A Maintainability Analysis of Dependability Evaluation of an Avionic System using AADL to PNML Transformation

Qaiser Mehmood

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full-time studies.

Contact Information:
Author(s):
Qaiser Mehmood
E-mail: qame09@student.bth.se

University advisor:
Prof. Mikael Svalnberg
Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden
Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
Abstract

Context. In the context of Software Architecture, AADL (Architecture Analysis and Design Language) is one of the latest standards (SAE Standard AS5506) used for analyzing and designing of architectures of software systems. Dependability evaluation of an avionic system, modeled in AADL, is conducted using petri nets standard PNML (ISO standard ISO/IEC15909-2). A maintainability analysis of PNML dependability model is also conducted.

Objectives. In this study we investigate maintainability analysis of PNML dependability model of an avionic system designed in AADL. Structural, functional, fault-tolerance and recovery dependencies are modeled, implemented, simulated and validated in PNML. Maintainability analysis with respect to ‘changeability’ factor is also conducted.

Methods. This study is a semi-combination of 'case-study' and 'implementation' research methodologies. The implementation of case-study system is conducted by modeling the case-study system in AADL using OSATE2 tool and simulating the dependability models in PNML using Wolfgang tool. PNML dependability models are validated by comparing with GSPN dependability models of previously published research.

Results. As a result of this research, PNML dependability model was obtained. The difficulties that influenced the research in AADL Error Model Annex and the OSATE2 tool are also analyzed and documented. PNML and GSPN are compared for complexity. And maintainability analysis for PNML dependability model w.r.t ‘changeability’ factor is also an outcome of this research. This research is recommended for software testing at architecture level as a standardized way for testing the software components for faults and errors and their impact on dependable components.

Conclusions. We conclude that PNML is an ISO standard and is the alternative for GSPN for dependability. Also, AADL Error Model Annex is still evolving and there is a need of availability of proper literature publicly for better understanding. Also, PNML dependability model possesses the ‘changeability’ factor of maintainability analysis and therefore it is able to adapt changes in the architecture. Also, dependability factors of a software can be tested at architecture level using the standards; AADL and PNML.

Keywords: AADL, PNML, Dependability evaluation, Maintainability analysis.
# Contents

**Abstract**

1 **Introduction**  
  1.1 An introduction to AADL  
  1.2 AADL Dependability Evaluation and Petri Nets  
  1.3 Brief Concept of PNML  
  1.4 Need of Case Study  

2 **Background**  
  2.1 Software Architecture and Model-driven Engineering  
  2.2 Analyzing Languages used for Model-driven Engineering  
    2.2.1 UML  
    2.2.2 SysML  
    2.2.3 EAST-ADL  
    2.2.4 AADL  
  2.3 Dependability Evaluation in Model-driven engineering  
  2.4 Related Researches  

3 **Methodology**  
  3.1 Research Methodologies  
    3.1.1 Interview  
    3.1.2 Survey  
    3.1.3 Case Study  
    3.1.4 Implementation  
  3.2 Literature Analysis  
  3.3 Research Questions and Data Collection  
  3.4 Motivation for Research Questions  
    3.4.1 RQ1:  
    3.4.2 RQ2:  
    3.4.3 RQ3:  
    3.4.4 RQ4:  
  3.5 AADL Model for French Air Traffic Control System  
  3.6 AADL Error Model Annex: Syntax Comparison  
  3.7 AADL to PNML Transformation Rules
3.8 AADL to PNML Dependability Evaluation ........................................... 18  
3.9 Maintainability Analysis of Dependability Evaluation ........................... 20  
3.10 Validity Analysis .............................................................................. 20  

4 Execution ............................................................................................... 21  
4.1 Sub-System of French Air Traffic Control System ............................... 21  
4.1.1 System Description ........................................................................ 22  
4.2 AADL model of French Air Traffic Control System ............................. 25  
4.3 AADL Error Model Annex to PNML Transformation ............................ 25  
4.3.1 Transformation Rules ..................................................................... 26  
4.4 PNML Dependability Model ................................................................. 29  
4.5 PNML Model Validity Mechanism ....................................................... 32  
4.6 PNML model for Structural Dependencies ........................................ 33  
4.7 PNML Model for Functional Dependencies ....................................... 36  
4.8 PNML Model for Fault-Tolerance Dependencies ................................. 37  
4.9 PNML Model for Recovery Dependency ........................................... 39  
4.10 Maintainability Analysis of PNML Dependability Model .................. 40  

5 Results .................................................................................................... 43  
5.1 PNML Dependability Model: an ISO standard .................................... 43  
5.2 Maintainability Analysis of PNML Dependability Model .................... 44  
5.3 AADL Error Model Annex and OSATE2 tool: Difficulties and Problems ........................................................................................................... 44  
5.4 Complexity of GSPN in comparison to PNML for Avionic Systems ...... 45  
5.5 Software Testing at Architecture Level using Standards ...................... 47  

6 Analysis and Discussion .......................................................................... 48  
6.1 Validity Threats .................................................................................... 48  
6.1.1 Construct Validity ........................................................................... 48  
6.1.2 Internal Validity ............................................................................. 49  
6.1.3 External Validity ............................................................................ 49  
6.1.4 Reliability ....................................................................................... 49  
6.2 Discussion ............................................................................................ 50  

7 Conclusion and Future Work .................................................................. 52  

A AADL Model of System .......................................................................... 58  
A.1 AADL Model for Flight Plan Component ........................................... 58  
A.2 AADL Model for Radar Data Component .......................................... 59  
A.3 AADL Model for Processors and Bus ................................................ 62  
A.4 AADL Model of Entire System .......................................................... 64
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>AADL Error Model to PNML transformation rules</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>AADL Architecture Model of Configuration 1</td>
<td>22</td>
</tr>
<tr>
<td>4.2</td>
<td>AADL architecture model of Configuration 2</td>
<td>23</td>
</tr>
<tr>
<td>4.3</td>
<td>AADL Error Model to PNML model component transformation</td>
<td>27</td>
</tr>
<tr>
<td>4.4</td>
<td>PNML Model of independent component in Wolfgang tool</td>
<td>28</td>
</tr>
<tr>
<td>4.5</td>
<td>XML Code of PNML model of component</td>
<td>28</td>
</tr>
<tr>
<td>4.6</td>
<td>Architectural Dependencies Block Diagram Config1</td>
<td>29</td>
</tr>
<tr>
<td>4.7</td>
<td>PNML Modeling an Independent Processor</td>
<td>30</td>
</tr>
<tr>
<td>4.8</td>
<td>PNML Modeling an Independent Bus</td>
<td>31</td>
</tr>
<tr>
<td>4.9</td>
<td>PNML Modeling and Independent sub-component (process)</td>
<td>32</td>
</tr>
<tr>
<td>4.10</td>
<td>PNML model for Structural Dependencies S1 and S3</td>
<td>34</td>
</tr>
<tr>
<td>4.11</td>
<td>Elaboration of Figure 4.10</td>
<td>35</td>
</tr>
<tr>
<td>4.12</td>
<td>Simulation Example</td>
<td>36</td>
</tr>
<tr>
<td>4.13</td>
<td>PNML Model for Functional Dependency F1</td>
<td>37</td>
</tr>
<tr>
<td>4.14</td>
<td>PNML model for Fault-Tolerance Dependency</td>
<td>38</td>
</tr>
<tr>
<td>4.15</td>
<td>PNML Model for Recovery Dependencies, R1 and R3 Config1</td>
<td>40</td>
</tr>
<tr>
<td>4.16</td>
<td>Changeability Test for PNML Recovery Dependency Model</td>
<td>42</td>
</tr>
<tr>
<td>A.1</td>
<td>Port Categories: AADL graphical Notation</td>
<td>58</td>
</tr>
<tr>
<td>A.2</td>
<td>Flight Plan component</td>
<td>58</td>
</tr>
<tr>
<td>A.3</td>
<td>Flight Plan Component AADL Implementation</td>
<td>59</td>
</tr>
<tr>
<td>A.4</td>
<td>Flight Plan AADL syntax</td>
<td>60</td>
</tr>
<tr>
<td>A.5</td>
<td>Radar Data Configuration 1</td>
<td>60</td>
</tr>
<tr>
<td>A.6</td>
<td>Radar Data Configuration 2</td>
<td>61</td>
</tr>
<tr>
<td>A.7</td>
<td>Radar Data Component AADL Implementation</td>
<td>61</td>
</tr>
<tr>
<td>A.8</td>
<td>Radar Data Component AADL Syntax</td>
<td>62</td>
</tr>
<tr>
<td>A.9</td>
<td>Systems Processors and Bus</td>
<td>63</td>
</tr>
<tr>
<td>A.10</td>
<td>AADL Model for Processors and Bus</td>
<td>63</td>
</tr>
<tr>
<td>A.11</td>
<td>AADL code for Processors and Bus</td>
<td>63</td>
</tr>
<tr>
<td>A.12</td>
<td>AADL Model of Subsystem of French Air Traffic Control System</td>
<td>64</td>
</tr>
</tbody>
</table>
The complexity of new-generation systems is increasing and the concerns related to dependability-related requirements, performance analysis and validation are an important issue [1]. Avionic systems are some of the most complex and safety-critical systems that are used in aerospace, road vehicles, aircraft, railways, nuclear systems and implanted devices [2]. Avionic systems being safety-critical [2], dependability-related information like fault assumption, repair assumption, fault tolerance mechanism, the occurrence of fault events and propagation, and characteristics of phases in phase mission systems [3] needed to be evaluated using international standards. Different engineering approaches have been used in industry during last decade to overcome the complexity of new-generation systems while improving the development approaches gradually [1]. Two such approaches that address the complexity issue during different phases of system development are component-based engineering and model-driven engineering [1]. Both approaches need a comprehensive support of languages and tools [1]. Model-driven approaches, being more recent, and being still under development, rely on UML (Unified Modeling Language) and/or ADLs (Architecture Description Languages) [1]. One such ADLs is AADL (Architecture Analysis & Design Language) [1]. It is important to mention that, functional and quality specifications of the system can be insured if the analysis related to dependability, performance and behavior is supported by the languages used in model-driven approaches [1]. Rugina [1], used AADL modeling language to architect an avionic system aiming to evaluate dependability factors using GSPN (Generic Stochastic Petri Nets) [1]. This research aims to use PNML (Petri Nets Markup Language), which is an ISO standard, instead of non-standard GSPN. The complete background of Rugina’s work is described in chapter 2. In this chapter we introduce AADL in section 1.1, AADL dependability evaluation and petri nets in section 1.2, brief description of PNML in section 1.3 and in section 1.4, the need of a case study.

1.1 An introduction to AADL

The Society of Automotive Engineers (SAE) provided a language in 2004 that was named as Architecture Analysis and Design Language (AADL), SAE stan-
Chapter 1. Introduction

As SAE AS5506, that is used for the formal specification and modeling of the software and hardware architectures of embedded computer systems especially avionic systems [3][4][5][6][7][8][9][10][11][12]. AADL provides components as a precise semantic way to describe the system’s architecture [4][13]. Each component has a unique name and runtime essence, expected interfaces with other components and some unique properties that distinguish one component from the other [4][13]. The AADL was developed on the bases of experience of mapping MetaH language [14] with Acme (an Architecture Description Interchange Language) [15] that led to the development of extensions in standard form [4]. MetaH was developed to support the analysis and auto-generation of embedded avionics software systems [4][14]. Later on, the SAE committee developed the core AADL standard AS5506 in 2004 that was revised in 2009 [16]. In this standard, the expressed needs of the industry representatives were fulfilled by the addition of many language features [4]. These industry representatives belonged to aviation and aerospace system companies from all over the world including Boeing and Airbus [4].

With more and more complex computer systems being developed using component based developing approach, more challenges are being faced while integration of components and communications amongst them [4]. These challenges are visible when a software component is modified or a hardware component is substituted. This change can affect system safety, fault tolerance, bus utilization, processor utilization, system latency etc. [4]. Consequently, this affect can lead to the failure of software component’s performance critical qualities, as a result, correct output is compromised [1]. Same is the case with hardware component substitution can lead to the failure of the system if the integration is not properly done and communication of the new hardware component is not as accurate with all other components as it was with previous component [1]. Even if the replacement is done properly, and all the integrations are redone and system is working with substituted hardware, or modified software components, still this is costly and time consuming [1]. To avoid such problems in developing of complex safety sensitive systems there was a need to develop a standard architecture modeling language that is not only capable of capturing properties of safety critical systems at architecture level, but also be capable of the analysis approaches so that critical quality requirements can also be evaluated at architecture level [1].

AADL was developed exactly for such intentions to face such challenges in complex safety critical systems [1]. AADL is designed to describe and analyze the real-time, embedded, safety critical software intensive and fault tolerant systems that belong to the fields of applications like avionics, industrial, automotive, medical and autonomous systems etc. [1]. AADL is a platform that unifies a Model-driven architecture of the system, the runtime architecture with providing communications amongst the components, the software components and their inter-component and subcomponent communications, and hardware components and their communications to other components including software components.
AADL, like all other programming languages, has a unique syntax that describes the software components in the form of process, thread, thread group, subprogram and data. Hardware components are described in the form of processor, bus memory and devices. Communications amongst the components are defined in terms of data and event flow that shows the interaction among the components and the flow of data and events information. AADL describes the architecture of any system with its implementation details both in the syntax form and graphical representation. The details of the language syntax will be further illustrated in Chapter 4.

1.2 AADL Dependability Evaluation and Petri Nets

As the aim of this research is to test PNML for dependability evaluation instead of GSPN, it is important mention that dependability is achieved by transforming AADL model to GSPN model by Rugina. While using PNML instead of GSPN, for dependability evaluation, there is a need to describe dependability evaluation at the first place. According to A. Rugina and P. Feiler, dependability-related information includes fault assumption, repair assumption, fault tolerance mechanism, the occurrence of fault events and propagation, and characteristics of phases in phase mission systems. It is important to mention that AADL is capable of conducting dependability modeling using Error Model Annex, yet it is incapable of describing and simulating complex dependability models that reflect real-life systems, especially with multiple interactions amongst the components and multiple dependencies. The entire AADL model is needed to be transformed to GSPN (Generic Stochastic Petri Nets) model in order to conduct dependability evaluation because GSPN supports systematic construction and validation of dependability models. However, GSPN is not a Petri Net standard, instead, PNML (Petri Net Markup Language) is the ISO standard for petri nets as ISO/IEC15909-2. Therefore, to achieve the dependability evaluation of a system modeled in AADL, the use of PNML is a standardized way. It is important to mention that A. Rugina and P. Feiler have described four steps to achieve the dependability evaluation of a system. These steps include 1) Modeling of the system architecture in AADL, 2) Modeling of system’s behavior in the presence of faults through AADL error model. 3) Generating global analytical dependability model in the form of GSPN. 4) This GSPN model is processed and dependability measures are obtained. Since our research is aimed to use PNML instead of GSPN so Sstep 3 and 4 of above model will be replaced with PNML instead of GSPN in proposed research. There are two major reasons for this variation; 1) PNML is an ISO standard and 2) it is also recommended by Rugina and Feiler to use PNML instead of
GSPN for future research [3]. It is important to mention that AADL to PNML transformation conducted in this proposed research is based on the principles of AADL to GSPN transformation as conducted by [1]. However, PNML being ISO standard, is not only syntactically different due to its XML properties but also the implementation and simulation mechanism of PNML is different than GSPN. Moreover, tools support and file formats are also different. Operating systems reusability is more supported with PNML than GSPN as PNML can be saved in .pnml format and the file can be read using any built in notepad tool in many operating systems. Furthermore, the tools used by [3] for GSPN is named as SURF-2 which is so outdated that the official website of SURF-2 tool was last updated in 1997. Although the tool is available for Solaris operating system, but the updated version of this system does not support nineteen years old tool. It is also important to mention that although there are other tools available that support GSPN, yet PNML being ISO standard for petri nets is more authentic to use than GSPN, especially while considering the safety-criticality of avionic systems. These above mentioned differences will be clearer to understand in Chapter 4, when the execution of transformation will be discussed.

1.3 Brief Concept of PNML

PNML is an interchange format of Petri Nets and it is designed to be platform and tool independent [18]. PNML is design to support different dialects of Petri Nets due to extensibility [18]. The extensibility of PNML allows incorporation of not only current versions of Petri nets but also the future versions. As PNML is designed to provide a format for different types of Petri nets as a syntactic language, therefore it includes the definition of different Petri Net Types. The evolving PNML conventions document guarantees the compatibilities between different Petri Nets types. For further syntactic details and concepts of PNML convention document and Petri net type’s definition, the work of Jonathan, et al. Billington [18] is referred.

1.4 Need of Case Study

To focus the research is upon dependability evaluation (and maintainability analysis latter on) of avionic systems in transformation of AADL models to PNML models; there is a need to use an avionic system as case study for conducting the research. Rugina [1] have already used an avionic system as case study (French Air Traffic Control System) for AADL to GSPN transformation. This research will use the same system for transformation of AADL to PNML. The major reason to use the same system is to compare the results of GSPN transformation with PNML transformation for validity. Also, the principles of transformation
Chapter 1. Introduction

from AADL to PNML will be same as that of AADL to GSPN; hence PNML will be tested against its complexity or easiness in comparison to GSPN which is also important to figure out because of PNML being standard. It is important to mention that Rugina’s [1] research used one standard that is AADL (SAE AS5506) for architecture analysis. GSPN is not an ISO or SAE standard. Therefore proposed research here use PNML on the same system as described by Rugina [1] using both standards languages; AADL as a SAE Standard (SAE AS5506) and PNML as an ISO standard (ISO/IEC15909-2).

In addition to dependability evaluation, maintainability analysis for different configurations of case study will also be conducted in this research. Four parameters of maintainability [19], as described under the ISO standard ISO/IEC 9126, will be considered for conducting maintainability analysis of proposed case study’s PNML dependability evaluation model. These four parameters are Analyzability, Changeability, Stability and Testability [19][20].

Next chapter will describe the details about the background of proposed research describing the concepts of software architecture and Model-driven engineering, the beginning of dependability evaluation concepts and languages for Model-driven engineering need for AADL, need of using petri nets, AADL to PNML related previous research and a description of French Air Traffic Control System; the case study.
On the basis of the introduction given in Chapter 1, it is important to mention that one of the focus of this research work in to enhance the previously done research by Rugina [1]. The enhancement is actually the proposed future work by Rugina and P. Feiler [3] that suggested PNML to be used instead of GSPN for dependability evaluation. In addition to use of PNML, this proposed research is different from Rugina’s and P.Feiler’s research as it also conducts maintainability analysis of obtained PNML dependability evaluation model of case study under four ISO standard parameters that include Analyzability, Changeability, Stability and Testability [19][20].

The background of proposed research is linked with the concepts of software architecture, Model-driven engineering, dependability evaluation of Model-driven engineering and languages and tools used to meet the standards and quality of dependability evaluation and maintainability analysis of a system. So there is a need to describe these concepts at first place to understand the need of proposed research and the missing gap in previously done research on this topic. Furthermore, as described in Chapter 1, avionics systems being some of the most safety-critical systems, dependability evaluation is highly critical for such systems. Therefore, it is important to understand the links among software architectures, Model-driven engineering and dependability evaluation concepts in the context of avionic systems.

2.1 Software Architecture and Model-driven Engineering

The concept of ‘Software Architecture’ is defined by IEEE Standard IEEE 1471, as the description of a system in the form of its components and the relationships amongst the components and to the system’s environment under the design and evolution principles of that system [21]. The concept of Model-driven Engineering (MDE) frameworks is based on component-based architectures independent of platforms and then transformation of these architectures to platform-specific models and eventually transformation to implementation [1]. Hence, in perspec-
tive of software engineering, MDE is dependent upon the software architecture of
the system to be engineered. The description of a system in form of components
is therefore similar to the concept of component-based engineering in which sys-
tems of large scale are build using small components that perform a particular
task and are possibly used and re-used on different platforms and systems [1].
The principles of MDE, in context of models of systems, focus on well-defined
notations and languages that follow the international standards in the process
of modeling and transformation of the system [1]. Some of such languages are
briefly described in the following section.

2.2 Analyzing Languages used for Model-driven
Engineering

The strength of Model-driven Engineering is the use of standards in terms of lan-
guages. There have been different languages, both standard and non-standard,
used for MDE based projects [1]. Some of these languages are UML (Unified Mod-
eling Language), SysML(System Modeling Language), EAST-ADL and AADL
(Architecture Analysis and Design language) [1].

2.2.1 UML

Unified modeling language is a diagram-based language to describe software in the
form of diagrammatic model of the software that include structural and behav-
ioral model in the form of diagrammatic elements [22]. UML describes system’s
structure (e.g., component, class, composite structure diagram) and behavior
(e.g., use case, activity, state machine diagrams). Although system’s components
and relationships are visible via diagrams’ elements in UML, still it is not an
executable language. It cannot be executed at runtime to view the behavior of
the system under certain constraints [23]. This incapability UML does not allow
the runtime manipulation and maintainability analysis, especially changeability
factor of the system.

2.2.2 SysML

SysML (System Modeling Language) is a domain-specific modeling language that
is based on UML but specifically designed for system engineers [24]. Although it
is rooted to UML, still it is designed to target wide variety of systems other than
software systems [24]. SysML includes new features, in addition to UML features,
that are specifically designed to meet the requirements of system engineers [24].
Like UML, SysML is also non-executable language at run-time [25]. So this
language is also incapable of analyzing changeability aspect of the system at run-
time. The behavior and the changeability factor cannot be studied at run-time as the language cannot be executed at run-time.

### 2.2.3 EAST-ADL

EAST-ADL (Electronics Architecture and Software Technology – Architecture Design Language) is an architecture description language used for the description of the architectures of embedded systems [26]. Unlike UML and SysML, EAST-ADL is executable but its domain is specific for system engineering and is capable only to model the hardware functional architectures of systems to hardware [27]. It does not support the modeling of software architectures [26]. Also, EAST-ADL supports dependability analysis of embedded systems but it uses the error behavior from AADL [26].

### 2.2.4 AADL

AADL, as described in Chapter 1 in detail, is better than all the above mentioned languages because of its three important features. First and most important of all is that AADL is a SAE standard (SAE AS5506) [1]. Secondly it is executable language and the system’s behavior with respect to its components and their mutual relationship in terms of communication of data, event and control can be analyzed and studied at run-time [1]. Thirdly, it has features of Error Model Annex that is used for dependability evaluation. Although EAST-ADL also has the dependability analysis features but those are limited to only hardware embedded systems [26]. Moreover, AADL can be used for both hardware embedded systems as well as software systems for modeling and analysis of the architectures and dependability analysis at runtime. These features of AADL not only make it the strongest of all the above mentioned languages but also usable for all variety of software and hardware systems irrespective of their complexity and safety-criticality.

Above discussion clarifies that AADL is the only language used in MDE that is capable of handling dependability evaluation of both software and hardware systems as complex as embedded systems and as safety critical as avionic systems.

### 2.3 Dependability Evaluation in Model-driven engineering

Model-based dependability evaluation helps to forecast faults in two ways depending upon the type of system to be evaluated; 1) Ordinal evaluation that aim to avoid failures mechanisms by ranking and identifying them. 2) Probabilistic evaluation that considers probabilities of particular dependability attributes [1]. The analytical models used for both of these approaches are different depending
upon the system itself with respect to its complexity and type [1]. By ‘type’ it
is meant that either system is reliability sensitive, availability sensitive, safety
sensitive or maintainability sensitive; four parameters used to figure out failure
and restoration of the a system [1]. The analytical models for dependability eval-
uation are of two type; 1) State –space models, and ,2) Non-state-space models
[1]. The state-space models are used for the dependability evaluation of large
and complex systems and to facilitate generation of such systems Rugina. A and
Feiler. P [3] used GSPN as a higher level specification language. PNML being
ISO standard and being supportive to any petri nets type, including GSPN [28]
will be used in our proposed research.

Following section describes the recent related researches that involve AADL
to GSPN and AADL to PNML transformations.

2.4 Related Researches

As described previously in Chapter 1, according to A. Rugina and P. Feiler [3],
dependability-related information includes fault assumption, repair assumption,
fault tolerance mechanism, the occurrence of fault events and propagation, and
characteristics of phases in phase mission systems. For dependability evaluation
of a system in AADL, four steps are described by A. Rugina [1], these steps
include 1) Modeling of the system architecture in AADL, 2) Modeling of system’s
behavior in the presence of faults through AADL error model. This behavior is
modeled stepwise component to component. In first step, each component is
modeled individually in presence of its own faults and events of repairing. Then
dependencies are modeled component-wise. Finally, overall system behavior is
modeled of all components that represents both fault and repair events of each
component as well as the overall environment. 3) Generating global analytical
dependability model in the form of Generalized Stochastic Petri Net (GSPN). 4)
This GSPN model is processed and dependability measures are obtained. The
process includes, i) Model’s syntactic and semantic validation and ii) model’s
quantitative dependability measures validation.

Above mentioned fourth step is carried out by Rugina [1] in a software tool
named as SURF-2. This tool is not only outdated but also currently not available
for major operating systems like Windows and Mac. The version available of
SURF-2 tool are only workable for open source operating system named as Solaris,
but since SURF-2 is not updated since 1997, so it is not able to be used in the
latest version of Solaris.

Rugina et al. [29] conducted a case study on a French Air Traffic Control
System and dependability evaluation was done using above mentioned four steps.
Moreover Rugina et al. [29] recommend the use of PNML (Petri Nets Markup
Language) instead of GSPN for dependability evaluation for future research be-
cause PNML is becoming an ‘extensible interchange standard’ for Petri nets.
Chapter 2. Background

Reza and Chatterjee [30] transformed an AADL model to a PNML model, yet dependability evaluation information is missing in this research. Furthermore, they recommend dependability evaluation as the future research by transforming AADL model to PNML model.

Research done by Reza and Chatterjee [30] used Cruise Control System CCS as case study to transform and AADL model to PNML. First, their study admired that direct mapping from AADL to PNML is difficult so they used an interchanged format for mapping as XML. This format was chosen because both AADL and PNML can be transformed to XML. At the first step, AADL-text to AADL-XML transformation was done. Then AADL-XML was transformed to PNML using XSLT templates. XSLT templates are facilitators of PNML in its XML format. The details of the transformation are referred as [30]. It is important to mention that this is the latest and only available AADL to PNML transformation with a clearly illustrated research gap as missing dependability information mapping in PNML.

In the next chapter, first, the methodology of steps involved in AADL modeling of Components of French Air Traffic Control System will be described, along with the description of the system itself and the components and subcomponents. Secondly, transformation rules for AADL to PNML will be described that will be on the same principles as those of AADL to GSPN transformation conducted by Rugina A. [1]. This is because, as described previously, that this research aims to extend the research work done by Rugina A. Feiler P. [3] in which the recommended the use of PNML instead of GSPN for the future research. Moreover, PNML being the ISO standard for petri nets, it is more authentic to use standards for dependability evaluation in general, and for avionic systems in particular as these systems are some of the most safety-critical systems today [2]. Thirdly, introduction of tools used for modeling AADL models and PNML models will be introduced along with the reasons that influenced this research to use these tools.
Chapter 3

Methodology

The main focus of this research is to conduct a dependability evaluation of a component-based system’s architecture of an avionic system. This is done under the internationally recognizable standards so that this research can be used for improvement of design of safety-critical systems in future. Analyzing and improving the safety features of a system before the start of the actual system’s development, especially for avionic systems, is also one of the objectives of this proposed research. As it is described in previous chapters that this research is based on the research gaps and future recommendations of previously conducted research by Rugina A. and Feiler P. [3], so the methodology, case-study system and data of system’s component-based configurations used in this research will be same as that of Rugina A. and Feiler P. [3]. The variation is the use of PNML instead of GSPN as a dependability evaluation tool. This is because this proposed research is intended to improve the previous research (using PNML instead of GSPN). Also, the purpose of use of same case-study system and data is to validate the results of proposed research (use of PNML) in comparison to the results of previous research (use of GSPN).

3.1 Research Methodologies

The research methodology followed herein is a mix-method approach. A part of ‘case-study methodology’ and ‘implementation methodology’ is combined to achieve this research. The details are described in coming sections. First, there is a need to discuss different types of research methodologies with respect to their relevance or irrelevance to this research.

There are different types of research methodologies [31] [32] used to conduct research. These include Interview, Case Study, Implementation, Survey and Experiment [31] [32]. A short description of these research methodologies as per guidance of [31] and [32] is given below so that it is easy to understand which research methodologies are used in this research and which are excluded.
3.1.1 Interview

In interview method, a set of questions is prepared and asked either using ‘Open Interview’ process or ‘Closed Interview’ process. Since this research is not related to asking questions to people, experts or organizations, therefore ‘Interview’ methodology is not used.

3.1.2 Survey

Survey is a statistical technique to analyze responses of multiple respondents \cite{32}. There are different limitations while conducting a survey as well as advantages \cite{32}. Our research does not match this research methodology type because there is not requirement of a survey to achieve aims and objectives of this research.

3.1.3 Case Study

A case can be an individual, group, institutions or community \cite{32,33}. Our research is based on an individual case. The case is an avionic system named as French Air Traffic Control System. A case study investigates the case to answer specific research questions in search of different types of scientific evidence, where, evidence lies in case setting and required to be abstracted and collated to get the answers to the research questions in the best possible way \cite{33}. An essential part of case study research is data collection (evidence collection) for many of the cases depending upon the nature of study and nature of case \cite{33}. There are different types of evidences mentioned by Bill Gilham \cite{33} that include, Documents, Records, Interviews, Observations and Physical Artifact. The documents in case of our research are the existing research done by Rugina \cite{1} and the related available literature that describes the terminologies, languages used (AADL, GSPN and PNML) and the other scientific literature that helps us understanding the dependability evaluation and maintainability analysis.

3.1.3.1 Relevance to this research

3.1.3.1.1 Selection of case:

The case selected in the same case previously used by Rugina \cite{1} because the aim of this research is to conduct dependability evaluation using PNML instead of GSPN. Therefore, we selected the same case to compare and validate our results.

3.1.3.1.2 Research Questions:

Research questions are formulated to investigate the case of the case –study. These research questions are formulated considering the case and all the research questions depends upon the case (Subsystem of French Air Traffic Control System). These questions are stated in section 3.3.
3.1.3.1.3 Evidence (Documents):

The evidence selected is the previously done research of Rugina [1]. Our research tends to use the evidence of dependability evaluation conducted by Rugina [1] using GSPN to validate the dependability evaluation conducted by this proposed research using PNML. The scientific literature to relevant terminologies such as dependability evaluation, AADL, GSPN, PNML, maintainability analysis and avionic system is gathered using literature analysis described in section 3.2.

Our research seeks an evidence that depends upon a sequence of steps of implementation and development that are required to be performed to obtain the answers to our research questions. For that, we need another type of research methodology also that help us completing our research. “Implementation” is one such research methodology.

3.1.4 Implementation

The main features of implementation methodology are developing new solutions, comparison with existing solutions and validity of proposed solutions reflected by the implementation [32]. The use of this methodology in our research completes the requirement of our research. Now, initial steps of the case-study methodology, plus, implementation methodology provide our research a complete mechanism to achieve our aims and obtain answers to our research questions.

3.1.4.1 Relevance to this research:

3.1.4.2 Developing New Solutions:

AADL model of subsystem of French Air Traffic Control System is developed using AADL development language in OSATE2 tool. AADL to PNML transformation rules are developed using knowledge of previously done research of Rugina [1]. PNML dependability model was implemented using PNML in Wolfgang tool. Maintainability analysis of PNML dependability model was conducted.

3.1.4.3 Comparison with Existing Solutions:

The outcome of AADL new model is compared with existing outcome of AADL model. The outcome of PNML dependability model is compared with outcome of GSPN dependability model previously developed.

3.1.4.4 Validity of Proposed Solutions:

The match of outcome of new AADL model with outcome of existing AADL model will validate the new AADL model to be correct. The match of outcome of simulation of PNML dependability model with the outcome of existing simulation
Chapter 3. Methodology

results of GSPN dependability model will validate PNML dependability model. A maintainability analysis will generalize the PNML dependability model.

3.2 Literature Analysis

Since this research mostly relies upon the previously conducted research of Rugina [1] [3], so this research does not require an extensive literature review for collection of data in advance. It is not required in this research to conduct a scientific literature review for exclusion and inclusion of literature. However, to understand the terminologies like AADL, PNML, GSPN, dependability evaluation, maintainability analysis and avionics systems, plus, definitions and explanations of scientific terms used within the obtained literature, there was a need of gathering of scientific literature. The query mostly consisted of these terminologies and also those terminologies that were not understood studying the found literature; for example, the definitions or certain terms. The literature was searched using search engine google scholar. Google scholar provides results (articles, journals, books, etc.) to query and the database name which contains the result. The required result (articles, journals, books, etc.) is then obtained from that database by opening the database using BTH databases website.

3.3 Research Questions and Data Collection

In terms of missing evidence, a case-study perspective [33], there is a need to describe what evidence is available and what is missing. The missing evidence is key to formulate the research questions [33]. The data available for this research is extracted by [1] and [3] which provides the following information. This is the previously existing data.

- A graphical and textual description of components of French Air Traffic Control System in two configurations including dependencies. It is important to mention that AADL syntax of the system is not available.

- A textual and syntax-based (AADL Syntax) description of Error Annax Model (Error Model Version 1) of components including dependencies. It is important to mention that syntax AADL used is no-more in use in latest versions of AADL (Error Model Version 2).

- Transformation rules of AADL to GSPN error model

- GSPN error model mapped with AADL error model for dependencies

Dependability evaluation model using AADL and GSPN Data missing in [1] and [3] that needed to be reconstructed for this research is as follows. It is important
to mention that following points are also important steps in this research in order to fulfill the aims and objectives of the research. Also, this data is missing in [1] and [3].

- An AADL syntactic executable description of the components of French Air Traffic Control System.
- Transformation rules for AADL to PNML
- Syntactical, executable and simulate-able PNML error model mapped with AADL error model for dependencies
- Dependability evaluation method using AADL and PNML
- Maintainability analysis of dependability evaluation under the new PNML model of system’s different configurations.

To obtain the missing evidence is the aim of this research. To achieve the aim, the objectives of this research are:

- To design, implement and simulate The ISO standardized PNML Dependability Model
- To conduct Maintainability Analysis of PNML Dependability Model
- To discover Difficulties and Problems involved in AADL Error Model Annex and OSATE2 tool.
- To discover complexity of GSPN in comparison to PNML in dependability model.

To fulfill the aim and objectives of the proposed research, following four research questions are formulated.

1. **RQ1**: How to design, implement and simulate The ISO standardized PNML Dependability Model?
2. **RQ2**: How to conduct Maintainability Analysis of PNML Dependability Model?
3. **RQ3**: While developing AADL model, what are Difficulties and Problems involved in AADL Error Model Annex and OSATE2 tool?
4. **RQ4**: While transforming from AADL to PNML model, what is complexity of GSPN in comparison to PNML in dependability model?
3.4 Motivation for Research Questions

The motivation of research questions is described below one by one

3.4.1 RQ1:

The main motivation of this research question is the previously conducted research of Rugina [3] where she recommended, for the future work, that PNML should be used instead of GSPN. Since PNML is an ISO standard for petri nets, therefore use of PNML will standardize the whole process of dependability evaluation. Avionic Systems being some of the most safety critical systems in the world [2] require a standardized way of architect, design and analysis. PNML, being ISO standard, will provide the system architects and designers a standardized way to simulate the PNML dependability model of the system and make analysis and decisions accordingly.

3.4.2 RQ2:

Rugina [1] used mostly two sub-components and one processors scenarios for conducting dependability modeling in GSPN for structural, functional and recovery dependencies. But there can be a possibility of other systems that have different processor to sub-component configuration. What if a system has three sub-components handled by one processor? To answer such questions and to generalize the PNML dependability model for any possible scenario, Maintainability analysis of PNML dependability model was conducted. Answer to this research question test the system by adding a sub-component to the system and look for error behavior on the system.

3.4.3 RQ3:

The AADL syntax of AADL error model of any kind of avionic system is not exclusively available in literature so that it can be directly reused, run or analyzed. Also, the only available book of AADL modelling [17] does not provide syntax to all AADL error models. There are two AADL error model versions; EMV1 (Error Model Version 1) and (EMV2 Error Model Version 2). So there was a probability that we face difficulties while developing AADL error model of our case system. Also, Rugina [1] used OSATE2 tool in 2006-2007. The version of OSATE2 tool at that time is old and outdated in comparison to the latest available and compatible with Windows Operating System. Most importantly, as to our knowledge, there is no evidence in literature that exclusively describe the difficulties and problems faced by any researcher who worked with both EMV1 and EMV2 of AADL Error Model Annex and OSATE2 tool. The answer to this
Chapter 3. Methodology

research question will provide the new knowledge to the literature that will show whether or not there are any difficulties and problems that exist.

3.4.4 RQ4:

As of our knowledge, so far there is no evidence in literature that compares GSPN and PNML in terms of complexity of any type. Answer to this research question will provide a comparison on the basis of the PNML dependability model obtained by the answer of RQ1. We shall be able to analyze the complexity of GSPN or PNML in terms of dependability evaluation. This will be a new knowledge for future researchers while making a choice depending upon their system of study.

The answers to these research questions can be obtained by following a step-wise process of research. These steps are mentioned one by one in the coming sections.

3.5 AADL Model for French Air Traffic Control System

Although two configurations of components of French Air Traffic Control System are described by Rugina A. [1] graphically and textually, yet the actual AADL code is not available for the system. To use the system for PNML transformation, the first step of this research is to reconstruct the executable AADL model of the system. An eclipse-based tool named as OSATE2 [34] was used to develop AADL model using AADL syntax. The complete textual, graphical and syntactical description of the system is explained in Appendix A.

3.6 AADL Error Model Annex; Syntax Comparison

The AADL Error Model Annex used by [1] and [3] is known as EMV1 (Error Model Version 1) and it is the first version of AADL error model [35]. EMV2 (Error Model Version 2) is an improved form of EMV1 but syntactically different than EMV1 [35]. Although Rugina A. [1] described the error model syntax for dependencies amongst the component of the case-study system, yet it is not executable and not supported by latest OSATE2 versions. Moreover, OSATE2 tool also do not support old EMV1 version and generates syntactic errors. The understanding of EMV1 syntax comparison to EMV2 syntax is not explicitly available in literature. So to process the research, an error model code for all the dependencies is needed to be re-written in EMV2 and executed to map it to PNML error model for dependencies. Related details of AADL error model
annex for components of French Air Traffic Control System is described in detail in Chapter 4.

3.7 AADL to PNML Transformation Rules

PNML by design is universal petri net language that is formatted to be readable and editable in a conventional text editor and is aimed to be as universal so that it can cover all versions and types of petri nets [28]. GSPN being designed for specific tasking, have a unique feature immediate and timed transactions which describes the time factor involved for a transition from one state to another [36]. Using PNML for dependability evaluation of avionic systems, flow of error from one state to another is analyzed during this research. As described previously, avionic systems are some of the most safety critical systems. So, any fault or error generated by any components or sub-components need to be analyzed seriously to avoid component or system malfunctioning. So, this research considers every error transition as serious as an immediate transition. If an error is triggered in any component or sub-component, it’s a potential threat to the entire system and it is treated as an immediate transition of error irrespective of the possibility of its delay. We consider system under worst case scenario. In worst case, an error will not be delayed when it is once triggered from its initial state. Although PNML does have a mechanism to make a delay in the transaction, yet most of the tools do not support this feature yet. Even if they do support, our system (avionic system) requires an immediate transition of error without any delay as the system is avionic and safety critical. So a worst case scenario is considered.

State of error in AADL error model annex is mapped with the ‘state’ of a PNML model and transition of error is mapped with ‘transition’ in PNML. The starting point of error is the ‘initial state’ and there is no delay in the transition. Figure 3.1 illustrates the mapping AADL to PNML mapping based on the AADL to GSPN mapping done by Rugina A. [1].

Implementation details of the transformation are described in Chapter 4.

3.8 AADL to PNML Dependability Evaluation

To obtain the dependability evaluation, Error model of French Air Traffic Control System is designed using the AADL and PNML to analyze and simulate the flow of errors in different configurations of system. AADL error model of the system is implemented using the eclipse based tool named as OSATE2 [37]. After the implementation of AADL model of the system, next step is to implement the AADL error model so that it can be mapped with PNML. The process of PNML error model is to initiate, generate and simulate the error in a particular component of the system and to analyze the impact of errors upon the other components in that
Chapter 3. Methodology

![AADL Error Model to PNML transformation rules](image)

<table>
<thead>
<tr>
<th>AADL Error Model Constructs</th>
<th>PNML elements</th>
<th>PNML Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Place</td>
<td></td>
</tr>
<tr>
<td>Initial State</td>
<td>Place with token</td>
<td>○</td>
</tr>
<tr>
<td>Event</td>
<td>PNML Transition</td>
<td></td>
</tr>
<tr>
<td>AADL Transition</td>
<td>Place to Place</td>
<td>○</td>
</tr>
<tr>
<td>Src_State-[Event]-</td>
<td>transition via arcs</td>
<td>□</td>
</tr>
<tr>
<td>&gt;Dest_State</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.1: AADL Error Model to PNML transformation rules

particular configuration of the system. This process relies upon the dependencies of the system within components modeled in AADL. Dependencies, in terms of AADL, are the propagations of control or data information from one component to another and the impact of this flow of information on the state (state is changed or not) of the sending or receiving component [1]. Dependencies, in term of PNML, are the propagation of token from one place to another via transition. This is corresponding to AADL component model considering the flow of information amongst the systems components [1]. The detailed implementation of the AADL and PNML model are described in Chapter 4. The tool used for PNML model design is recommended by the official PNML website and is named as Wolfgang [38].

As PNML is an ISO standard and it is designed to accommodate all types of petri nets with XML features of syntax and reusability of .pnml files on various operating systems [18], the state-space model in this research is used to analyze and simulate errors in different configurations of AADL model of French Air Traffic Control System. This model, being developed in PNML, is not only more authentic (being ISO standard) than GSPN model but also have syntax in XML format. This allows the users to read and edit the code in any operating system on a simple notepad file. Moreover, .pnml extension of file can be transported to any operating system environment and can be reused in PNML supported tools. These features of PNML make the error model simulations more authentic and reliable in terms of dependability evaluation in comparison to GSPN.
3.9 Maintainability Analysis of Dependability Evaluation

The focus of this research in perspective of maintainability analysis is to consider the ‘changeability’ factor of maintainability analysis. Although there are three more factors (analyzability, stability and testability) as well, yet the scope of this research under the time constraints is not sufficient enough to cover them all. ‘Changeability’ factor is studied for determining its impact on the system under different scenarios and configurations. Analyzability, stability and testability factors are recommended for future work. The details of the maintainability analysis are described in Chapter 4.

3.10 Validity Analysis

Since this research is the extension of previously conducted research of Rugina A. and Feiler P. [3], with the variation of PNML instead of GSPN, so it is assumed that AADL model and GSPN model have been validated already. The proposed research uses PNML model that is validated by comparing the simulations of PNML model with the outcomes of GSPN model. GSPN model outcomes are already available and published as [3]. So by comparing the results of PNML model simulations with GSPN model, PNML model is validated. The process of validation is based on the analysis of the dependability model designed in PNML using the Wolfgang tool. The metric of comparison is the concurrence of error flow and its propagation once it is triggered in any of the components of the case-study system. If the PNML model is able to generate the same propagation of errors as that was done by [3] using GSPN, the research will be validated. It will be validated that PNML can produce similar results as GSPN and being an ISO standard, it will be recommended to use for future researches. The details of the validation process are described in Chapter 4.
Chapter 4

Execution

This chapter focuses the implementations and executions of all the data presented in previous chapter. As described earlier in Chapter 3 section 3.1, the first part of the research is to construct the syntactical and executable AADL model of components of French Air Traffic Control System. As described previously, this system is used by [1] and [3] and our research use the system as it is in terms of two configurations, as done by [1] and [3] and same components and their inter-communications in terms of control and data signals are used.

The implementation of AADL Model with description of dependencies within components and sub-components of the system and AADL Error Model Annex to PNML Dependability Model transformation mechanism are described in this chapter. Section 4.1 describes the available system of components of French Air Traffic Control System. Section 4.2 then describes component by component implementation of the system in AADL with description of dependencies. Section 4.3 describes transformation rules of AADL Error Model Annex to PNML Dependability Model based on Rugina’s research [1]. Section 4.4 describes the implementation of PNML Dependability Model based on structural, functional, recovery and fault-tolerance dependencies.

4.1 Sub-System of French Air Traffic Control System

As described previously in Chapter 3, Section ?? that the system available is only in textual and graphical form and the AADL syntax of the system is missing. So the first step is to build the system in AADL again on the basis of available graphical and textual information. Figure 4.1 and Figure 4.2 show the graphical model of two configurations of components of French Air Traffic Control System. These figures are exact graphical representation of the system explained by [1] and [3] and because the AADL model reconstruct of our research is based on this exact same graphical representations, therefore these figures are exact copies of graphical representation presented by [3][1].

Before going into syntactical construction of AADL model, there is a need to
describe the system at the first place.

4.1.1 System Description

The system chosen for this research is exactly the same system that is used by [1] and [3] in their research. So the description of the system is also based on the description of [1] and [3].

The system chosen is a subsystem of French Air Traffic Control System that contains two pairs of software components and two hardware platforms. One pair is responsible for Flight Plans (FP) and the other one is responsible for Radar Data (RD). There are two components in each pair. Two components of Flight Plan (FP) are named as FP_Comp1 and FP_Comp2. Also, two components of Radar Data are named as RD_Comp1 and RD_Comp2.

Two processors, that are part of hardware architecture, are also part of the system and these processors run these pair of components. Radar Data (RD) is a real-time software system and it is responsible for processing the flight information received from Flight Plan (FP). RD is responsible for facilitating air traffic controller about the map of airspace. Flight Plan (FP) is also a real-time software system and it is responsible for processing flight plans and it facilitates air traffic controllers with all the necessary information regarding flight plans at real time. Both RD and FP are interdependent and a failure to communication between these two can cause system malfunction at real time with probable catastrophic
Chapter 4. Execution

circumstances. FP is dependent upon RD in a way that RD provides position, speed and altitude information of the airplanes to FP. So FP updates the flight plans of airplanes being in flight in that particular controlled air space. RD in dependent upon FP in a way that RD receives information about the maps from FP. This information helps RD to map the airplanes in the airspace to inform the controllers in case of any aircraft enter the space that miss match the flight plan. FP_Comp1 and FP_Comp2 are two components of identical functionality with one acting as primary component and other as backup component. Same is the case with RD_Comp1 and RD_Comp2.

![Figure 4.2: AADL architecture model of Configuration 2](image)

Figure 4.1 and Figure 4.2 show the diagrammatic description of the system. It shows that all the components are interdependent and the connections (flow of information) are visible amongst all the software and hardware components. This indicates that fault or failure originated in one component can affect the other components as they are interdependent. Two configurations are used for connectivity to the software components to the processors. In configuration1 primary component of FP, that is FP_Comp1, is processed by processor1 while secondary component or backup component of FP, that is FR_Comp2, is processed by processor 2. The difference between configuration 1 and configuration 2 is the connectivity of RD components to the processor. In configuration 1, the RD primary component, that is RD_Comp1, is connected to processor 2 and RD_Comp2 is connected to processor1 while in configuration2 RD_Comp1 is connected to processor1 and RD_Comp2 is connected to processor2. These two configurations are designed to conduct dependability evaluation of the system.
by analyzing the impact of the origination of an error in one component of the system upon the other components.

The interactions between the sub-components in the form of arcs represent the architectural dependencies amongst them. These dependencies are described component by component in the following sections as the AADL model is implemented. F1, F2, F3, FT1, S1, R1, S2 and R2 are architectural dependencies are explained below,

- F1 and F2 are functional dependencies between FP and RD. These are visible in Figure 4.1 and Figure 4.2. This indicates the RD in functionally dependent on FP. If error is generated in FP_Comp1 or FP_Comp2, it will propagate to RD. Also, the error generated in RD will not propagate in and of the FP sub-components.

- F3 is the functional dependency between FP_Comp1 and FP_Comp1. It is considered that the flow of error is always directed from primary component to secondary.

- FT1 is the fault-tolerance dependency. It is considered for the switch of roles between FP_Comp1 and FP_Comp2 if replica is acting as primary and it fails and the other one is error free, the roles are switched.

- S1 and S2 are structural dependencies. These are considered to be the hardware faults that can propagate and influence software components. If the error is originated in a processor it is considered to influence and propagate the respective attached FP component.

- R1 and R2 are recovery dependencies. It is considered that if FP_Comp1 fails, it cannot be recovered or restarted if the corresponding processor is also in the fail state.

It is important to mention that both AADL and PNML dependability models are based on the above mentioned dependencies. AADL error model annex implements the dependencies but the simulation at run-time is beyond the scope of AADL. To simulate and analyze the dependencies at run-time, PNML is used. AADL dependency model is mapped to PNML dependency model to simulate and analyze the impact of dependencies on the system, hence simulate the dependability model for real-time analysis of dependencies. In both configurations (Figure 4.1 and Figure 4.2) there are two sub-components bound to one processor (E.g. FP_Comp1 and RD_Comp2 and bounded to processor1 in Configuration1) with respect to structural dependencies always. Same is the case with recovery dependencies. This implies that while modeling the structural and recovery dependencies in PNML, any two sub-components and one processor combination is sufficient to analyze the structural and recovery dependencies.
Chapter 4. Execution

The first step towards the dependability model is to construct the AADL model of the system based on Figure 4.1 and Figure 4.2 and textual description explained above. Following section describes the syntactical details of AADL model using OSATE2 tool.

4.2 AADL model of French Air Traffic Control System

The AADL model is constructed using the OSATE2 tool on the basis of the available information as described in Section 4.1. The model is constructed component by component. Since it was the reconstruction of previously done work of Rugina A. [1] So the implementations details are described in Appendix A. The implementation was, although previously done by Rugina A. [1], yet it was necessary to re-build the model to achieve research aims.

Next section describes the AADL error model transformation mechanism to the PNML model. It is important to describe that the dependability evaluation is done on the basis of architectural dependencies that exist in the system. These dependencies include structural, functional, fault-tolerance and recovery dependencies.

4.3 AADL Error Model Annex to PNML Transformation

As previously described, AADL Error Model is used to implement the dependencies in AADL model. These dependencies are modeled on in and out propagations from the components and sub-components as visible if Figure 4.1 and Figure 4.2 in the form of F1, F2, F3, R1, R2, R3, R4, S1, S2, S3, S4 and FT1. Error Model Annex exists in two formats; EMV1 (Error Model Version 1) and EMV2 (Error Model Version 2). EMV1 is the older version of AADL Error Model Annex [35]. The major difference between both the versions is the syntax. Although both versions are aimed to perform the same task yet the syntax implementation mechanism of both the versions is totally different. Rugina A. [1] used EMV1, the older version throughout the development process of AADL error model annex for the system. OSATE2 tool does not support the EMV1 completely. Though it supports EMV2, yet the learning material for EMV2 syntax is not publically available and the available EMV2 does not produce the desired output on OSATE2 tool. Still, considering the fact that the Rugina A. [1] has already used EMV1 and her work is already been published, it is assumed that the use of EMV1 by Rugina A. [1] was valid. Also, since the main difference is the syntax between EMV1 and EMV2, so it is also considered that the logic of implementa-
tion will remain the same. Therefore transformation rules for AADL Error Model Annex to PNML Model will remain the same as those were used by Rugina A. [1]. Hence, the transformation rules in our research are based on the transformation rules of Rugina’s research and we assume that the EMV1 used for AADL model by Rugina was correct considering the fact that her work is published and it must be validated. Considering this assumption, this research will focus on the transformation rules rather than the EMV1 or EMV2 syntax. The transformation rules are described using the EMV1 syntax mapped with PNML model. It is important to mention that this EMV1 syntax is only used to describe the transformation mechanism.

4.3.1 Transformation Rules

Transformation rules presented in this section are based on AADL Error Model transformation to GSPN model set by Rugina A. [1]. As described in Chapter 4.3, GSPN differs from PNML in terms of immediate and delayed propagations. Also, we described in Chapter 3 the assumption made for error propagation that all the errors are assumed to propagate immediately. This research does not consider the delayed errors because the system in study is avionic system and it is considered that due to safety sensitivity nature of avionic systems, the propagation of errors must be considered as immediate. It is considered that any error, whether it is triggered immediately or with a delay, needed to be considered as immediate while developing PNML model. Also, whether the error propagates immediately or with a delay, it has the same impact upon other components in terms of dependability.

The objective of studying all the dependencies (structural, functional, fault-tolerance and recovery) is to analyze the inter-dependability of connected software and hardware components in case of occurrence of an error in one of the components. Whether this error propagates immediately or with a delay, it is considered that it has same effect on the dependable components. If it does not have same impact, it is assumed in this research that it will have same impact. Hence, only immediate propagations are considered in this research.

Figure 4.3 shows the basic transformation rule of AADL error model to PNML model transformation of an independent component. The name independent is just a name to demonstrate the transformation rule. Part ‘a’ of figure shows the AADL Error Model Annex syntax of EMV1 and part ‘b’ of figure shows the correspondent PNML Model.

Features of error model of component named as independent are declared in EMV1. The first feature is initial error state that is AADL syntax for declaring the initial error state and is named by user as Error_Free. The corresponding PNML model shown in part ‘b’ of the figure shows the petri net initial place with token and same name as that of AADL model; that is Error_Free. Similarly, in AADL Error Model implementation of component in part ‘a’ of figure shows the
Figure 4.3: AADL Error Model to PNML model component transformation

AADL syntax for place to place transition. ‘Error_Free-[Perm_Fault]->Failed’ means that the component changes state from Error_Free state to Failed state via Perm_Fault transition. The corresponding PNML Model shows the transition in part ‘b’ of 4.3.

The error models the components of the case study systems are modeled using this basic rule of transformation. Section 4.4 describes the dependencies among the components modeled in PNML using the same rule of transformation as shown in Figure 4.3.

Figure 4.4 shows the PNML model developed in Wolfgang tool. As described previously in Chapter 2 and 3 that PNML is XML based language and the graphical model is also simultaneously available in XML format. Wolfgang tool provides the .pnml extension to save the file. This file can be read by any commonly available XML reader such as Notepad in Microsoft Windows operating system. The XML format describes the petri nets in XML language with all the features.

It is important to notice that the XML code generated for PNML model developed in Figure 4.5 is the description of PNML unique feature ‘arc’. Arc represented in the code describes a ‘source’ and ‘target’. Rest of the notation is XML graphical notation. Since PNML also covers all kinds of petri nets, including colored petri nets, so this notation supports all kinds of petri nets in XML. The .pnml file is reusable, readable, editable and easily understandable in XML format. It is platform independent and any operating system that has a notepad application is capable of opening this file format. Moreover, since PNML
is in language form so it can easily be edited using any XML reader software tool including the notepad.

It is important to mention that all the dependencies (structural, functional, fault-tolerant and recovery) are modeled using above mentioned transformation rule. AADL Error Model code of EMV1 for all the dependencies mentioned in configuration1 (Figure 4.1) and configuration2 (Figure 4.2) is already provided by Rugina A. [1] in detail. Until it is very necessary, the AADL Error Model code for dependability modeling will not be repeated as the basic principal of transformation has already been explained in 4.3.

In coming sections, the PNML models for architectural dependencies will be drawn one by one with simulation results and dependability analysis.
4.4 PNML Dependability Model

PNML dependability model aims to describe and simulate the architectural dependencies among the components of the system. Previously these dependencies were modeled using GSPN by Rugina A. and Feiler P. [1] [3]. This research focuses modeling these dependencies using PNML. Focus of this research is four architectural dependencies for dependability analysis. These four dependencies are:

- Structural Dependencies
- Functional Dependencies
- Fault-Tolerance Dependencies
- Recovery Dependencies

Figure 4.6 shows the block diagram of architectural dependencies in reference to the software and hardware components and sub-components. This diagram is based on Configuration1 shown in Figure 4.1. The details of the individual dependencies are previously described in Section 4.2. Configuration2 diagram is similar with a small variation of links of Processor1 and Processor2 to the dependencies S3, R4 and R3, S4. In that case Processor1 is linked with S4 and R4 while Processor2 is linked with R3 and S3.

These dependencies are modeled using transformations rules of AADL error model to PNML model as described in Figure 4.3. It is important to mention that AADL error model code of EMV1 is already available in the research done by [1]. This section will focus on the simulation of architectural dependencies using PNML and the details of corresponding EMV1 code will not be considered.
On the basis of rules defined in Figure 4.3, first we define the PNML models for independent processor, independent bus and independent sub-component. Following behaviors of the Processors are considered. Both the processor; processor1 and processor2 are assumed to have similar behaviors.

1. Initial state of component is Error_Free.
2. Two types of faults are activated; temporary faults (Temp_Fault) and permanent faults (Perm_Fault).
3. Temporary faults after activation move to TempErr state from where those are recovered by using Recover transition and the component is back to initial Error_Free state.
4. Permanent fault moves to PermErr state with two possibilities of Detected or NonDetected transition.
5. If the NonDetected transition is triggered, the component moves to error state named as ErrND and then transited to Failed state via Eh transition. From Failed state, the fault is repaired via Repair transition and the component is back to initial Error_Free state.
6. If the Detected transition is triggered, the component moves to Failed state. From Failed state, the fault is repaired via Repair transition and the component is back to initial Error_Free state.
7. All the transitions are assumed to be immediate.

Figure 4.7 shows the corresponding PNML modeling of independent processor.

![Figure 4.7: PNML Modeling an Independent Processor](image)

Following behaviors of the Bus are considered.

1. The initial state of the component is Error_Free state.
2. If the fault occurs, the component moves to Failed state via Fail transition.

3. From Failed state, the component moves to Error_Free state again via Repair transition.

4. All the transitions are assumed to be immediate.

Figure 4.8 shows the corresponding PNML modeling of independent bus.

![PNML Modeling an Independent Bus](image)

Following behaviors of the sub-components (processes) are considered. All the sub-component; FP_Comp1, FP_Comp2, RD_Comp1 and RD_Comp2 are assumed to have same behavior.

1. The initial state of the component is Error_Free state.

2. If fault is triggered, the component moves to Activation state via Fault transition.

3. Once the fault is activated, it’s either detected or not detected. If the fault is not detected, the component moves to an error state named as ErrND via NonDetected transition.

   (a) In ErrND state, either the fault is recovered via Recover transition to move the component back to initial Error_Free state, or the component moves to Failed state via PerceiveFail transition.

   (b) From Failed state, the component is restarted to Error_Free state via Restart transition.

4. If the fault is detected, the component moves to an error state named as ErrD.

   (a) In ErrD state, either the component moves to Failed state via Fail transition, or the component moves to Error_Free state by eliminating the fault via Eliminate transition.

5. All the transitions are assumed to be immediate.
Chapter 4. Execution

Figure 4.9: PNML Modeling and Independent sub-component (process)

Figure 4.9 shows the corresponding PNML modeling of independent sub-components (process).

It is important to mention that above mentioned models are generalized models for processor, bus and component (software component). In latter sections it will be clear that the model expressed in Figure 4.9 is used for both RD and FP components.

Next sections of this chapter describe the implementation of PNML model in terms of structural, functional, fault-tolerance and recovery dependencies. Before that, section 4.5 describes the validity mechanism for PNML model.

4.5 PNML Model Validity Mechanism

PNML Model for structural, functional, fault-tolerance and recovery dependencies will be developed in coming sections. Before we start developing the models, there is a need to define the validity mechanism for our PNML model. PNML model developed in this research needed to be validated in terms of all the structural, functional, fault-tolerance and recovery dependencies. We describe the mechanism in terms of structural dependencies and same will be applied to other dependencies as well.

Structural dependencies in terms of a processor to a sub-component (in terms of AADL, a sub-component is called as a process or thread) are described by Rugina A. [1] as the binding of the sub-component to the processor. The sub-component is bound to processor in terms of how the processor behaves. Also, it is illustrated that in structural dependency, the sub-component is dependent upon the behavior of the processor. This means that if an error is triggered in a pro-
cessor it will affect the sub-component bound to it. This structural dependency was simulated by Rugina using GSPN on a tool named as SURF-2. Although the simulation is not publically available, yet the GSPN models provided and theoretical explanation of structural dependencies illustrated by Rugina A. in her research referred as [1], indicate that GSPN model shows the structural dependencies in sub-components bound to processor. These dependencies are indicated in terms of error triggered from the processor that changes the state of the structurally dependent sub-components. One process and two sub-components model is used by Rugina A. and we also use the same model. GSPN model of Rugina A. shows that the error triggered in the processor changes the state of both the bounded sub-components from an error free state to error activation state. Also, it is shown that the error triggered in sub-components does not trigger any errors in processor and the state of the processor remains as error free.

We shall simulate same one processor and two sub-components combination with similar dependencies but using PNML. If the behavior of the sub-components remains the same in case of triggering of error in linked processor as that of GSPN, our research will be validated. Our PNML model must simulate the triggering of error from processor and the error must propagate in both bounded sub-components. Also, the state of the sub-components must change from Error Free state to some error state. Moreover, if we trigger the error in any of the sub-components, the propagation of error must not cause the change in Error Free state of neither the processor nor the other components. When all the above mentioned requirements are met, it means that our PNML model for structural dependencies produce similar results of GSPN model of structural dependencies developed by Rugina A [1]. This will not only validate the PNML model but also will be the first PNML model for structural dependencies; an ISO standard to conduct dependability evaluation for future research. Same mechanism will be used for functional, fault-tolerance and recovery dependencies.

4.6 PNML model for Structural Dependencies

On the basis of all the knowledge of section 4.1, 4.2, 4.3 and 4.4, the PNML Model for structural dependencies is developed and simulated in Wolfgang tool. The model is designed for two sub-components (FP_Comp1 and RD_Comp2) and one processor (processor1). S1 and S3 dependencies, as described in section 4.2, are considered in this model. The behavior of the model for S2 and S4 bounded to processor2 in terms of error propagation will remain the same as that for S1 and S3 bounded to processor1. Hence only model for S1 and S3 is developed. Figure 4.10 shows the Wolfgang output of PNML model for structural dependencies.

Figure [4.10] is the un-edited form of PNML model obtained for structural dependencies in Wolfgang tool. To elaborate what is inside happening in Figure
Figure 4.10: PNML model for Structural Dependencies S1 and S3

??, an edited form of Figure ?? is shown in Figure 4.11. In Figure 4.11 the FP_Comp1 is distinguished by enclosing it within green colored boundary. Similarly, RD_Comp2 is enclosed in red color boundary and processor1 is enclosed in blue color boundary. Also, dependency S1 is enclosed in blue color boundary and dependency S3 is enclosed in red color boundary.

All three components, processor1, FP_Comp1 and RD_Comp2 are on the initial state with one token. The number tokens will always remain one in any particular component because the component is assumed to be only at one state at a time. In both Figure 4.10 and Figure 4.11, the next transitions after the initial error state (Error_Free state) are red colored in all components. This means that when simulation is started, the next possible transitions of all the components are ready to be triggered. We can simulate one transition at a time. For example, in component FP_Comp1 in Figure 4.12 if we click on the transition labeled as Fault, the token will move from Error_Free state to Activator state. Also, when the token reaches to Activator state, NonDetected and Detected transitions will glow to red color, which indicates the next possible path to token transition. Figure 4.12 shows this transition.

The components distinguished in Figure 4.12 are modeled based on same principle as described in section 4.4. Temporary error triggered from processor1 propagates to FP_Comp1 and RD_Comp2 via TempErr state. The out propagation of temporary error from processor to FP_Comp1 and RD_Comp2 is
Figure 4.11: Elaboration of Figure 4.10

triggered by PrsrErrOut state. Since both FP_Comp1 and RD_Comp2 and dependent upon processor1, so propagation of error in any of these components from linked processor, in this case processor1, changes the state of the component from Error_Free to Activation state. This indicates that the any error triggered in linked processor will propagate in the corresponding linked components and the Error_Free state of the component will be changed to Activation state; that is an error activation state. This is true for both the temporary error and non-detected permanent error triggered from processor. Simulation also shows that any error triggered within the FP_Comp1 and RD_Comp2 does not propagate to processor. Which indicates that FP_Comp1 and RD_Comp2 and structurally dependent upon processor and processor is not structurally dependent upon any of these components.

This PNML simulation has similar output as that of GSPN model described by Rugina A. [1] for structural dependencies. Hence this PNML model is validated for AADL to PNML transformation for structural dependencies with simulation results similar to AADL to GSPN transformation conducted by Rugina A. [1].
Figure 4.12: Simulation Example

4.7 PNML Model for Functional Dependencies

There are two functional dependencies that are modeled here in PNML; F1 and F2. As demonstrated in Figure 4.1 and Figure 4.2, F1 describes the propagation of error from FP_Comp1 to both the RD components; RD_Comp1 and RD_Comp2. Similarly F2 describes the propagation of error from FP_Comp2 to both RD components; RD_Comp1 and RD_Comp2. This indicates that Rd component is functionally dependent upon the FP component. Rugina A. [1] modeled the GSPN model which indicates the functional dependencies F1 and F2 triggers error in FP_Comp1 and FP_Comp2, one at a time, which propagates through both the RD components; RD_Comp1 and RD_Comp2. GSPN model also indicates that any error triggered in any of the RD components does not propagate backs to FP component. Our PNML model is intended to fulfill the same above mentioned conditions as those are of GSPN model.

Figure 4.13 shows the PNML model for functional dependencies and the simulation of the model produces the desired results. Both RD_Comp1 and RD_Comp2 are functionally dependent and the PNML model simulates the propagation of error from FP_Comp1 to both the RD_Comp1 and RD_Comp2.
Also, when error is triggered in any of the RD components, it does not propagate back to the FP_Comp1. This indicates that this PNML simulation for functional dependency F1 is identical to GSPN simulation for F1 as done by Rugina A. [1]. For F2, the simulation remains the same as only the component that triggers the error is switched from FP_Comp1 to FP_Comp2. No matter which one of these components trigger the error, the propagation and the behavior of error will remain the same.

![Figure 4.13: PNML Model for Functional Dependency F1](image)

It is important to mention that PNML model presented above for functional dependencies is valid in both the configurations of system Configuration 1 and Configuration 2 (Figure 4.1 and Figure 4.2).

### 4.8 PNML Model for Fault-Tolerance Dependencies

The concept of fault-tolerance dependency is described by Rugina A. [1] as it influences the mode of the system rather than its state. The mode of the system is referred to the roles of the components that are being considered for fault-tolerance dependency. These roles of the components are considered when a number of replicas of the same component are used to perform a task. For example, if a task is performed by three replicas, R1, R2 and R3, of a component, then it is considered that component is performing in R1R2R3 combination or mode.
If one of the replicas fails, for example R1 fails; it is considered that component is performing in R2R3 mode. One of the replicas always acts as primary replica. This means that if primary replica fails the mode of the system changes and another replica acts as primary replica until the original replica is repaired. This indicates that fault-tolerance dependency considers the mode of the replicas. In case of two replicas of a component, if one replica fails, the other replica takes charge until original replica is repaired. It is important to mention that fault-tolerance dependency only focuses the switch of roles of replicas in case of failure of primary. It does not focus the repair mechanism of the faulty replica. This indicates that in two replicas scenario, if error is triggered in primary replica, secondary replica becomes the primary replica. If error is triggered in secondary replica, it does not affect the primary replica. Also, for the shifting of roles between the two replicas it is necessary that if primary replica fails, the secondary replica must be in the error free state. Rugina A. [1] implements this above mentioned scenario using GSPN model. We implement and simulate this by using PNML.

Figure 4.14: PNML model for Fault-Tolerance Dependency

If the error triggered in primary replica of our PNML model simulation makes the secondary replica as primary, and, the error triggered in secondary replica does not switches the primary replica to secondary, fault-tolerance dependency will be validated for our model. Both considered configurations of the system (Figure 4.1 and Figure 4.1) show that the fault-tolerance dependency exists between the two sub-components (FP_Comp1 and FP_Comp2) of FP component named as FT1. We assume that FP_Comp1 acts as primary replica and FP_Comp2 as secondary
replica. Figure 4.14 shows the PNML model of fault tolerance dependency. The simulation of the model shows that when we triggered error in FP_Comp1, it propagates until it switches the mode of the FP_Comp2 as primary. Token in place Prim1 indicates that FP_Comp1 is the primary component. When error is triggered in FP_Comp1 in simulation, the token is shifted to place Prim2 which indicates that FP_Comp2 is now primary.

Same mechanism will be used for RD components RD_Comp1 and RD_Comp2. This model is applicable for both the configurations (Figure 4.1 and Figure 4.2) as fault-tolerance between the component replicas is independent of other components connected as the switch mechanism is assumed to switch all the responsibilities from one replica to the other.

### 4.9 PNML Model for Recovery Dependency

Recovery dependencies are described by Rugina A. [1] as R1, R2, R3 and R4 as shows in Figure 4.1 and Figure 4.2. The concept of recovery dependencies is described in one processor and two sub-components scenario by Rugina A. [1]. The scenario of processor1 and sub-components FP_Comp1 and FP_Comp2 are used with R1 and R3 recovery dependencies. Recovery dependencies described in this scenario as if any of the sub-component or both fail, they cannot be restarted until the restart event is triggered by the processor. This restart event can only be triggered, if the processor is in error-free state. Until the processor trigger the restart event, both sub-components (if both are in failed state) and one sub-component (if one of them is in failed state) will remain in the failed state and cannot recover them to error-free state. Also, if the processor fails, both the components will go to fail state. Moreover, if the processor recovers from failed state, the sub-components will remain in the failed state until the processor triggers the restart event. The GSPN model of Rugina A. [1] is modeled on the basis of above described scenario. We build the PNML model to meet the above scenario and simulate to validate whether or not it meets the scenario.

Figure 4.15 shows the PNML model for recovery dependencies for R1 and R3 dependencies (Figure 4.1) in Configuration 1. Recovery dependencies model for R2 and R4 in Configuration 1 will be the same. Only the sub-components are needed to be switched. Same is the case with Configuration 2 where one-processor and two-sub-component model can be developed for processor1 and FP_Comp1 and RD_Comp1 for R1 and R3 recovery dependencies. Also, the same model works for processor2 and FP_Comp2 and RD_Comp2 for R2 and R4 recovery dependencies to meet the scenario described above. The simulation of PNML model shown in Figure 4.15 meets the scenario described above, hence is a valid PNML model for recovery dependencies R1 and R3 in Configuration1.

AADL model and corresponding PNML model is implemented and the PNML model is also validated for structural, functional, fault-tolerance and recover de-
Figure 4.15: PNML Model for Recovery Dependencies, R1 and R3 Config1

dependencies against the GSPN models of previously done research of Rugina A [1]. Next section describes the maintainability analysis of PNML dependability model.

4.10 Maintainability Analysis of PNML Dependability Model

As described in Chapter 1, maintainability analysis is carried out on four characteristics or factors [19] [20]. These factors include analyzability, changeability, stability and safety. As described in chapter 3, only the ‘Changeability’ factor is focused in this research due to time constraints. Analyzability, stability and testability are recommended for future work.

‘Changeability’ is defined as the ability of the system to adapt changes [20]. The focus of study here is the ‘changeability’ of PNML dependability model. It means that the concept of ‘changeability’ in this case will be the ability of PNML dependability model to adapt changes. As described in previous sections of this chapter, PNML dependability model is built on the basis of structural, functional, fault-tolerance and recovery dependencies. This means that, for example, ‘changeability’ in recovery dependency will be the ability to adapt changes if the scenario of the PNML model for recovery dependability is changed. For example, in section 4.9, we used one-processor and two-sub-components scenario for recovery dependability. If we change the scenario as one-processor and three-sub-
components then ‘changeability’ for ‘recovery dependability’ will be the ability of ‘PNML recovery dependency model’ to adapt this change. This ‘adaptation’ will be validated if the third sub-component added in this scenario behaves in the same way as the other two already existing components. By ‘behavior’ we mean that, for example, the sub-components FP_Comp1 and RD_Comp2 fail if the processor moves to Failed state. Also, FP_Comp1 and RD_Comp2 do not restart until the restart event is triggered by the processor. Also, if FP_Comp1 and/or RD_Comp2 fail, the impact of this failure does not change the state of processor.

On the basis of above discussion, we derive test-cases for testing the hypothesis that the PNML recovery dependency model contains the changeability factor of maintainability analysis. We shall test our hypothesis on the basis of following test-cases.

1. **Test-Case 1:** The hypothesis is true if and only if new component added fail if the processor fails.

2. **Test-Case 2:** The hypothesis is true if and only if the new component does not restart until the restart event is triggered by processor.

3. **Test-Case 3:** The hypothesis is true if and only if the new component, if fails, this failure does not change the state of the processor.

If newly added component passes these test-cases, this will validate that ‘PNML recovery dependency model’ possesses ‘changeability’ factor as it is able to adapt change in the model. We test the hypothesis by inserting a new component for an assumed recovery dependency ‘Ra’ and make it dependent to the processor by extending the PNML recovery dependency model described in section 4.9 and shown in Figure 4.16. The resulted PNML recovery dependency model with newly added component is shown in Figure 4.16.

This model is tested against all the above mentioned test cases. The test is carried out by simulating the model for each case. All the simulations verified all the test cases to be true. This validates the hypothesis that the PNML recovery dependency model possesses the changeability factor of maintainability analysis. On the basis of this result we can conclude that in case of any dependency (structural, functional, fault-tolerance and recovery) the respective PNML model possesses the changeability factor. This is concluded because existing dependencies and their impact among any components in the system remains the same. For example, in case of structural dependencies, a component structurally dependent upon the other component is because of a certain connectivity mechanism. AADL model explains the connectivity aspects in terms of in and out propagation of events and data. The connectivity mechanism remains constant for types of components; for example, a processor is a hardware component and processes the
software components. So the software components are always dependent upon
the processor. If the processor fails, the software component also fails. Hence,
the software component is always structurally, functionally and in terms of recov-
eries dependent upon the processor. In sections 4.6, 4.7 and 4.9 we studied these
dependencies in one-processor and two-sub-components scenario. So any number
of sub-components connected to processor will fail if the processor fails. In case
of multi processors for multiple software component scenario also, the proces-
sor cannot fail if software component or components dependent upon processor
fail. But the software component will fail if the corresponding processor (that
processes the software component) fails. Therefore, the ‘changeability’ factor of
maintainability analysis exists in case of structural and functional dependencies
as well. For fault-tolerance dependency, the basic condition for this dependency
to exist is the presence of identical replicas that are set to switch roles if one
of them fails. In this case also ‘changeability’ factor always exists because any
number of replicas can be used with any number of FT dependencies. The basic
condition must be met that all the replicas must be identical in role (the tasks
they perform).

This concludes this chapter. Next chapter describes the results obtained in
this chapter with respect to aims and objectives of this research.
Chapter 5

Results

This chapter discusses the outcomes of this entire research work. One of the outcomes is the difficulties and issues found in AADL and OSATE2 tool. GSPN not being the petri net standard is found to be much complex than PNML, although the both achieve the same objectives, yet it is found that PNML is better option for dependability evaluation of avionic systems. Also, PNML dependability model for avionic systems is one of the main outcomes of this research. This research also obtained a standardized mechanism to conduct maintainability analysis at architecture level of software development. In the process of conducting maintainability analysis, an arguable process of software testing at architecture level is also found. All these outcomes, which are also the answers to our research questions, obtained from this research are listed below;

1. PNML Dependability Model: an ISO standard
2. Maintainability Analysis of PNML Dependability Model.
3. AADL Error Model Annex and OSATE2 tool: Difficulties and Problems
4. Complexity of GSPN in comparison to PNML
5. Software Testing at architecture level

All the above listed outcomes are described one by one in the following sections.

5.1 PNML Dependability Model: an ISO standard

The PNML dependability model implemented in chapter 4 is the first knowledge to the scientific literature of its kind. PNML dependability model, implementing the structural dependencies and simulating those dependencies successfully was not done previously in any research. PNML being the ISO standard, this knowledge of PNML use, especially for dependability evaluation, is first time available as a result of our research and is very useful for researchers who aim to use PNML language for their related respective research projects.
Since the dependability model of an avionic system is discussed in our research, and as described in chapter 1 that avionic systems are some of the most highly safety-critical systems, so this outcome of our research, that focuses the dependability modeling of architectural dependencies in PNML, provides new knowledge to the avionic system researchers for use of PNML dependability model for analyzing architectural dependencies in their related researches. Also, the entire dependability evaluation process is now standardized as a result of our research. Previously only AADL was standard and GSPN was not. Now both AADL and PNML being industry standards, the dependability evaluation process is entirely standardized; hence answer to research question 3 is obtained.

5.2 Maintainability Analysis of PNML Dependability Model

PNML dependability model in our research is built for architectural dependencies at architectural level. We focused only ‘changeability’ factor of maintainability analysis is our research. The outcome of our research, as describes in chapter 4 section 4.10, shows that our PNML dependability model is maintainable in terms of any changes made to the model in terms of adding or removing new components of the system.

This knowledge of PNML dependability model maintainability in terms of changeability is first time available to the scientific literature as a result of our research; hence providing answer to research question 4.

5.3 AADL Error Model Annex and OSATE2 tool: Difficulties and Problems

During the development of AADL model, a number of difficulties and problems were faced. These problems faced are not discussed in scientific literature so far, with best of our knowledge. First, the literature available for AADL language is very limited. There is only one book [17] that describes the AADL language and some of the information that was required to conduct this research was missing in the book. The information missing was the syntactical and tool specific details about AADL EMV1 (Error Model Version 1) and EMV2 (Error Model Version 2). Although the syntax of EMV1 is understandably available by the research work done by Rugina A. [1], yet EMV1 is no more used as syntax for error model. The tool used for AADL modeling as recommended by AADL official website [31] is OSATE2 and it is the same tool that is used by Ruguna A. and Feiler P. [3]. Available downloadable versions of OSATE2 do not accept the EMV1 syntax and generates the syntactical error. On investigating the internet for reasons, it was
clear that the latest version of OSATE2 uses EMV2 and does not support EMV1 anymore. The problem with EMV2 is that very little literature that uses EMV2 in some research is available. Still the knowledge regarding how to use EMV2 and how EMV2 works in an AADL model is missing. The only book of AADL available [17] does not describe EMV2 and its use.

It is important to mention that the working of EMV1 syntax and EMV2 syntax available is understandable and the logic of available EMV1 code for error model of subsystem of French Air Traffic Control System used by Rugina A. [1] is understandable. Not only the logic is understood but it was also converted to EMV2 code to run in OSATE2 by us for this research. But the use of EMV2 within the AADL code of the system always generated unknown errors. Also, the unavailability of the literature about the use of EMV2 made us unable to fix the errors generated. We also contacted relevant AADL expert for guidance but never got a response.

Since the logic was understood and the AADL error model transformation to PNML model mechanism was understood, so the research was carried out and the errors were bypassed. Moreover, even if the OSATE2 would have executed the error model for EMV2 and the code would have worked somehow, the output of AADL error model cannot be visualized in the graphical output of the system. Furthermore, this research and the research of Rugina A. [1] were dependent upon a petri net modeling language to simulate the dependability evaluation of the system. AADL is incapable of simulating the dependability model by itself. So, the logic of EMV1 and EMV2 is important for transforming the AADL model to its corresponding petri net Model (GSPN in case of Rugina A. and PNML in our case) for simulation. That is also one of the reasons that this research ignores the syntax of EMV1 and EMV2 execution problems and focused on logic. The logic is described in Chapter 4, section 4.3 and Figure 4.3. That logic was used to model the PNML model and the outcome of our PNML dependability model is validated to match with Rugina’s GSPN model w.r.t architectural dependencies, as described in Chapter 4.

This makes us state that one of the findings of this research is that even if the EMV1 and EMV2 are not implemented within the system’s AADL model but the logic is understood, the dependability evaluation can be conducted, provided that the AADL error model is properly mapped with petri net (GSPN or PNML) model related to particular respective component or subcomponent of the system; hence answering research question 1.

5.4 Complexity of GSPN in comparison to PNML for Avionic Systems

The dependability models developed in GSPN by Rugina A. [1] are complex to understand and simulate due to the existence of ‘prohibition arc’ [39] along with
the conventional ‘arrow arc’ to connect places and transitions. The purpose of prohibition arc is to destroy the ‘state’ of the system and to prohibit the movement of the component from one state to the other [39]. The use of this arc is an extra graphical element that makes the graphical model difficult to understand. PNML, on the other hand, uses only arrow arc to graphically connect places and transitions and instead of using prohibit arc between two elements, PNML does not use any arc at all to prevent the unwanted transition. Because arrow arc already serves the functionality of providing dual direction to the transitions between two elements.

This existing difference between GSPN and PNML was not previously specified in scientific literature. This is discovered and stated here in this research for the first time. GSPN used immediate and timed transitions. This means that a component may change its state either immediately or with a delay. PMNL considers only the change of state, not the time involved. According to the survey paper available on the official website of PNML, immediate and timed transitions are not added in so far released parts of standard (Part 1 and Part 2) [38][40]. It also states that the immediate and timed transitions are aimed to be added in Part 3 of the standard. Part 3 of the standard is not released yet.

We consider that the timed and immediate transitions of GSPN may be effective in some other cases where the system may be required to be analyzed to delayed or immediate change of state, but in case of our system, the only transition important is the change of the state of a component from an error-free state to an error state. It is crucial to realize that this particular change of state (from error-free to error) state of a component of a system is different than any other states GSPN is designed for. For example, let us assume that another system that is designed to observe the motion of one of its components. If the component changes its state from moving state to standing state, in this case GSPN immediate and timed transitions may help to determine whether the components stops immediately or after some delay. But in case of error-free state to error state, it is crucial to consider that by the end of transition system fails (a component or sub-component and eventually the system). If a component is moving from moving state to standing state it may be designed to move and stand. But in our case system is not designed to fail, especially avionic systems with higher safety criticality. So this particular change of state needed to be treated differently. For example, in two component scenario, one being processor and other being software component, if processor fails, its impact will cause the software component to be failed as well. Now if the processor fails immediately or with a delay, the software component will only be affected when the state of processor is changed to fail state. This implies that even if there is a delay before failure, the system will be considered as error-free as the impact of the failure is not visible yet. Hence, GSPN design property of immediate and delayed transition are useless in the particular case of error-free state to an error state scenario. The use of this property in this case will only increase the complexity of the model and
the outcome of the model is not affected in any case. The outcomes of all the architectural dependencies in our PNML dependency model described is chapter 4 that are identical to GSPN model described by Rugina A. validate this argument. This was the reason that we made the assumption that all the transitions will be considered immediate in our research.

This research adds this new knowledge to scientific literature that immediate and delayed transitions property of GSPN, if used, in case of any system where state of the component changes from error-free to error increases the complexity of the model without affecting the outcomes; hence answering to research question 2.

5.5 Software Testing at Architecture Level using Standards

Software testing is defined by Kim M. in reference to IEEE standard 829-1998 as, “the process of investigating the system or system components to verify that whether or not it satisfies the requirements” and Kim M, also defines software testing in reference to Myers as “The process of execution of a software system for the purpose of finding error in that”. Both above mentioned definitions are intended for those systems that have already been developed.

Taking both definitions under consideration, our PNML model is capable of testing the system w.r.t architectural dependability requirements of components, yet it still needs further experimentation and research to validate the argument. Also, it is arguably capable of testing the impact of failure of one component on the other in case of inter-dependent/inter-connected components; and the impact of new components added to the system. PNML dependability model obtained in this research is capable of finding architectural dependencies among the system components on architectural level in a standardized way for the first time (PNML being ISO standard). Using this knowledge, most of the dependability features of component-based software can be tested on architecture level of software development process. These features, as obtained by our research are, structural dependency among the components, functional dependency among the components, fault-tolerance dependency between the identical components and recovery dependency among the components in case of system failure. Testing architectural dependencies on architectural level can help fixing the system design at architecture level w.r.t connectivity and data flow among the components.

This outcome was not included or intended in aims and objectives of the research. Yet by the end of the completion of research, this outcome also is a helpful knowledge for those researchers who are intended to research on architecture level system testing in future.
Chapter 6

Analysis and Discussion

In this chapter, an analysis with respect to validity threats is done. Also, we discussed the general aspects of this research, especially difficulties faced to guide future researchers.

6.1 Validity Threats

Validity of a research is concerned with the trustworthiness and truthfulness of the study [43]. Validity is to find out the extent to which the research is trustworthy and true and not biased by the researchers’ subjective school of thought and viewpoint [43]. There are four aspects of validity [43], and following sections will describe the validity threats to our research in terms of these four aspects.

6.1.1 Construct Validity

This aspect of validity reflects the reality of actual operations representation in comparison to the imaginations of researcher about those [43]. The threats to construct validity in our case are listed below.

1. The first threat to construct validity was the architecture of the system available. The source code of the system was not available so first threat was to know that system reflected in Figure 4.1 and Figure 4.2 is actually an outcome of a source code or it is just a design. Because all the next steps of research depend upon the information available in these two figures.

   (a) To avoid this threat, AADL model was reconstructed using OSATE2 tool and AADL language. The code was written in tool and run. The AADL outcome of the code was same as reflected in Figure 4.1 and Figure 4.2.

   The details of source code and OSATE2 tool outcomes are available in Appendix A.

1. Second threat to construct validity was to verify that whether the source code of AADL Error Model Annex, described by Rugina [1], also provides the same output in OSATE2 tool or not. When source code of Rugina was
implemented, it appeared that OSATE2 tool does not accept this code and generates the syntax error. Further investigation revealed that this code was written by using older version of AADL Error Model Annex named as Error Model Version 1 (EMV1). More investigation revealed that EMV1 is out of date now and is not being used by AADL community anymore because they are using a new version Error Model named as EMV2.

(a) To overcome this threat, EMV2 was tried out, but very few literatures of EMV2 was available, such as [35]. When EMV2 did not work and no further literature was available to learn EMV2, we contacted AADL research community via email. They never replied.

(b) To overcome this issue, we mapped the ‘logic’ (reflected from EMV1 syntax used by Rugina) of error model with PNML and done the transformation. The details of transformation are described in section 3.6 and 3.7.

6.1.2 Internal Validity

The threat to internal validity are a concern when casual relations are examined [43]. Since our research is a mixed method approach, and there are no casual relations involved in our research, therefore this aspect of validity is not reflected in our research. Hence, in case of our particular research, the aspect of internal validity is not reflected.

6.1.3 External Validity

This aspect of validity is related to the extent to which, the generalization of findings can be done [43]. The threat to external validity to our research was to whether or not this research remains to only case-study system or in future other avionic systems can be designed using this process of dependability evaluation.

To avoid this threat, we conducted maintainability analysis with respect to ‘changeability’ (Section 4.10 of Chapter 4 describes the details of maintainability analysis). As a result, our PNML dependability model is tested for multi-component and processor scenario and in this way this threat to external validity was avoided.

6.1.4 Reliability

The measure of extent to which the data and analysis are dependent upon the individual or individuals, who are conducting the research, is the reliability aspect of validity [43]. Considering the ‘data’ in our research, since most of it is extracted from previous research and that is publically available as well, so anyone can use the data and carry out the research of same kind as we did.
‘Analysis’ of this research cannot be dependent upon the individual researcher or researchers, because all the steps of system architect and design, as well as, the system’s dependability models (both GSPN and PNML) are based on the ‘System’s Requirements’. The behavior of the component depends upon its ‘requirements specifications’ that is assumed to be done previously (prior to ‘design phase’ of development by the requirements engineers who specified requirements of French Air Traffic Control System). In our case, FP_Comp1 and FP_Comp2 are replicas of each other. We, neither Rugina \[1\] decided this to make replicas. They were assumingly ‘requirements engineers’ of the system who made these replicas of the same component. Therefore, steps of dependability evaluation presented in our research are not dependent upon the individuals’ thought but these are a sequence of steps that any researcher can follow to obtain the same result.

6.2 Discussion

Before we conclude the thesis, it is important to mention the difficulties faced during this research to assist future researchers. This research involves two of the most recent modeling languages AADL and PNML. These languages were not the part of academic courses studied. Also, very little development related guideline material was available in literature. One of the issues to carry out this research was to first understand the languages themselves and their working. As described earlier, some of the tools for GSPN dependability evaluation mentioned in previous research (Rugina A. \[1\]) were outdated (SURF-2). Although AADL tool OSATE2 was available, yet very little literature or guideline for its usage was available. Tool was understood by experimenting with the GUI until first prototype AADL model was ready. PNML tools were even more difficult to find as PNML is rarely used in previous research for modeling. The official website of PNML has recommended a number of tools. Some of them are ePNK, PNML Framework, PIPE and Wolfgang. ePNK and PNML framework are Eclipsed (Oracle Framework) based and their extensions are imported to eclipse. After experimenting with ePNK and PNML Framework, we concluded that it is very hard to simulate our prototype PNML model on these tools until a proper guideline book is available, which unfortunately is not available in literature. PIPE tool have different versions and some of them do not work on Windows platform. Those which works, are disabled by-default with simulation functionality. Wolfgang tool, which was listed at the end of the tools list on PNML website, was tested at the end. This tool was not as difficult to use as others and was the first tool that ran our prototype simulation. At the end, OSATE2 tool was used for AADL modeling and Wolfgang tool was used for PNML dependability modeling and simulation.

Another difficulty involved in conducting this research was the unavailability
of guideline literature of AADL error model annex EVM2. EMV1 syntactically and logically was available enough (in research publication of Rugina A [43]) to carry on this research. But OSATE2 tool does not support EMV1 anymore. This reality was understood when after many experimentation with EMV1 code in OSATE2, we contacted AADL experts on GitHub for guidance. One of the AADL researcher responded with confirmation that OSATE2 does not support EMV1. Infact EMV1 is no longer in use and EMV2 is now used for error annex modeling in OSATE2. Little literature available about EMV2 revealed that EMV2 is syntactically different than EMV1. Although we tried to convert AADL annex model of our case-study system EMV1 to EMV2, yet OSATE2 tool did not accepted the EMV2 syntax. Due to time constraints, we used EMV1 logic for transformation to PNML, as described in Chapter 4. The outcomes of simulations conducted in Chapter 4 show that the PNML simulation was dependent upon the error-free execution of EMV2, but it was dependent upon the logic of transformation of AADL error annex model to PNML dependability model.
Chapter 7  
Conclusion and Future Work

By the end of this research, it is concluded that PNML dependability model is successful to design, implement and simulate the case-study system French Air Traffic Control System. It is also concluded that PNML being ISO standard is a standardized choice for dependability evaluation. It is also concluded that PNML dependability model is maintainable and can be used for other component based systems, especially avionic system.

This research carried out the dependability evaluation process consisting of AADL modeling of the case-study system (subsystem of French Air Traffic Control System), AADL to PNML transformation rules and PNML dependability model development, execution and simulation. This research was a continuation of previously conducted research by Rugina A. and Feiler P. [3]. They carried out dependability evaluation using AADL and GSPN. Our research, in addition, also carried out maintainability analysis of obtained PNML dependability model, considering the ‘changeability’ factor of maintainability analysis. As first step of this research, AADL model for the case-study is developed system using OSATE2 tool. Then, as second step, AADL error model annex EMV1 was transformed to respective PNML model. Third step was to develop, execute and simulate the PNML dependability model considering the architectural dependencies; structural, functional, fault-tolerance and recovery using Wolfgang tool. The outcome of PNML dependability model of these dependencies is compared with the outcome of GSPN dependability model previously obtained by Rugina A. [1]. The match of outcomes of PNML dependency model and GSPN dependability model validated the PNML dependability model for dependability evaluation. In fourth and last step, maintainability analysis of the obtained PNML dependability model was carried out considering the ‘changeability’ factor of maintainability analysis.

PNML is an ISO standard and comparatively less complex than GSPN for dependability modeling. Since dependability model involve the analysis of change of state of system’s components in terms of error-free state to fail state, GSPN’s ‘immediate’ and ‘timed’ transition properties used for dependability model increases the complexity of the system. The simulation result shows that PNML dependability model produced the same outcome as that of GSPN dependability modeling without using ‘immediate’ and ‘timed’ transitions. PNML standard
Chapter 7. Conclusion and Future Work

(Part 1 and Part 2) so far does not have capability of handling ‘immediate’ and ‘time’ transitions. Even if PNML had this capability, the dependability model specific to studying error-free and fail states, is not relevant to use these properties.

PNML dependability model obtained as a result of this research is the first research on dependability evaluation of an avionic system using an ISO petri net standard, which is PNML. Previously GSPN was used which is not a petri net standard. Maintainability analysis was done using ‘changeability’ factor for obtained PNML dependability model. This is also first maintainability analysis on PNML dependability model. It is recommended for future work to carry out maintainability analysis on PNML dependability model using other factors; analyzability, stability and testability.

On the basis of chapter 4 and chapter 5, this research can be carried on further in future in more than one directions. First, due to time factor, we were able to touch only the ‘changeability’ factor of maintainability analysis. Other factors, analyzability, stability and testability can be studied in future for PNML dependability model. Secondly, in future, if the proper guideline literature is publically available, EMV2 of AADL error model annex can be used for the validation by reverse engineering the PNML dependability model. Also, the comparative analysis of EMV1 and EMV2 with an implemented case study can also be done in future, provided that if the guideline literature is available for implementation. Thirdly, some of the conventional software testing techniques currently used for post-development testing can be used in future for architectural-level testing using PNML dependability model.

PNML dependability model for subsystem of French Air Traffic Control System is now completely standardized after the completion of this research. Previously AADL was the standard and GSPN was not. Now by using PNML instead of GSPN, we are able to completely standardize the process. This standardized process for dependability evaluation can be used in future for any types of safety critical systems, especially avionic systems. Moreover, this process can be used for dependability evaluation of other component-based systems where the system requirements prioritize the change of state of components and its impact on other dependent components.


Appendix A

AADL Model of System

This appendix describes the implementation details of components of French Air Traffic Control System in AADL using OSATE2 tool. The system’s implementation is described component by component with detailed figures.

Figure A.1 shows the symbolic ports that represent different types of connection. These ports include ‘event data’, ‘event’ and ‘data’ ports and are illustrated in Figure A.1. The component by component implementation of the system is described in following sections.

A.1 AADL Model for Flight Plan Component

As described in Section 4.1, the Flight Plan component has two sub-components as FP_Comp1 and FP_Comp2. One of these components acts as primary component at a particular time with FP_Comp1 as default primary component. Figure A.2 is the representation of Flight Plan component. Figure A.3 shows the AADL implementation of the FP component. As visible in Figure 4.1, this component is connected to the RD component by ‘event data ports’ that are capable
to transfer both control and data information. Figure A.3 also shows the output and input ports named as ‘sysinput’ and ‘sysoutput’. These ports are created to communicate with RD component. ‘sysinput’ port receives control and data information from RD component and ‘sysoutput’ sends control and data information to RD component. Inside Flight Plan, both the sub-components FP_Comp1 and FP_Comp2 are capable to send and receive this information independently. This shows the integrity of each of these sub-components.

Moreover, since both sub-components are identical in functionality and one act as primary and the other as secondary, so they also are connected to each other using ‘data’ ports. These data ports are responsible to shift the responsibility of being primary sub-component in case of failure. The event port named as ‘IAmPrim’ indicates which of the sub-components is acting as primary. Figure A.3 shows the AADL syntax of the Flight Plan component.

It is important to mention that dependencies are not visible in AADL model. They are implemented using Error Model Annex syntax. AADL is incapable of simulate the dependencies and output them to visualize the error propagation in graphical form. That is why PNML is used to view and simulate the dependencies.

### A.2 AADL Model for Radar Data Component

Flight Radar component is also programmed in AADL on the basis of the information in Figure 4.1 and Figure 4.2. There are two sub-components of Radar Data component, RD_Comp1 and RD_Comp2, as shown in Figure A.3 which is a part of Figure 4.1 and focuses only Radar Data component. As previously
described the output and input ports, named as ‘sysinput’ and ‘sysoutput’, are created to communicate with RD component. ‘sysinput’ port receives control and data information from RD component and ‘sysoutput’ sends control and data information to RD component. Unlike FP components, RD_Comp1 and RD_Comp2 do not communicate with each other [1].

In Figure A.5 and Figure A.5, S3 and S4 are structural dependencies with the assumption that any hardware faults can propagate to the software components that are running on top of that particular hardware, in this case RD_Comp1 and RD_Comp2 are software components and processor1 and processor2 are hardware components. In Figure A.5 the RD_Comp1 is bound to processor2 and RD_Comp2 is bound to processor1, whereas in Figure A.5 RD_Comp1 is bound to processor1 and RD_Comp2 is bound to processor2. Similarly, R3 and R4 and recovery dependencies with assumption that if the sub-component fails it cannot be restarted provided that the corresponding processor is also in a failed state. Again, it is important to mention that the dependencies S3, S4, R3 and
R4 cannot be visible in Figure A.7. This is because it is the AADL output of the syntactical implementation of the RD component. This is due to the fact that AADL only shows the structural and communication (data and event connectors) details of the system. Although these dependencies can be implemented using AADL Error Model Annex, still these are not visible in output diagram of the AADL model. This is one of the reasons to use PNML to view these dependencies and their impact.
Appendix A. AADL Model of System

sub-components RD_Comp1 and RD_Comp2 visible in Figure A.5 and Figure A.6 as this_RD_Comp1 and this_RD_Comp2. There is nothing different than names. Names are set different only because of the syntactical requirements and readability reasons of AADL language. Figure A.8 shows the AADL code of RD component. The code shows the need of using different names for the sub-components.

![Figure A.8: Radar Data Component AADL Syntax](image)

Although Figure A.5 and Figure A.6 show two different bindings of RD_Comp1 and RD_Comp2 to processor1 and processor2, still it is important to notice that in both configurations (Figure 4.1 and Figure 4.2) there are two sub-components bound to one processor (E.g. FP_Comp1 and RD_Comp2 and bounded to processor1 in Configuration1) with respect to structural dependencies always. Same is the case with recovery dependencies. This implies that while modeling the structural and recovery dependencies in PNML, any two sub-components and one processor combination is sufficient to analyze the structural and recovery dependencies. Also, as the AADL model does not show dependencies in its output, therefore the implementation of both the configurations in AADL is not required separately.

A.3 AADL Model for Processors and Bus

The two processors and one bus are connected through bus access connector. Processors and bus are hardware components. Figure A.9 is extracted from Figure 4.1 which shows the processors and bus connected to each other through bus access connectors.

There are no architectural dependencies shown in the Figure A.9 because these are physical components and are physically connected. A physical damage to the bus is assumed to be the system failure in both configurations.
Appendix A. AADL Model of System

Figure A.9: Systems Processors and Bus

Figure A.10: AADL Model for Processors and Bus

AADL model of the processors and bus is shown in Figure A.10 and the AADL code is shown in Figure A.11.

```
processor Processor1
    features
        bus_access: requires bus access Bus1;
    end Processor1;

processor implementation Processor1.impl
    end Processor1.impl;

processor Processor2
    features
        bus_access: requires bus access Bus1;
    end Processor2;

processor implementation Processor2.impl
    end Processor2.impl;

bus bus1
    end bus1;

bus implementation bus1.impl
    end bus1.impl;
```

Figure A.11: AADL code for Processors and Bus

Although there are no direct architecture dependencies visible among the processors and the bus yet the connections between the FP and RD components are bound to the bus. Failure to bus means that FP and RD will fail functionally as the functional dependencies F1 and F2 among FP and RD are compromised because of the failure of the bus. The overall system in corresponding to Figure 4.1 and Figure 4.2 is described in next sub-section that illustrates the overall AADL model.
A.4 AADL Model of Entire System

As the component by component implementation of the system with the AADL syntax is described in previous sub-sections, this section describes the system as a whole in reference to Figure 4.1 and Figure 4.2. It is important to mention that all the components, sub-components and the system itself is developed in OSATE2 tool using AADL language. Figure 4.1 and Figure 4.2 are not the exact outcome of the system as when the system is actually developed in AADL. The readers must consider that some of the information available in Figure 4.1 and Figure 4.2 is not the features of AADL. Dotted lines and blue lines are not included in the features of AADL. These are just for description. The reader must consider that these lines are assumed to be used for explanation purposes by the original author, who is Rugina A. [1].

However the textual information available in the work of Rugina A. [1] is used to develop the model of the system. Figure A.12 shows the AADL model of subsystem of French Air Traffic Control System.

Figure A.12: AADL Model of Subsystem of French Air Traffic Control System

Figure A.12 is the output of the AADL model of the system saved directly from the OSATE2 tool in JPEG format. The visible system in Figure A.12 is different than that of Figure 4.1 and Figure 4.2 with respect to what is visible only. Rest of everything is implemented the same and have been described previously component by component. The inside sub-components of FP and RD are not visible in Figure A.12. Those have already been described in previous sub-sections. It is also important to mention that physical availability of Nominal and Reconfig modes in the Figure A.12 architecture does not provide any means to switch between these two modes for analysis. Theoretical details of these modes are
Appendix A. AADL Model of System

available in [1]. In this research, any impact of these modes on the dependability analysis is assumed to be minimal hence ignored in PNML model.