhXCSR0KHb3yBJM4ChDoAQgyMAQ#v=onepage&q=AADL%20osate&f=false](https://books.google.fr/books?id=T_nOltBoX64C&pg=PT449&lpg=PT449&dq=AADL+osate&source=bl&ots=pgxlIIEQmH&sig=TvevlsaVxuBl0VUikDpILjQv2bs&hl=fr&sa=X&ved=0ahUKEwiG2e2m1s3OAhXCSR0KHb3yBJM4ChDoAQgyMAQ#v=onepage&q=AADL%20osate&f=false)

- [24] Kelly T. P. Wilkinson P. J., "Functional hazard analysis for highly integrated aerospace systems. In Certification of Ground/Air Systems Seminar (Ref. No. 1998/255), IEE (pp. 4-1). IET.," February 1998.
- [25] FTA. [Online]. <http://www.weibull.com/basics/fault-tree/>
- [26] C. Checks. [Online]. <http://aadl.info/aadl/osate/osate-doc/osate-emv2/consistency.html>
- [27] J., Feiler, P., Gluch, D., & Hudak, J. J. Delange, "AADL fault modeling and analysis within an ARP4761 safety assessment.," 2014.
- [28] RBD. [Online]. <http://www.reliabilityeducation.com/rbd.pdf>
- [29] Weiss K. A. Evensen K. D., "A comparison and evaluation of real-time software systems modeling languages. In Aerospace Conference, Georgia, Atlanta.," April 2010.
- [30] A. summary. The SAE Architecture Analysis Design Language (AADL) Standard: A Language Summary. [Online]. <http://www.aadl.info/aadl/downloads/papers/AADLLanguageSummary.pdf>
- [31] Feiler P, "SAE AADL Error Model Annex: An Overview," Software Engineering Institute, Carne, Pittsburgh, PA , 2011.
- [32] EBrahimi. A., "Effect analysis of Reliability, Availability, Maintainability and Safety (RAMS) in design and operation of Dynamic Positioning (DP) systems in floating offshore structures. ," October 2010.
- [33] M. S., "System Safety M7 Functional Hazard Assessment (FHA) V1.2," 2015.
- [34] K. J., Maddalon, J. M., Miner, P. S., Szatkowski, G. N., Ulrey, M. L., DeWalt, M. P., & Spitzer, C. R. Hayhurst, "Preliminary Considerations for Classifying Hazards of Unmanned Aircraft Systems," 2007.
- [35] /MIL-STD-882E. [Online]. <http://www.system-safety.org/Documents/MIL-STD-882E.pdf>
- [36] ARP4761. [Online]. <http://standards.sae.org/arp4761/>
- [37] Springer, "An Overview of System Safety Assessment,".
- [38] F. Impact. [Online]. <http://www.aadl.info/aadl/osate/osate-doc/osate-emv2/fault-impact.html>
- [39] Consistency Checks. [Online]. <http://aadl.info/aadl/osate/osate-doc/osate-emv2/consistency.html>
- [40] U. Faults. [Online]. <http://aadl.info/aadl/osate/osate-doc/osate-emv2/unhandled-faults.html>
- [41] Y. L. Cheng, "Uncertainties in fault tree analysis. 淡江理工學刊, 3(1), 23-29.," 2000.
- [42] F. T. Analysis. [Online]. <https://www.ferc.gov/industries/hydropower/safety/initiatives/november-workshop/fault-tree-analysis.pdf>
- [43] Vesely W., Dugan J., Fragola J., Minarick J., Railsback J. Stamatelatos M., "Fault tree handbook with aerospace applications," NASA Office of Safety and Mission Assurance, NASA Headquarters, Washington DC, 2002.
- [44] R. B. Diagram. [Online]. <http://www.isograph.com/software/reliability-workbench/rbd-analysis/>
- [45] E. R. Chelson P, "Reliability Computation from Reliability Block Diagram," 1971.
- [46] Convert failure rate in E6 units to failures per hour. [Online]. <https://www.rmqs.org/question/fail-per-e6-units-4/>
- [47] Calculate to number of failures per year. [Online]. <http://math.stackexchange.com/questions/93227/calculate-probability-of-a-failure>
- [48] Failure\_rate\_to\_probability. [Online]. <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20060024007.pdf>
- [49] Convert MTBF to Failure Rate. [Online]. <http://www.reliabilityeducation.com/ReliabilityPredictionBasics.pdf>
- [50] T. documentation. [Online]. <http://www.weibull.com/hotwire/issue152/hottopics152.htm>

- [51] M.-H.-2. documentation. [Online]. <http://www.sre.org/pubs/Mil-Hdbk-217F.pdf>
- [52] Greg Chandler, William Crowell, Rick Wanner William Denson, "Nonelectronic parts reliability data," Reliability Analysis Center, Rome, 1991. [Online]. [http://www.mwfr.com/CS2/NPRD-91\\_a242083.pdf](http://www.mwfr.com/CS2/NPRD-91_a242083.pdf)