International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.6 (2016), pp.153-170
http://dx.doi.org/10.14257/ijmue.2016.11.6.14

Research on AADL Model for Qualitative Safety Analysis of Embedded Systems

Yinling Liu, Guohua Shen, Fei Wang, Jia Si and Zi Wang

College of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
{ylliu2013, ghshen, wangfei, jiasi and wangzi}@nuaa.edu.cn

Abstract

As embedded software is taking an important part in safety critical filed, how to ensure the safety of safety-critical software has recently become a research focus. Developing safety-critical and highly reliable systems almost always includes significant emphasis on safety analysis and risk assessment. There have been substantial improvements in automation and formalization of other aspects of critical system engineering. However, safety analysis and risk assessment are still largely manual and informal activities and tool support is limited. In this paper, we propose a qualitative safety analysis method of embedded system based on AADL (Architecture Analysis & Design Language) model. Firstly, we extend the Error Model Annex with the R-FMSE (Risk-based Failure Mode Safety Effect) property to express the failure mode formally. Then, we give a detail illustration for qualitative safety analysis based on AADL model. Thirdly, we give a algorithm to develop a R-FMSE analysis Eclipse plug-in to realize the automation of the method. On using the Isolette system (an infant incubator), a case study is demonstrated the feasibility of this method.

Keywords*: Embedded system, AADL, Safety analysis, Error model, R-FMSE*

1. Introduction

Safety critical embedded systems are extensively employed in fields like avionics, spacecraft, energy, defense and transportation, which have high requirements for resource, response time, fault tolerant and special hardware, especially for safety. So, safety assurance of embedded system has become one of research hotspots. More attention should be paid to finding critical failures. Critical failures can lead to hazard. Hazard can cause death, injury, damage to or loss of equipment or property, or damage to the environment. To find critical failures at early design stage, MDE (Model-Based Engineering) is proposed.

With the development of MDA, SAE (Society of Automotive Engineers) released the aerospace standard AS5506, named AADL, which is a mature industry-standard for embedded systems and has proved to be efficient for architecture modeling [1]. AADL supports nonfunctional attributes analysis by adding EMA (Error Model Annex) at the early development stage. EMA allows the user to annotate system and software architectures expressed in AADL to be annotated with hazard, fault propagation, failure modes and effects due to failures, as well as compositional fault behavior specifications to facilitate incremental and scalable automated safety analysis [2]. Thus, AADL model plays an important role in safety analysis at model level.

Risk-based failure modes and safety effects analysis(R-FMSEA) and Failure modes and effects criticality analysis (FMECA) are both safety analysis methods. FMECA is a design discipline where an engineer examines and records the consequences of any (usually only single point) failure on the operation of a system. The purpose of FMECA is to highlight any significant problems with a design and, if possible, to change the design

*:corresponding author

ISSN: 1975-0080 IJMUE

to avoid those problems. In contrast, R-FMSEA is a detail design discipline that examines and records the safety consequences of a system through safety criticality analysis. The difference between R-FMSEA and FMECA is in the construct of the inductive R-FMSEA spreadsheet that, in addition to the standard columns of an FMECA, includes risk-based safety-related aspects such as failure rates, risk, safety criticality and inspection methods [3]. So in the case of safety analysis, R-FMSEA is better than FMECA.

Safety analysis method can be divided two groups: inductive and deductive methods [4]. Inductive analysis methods like FMEA, PHA (Preliminary Hazard Analysis) or HazOp are used to determine causal relationships between failure of individual components and failure at system level. These traditional methods are very mature, but laborious, costly, time-consuming and error-prone for the poor integration between safety analysis and design techniques. Deductive analysis methods like RBD, FTA or FTA extension. These methods can compute the failure probability of the occurrence of the system from the probability of the subcomponents. As for AADL model, both methods can bring available information to it. However, inductive analysis is more important and essential part for introducing faults into the system model. Furthermore, in [5], the FMECA has been adopted as inductive safety analysis based on AADL model. Therefore, in the case of safety analysis, through the comparison between FMECA and R-FMSEA above, this paper focuses on AADL-based safety analysis using R-FMSEA method.

R-FMSEA for AADL model holds the most advantages of FMECA for AADL model. In addition, R-FMSEA still offers the following benefits:

(a)This method can provide more accurate data to assess the degree of the component's safety. (Risk)

(b)It offers a way of assessing the consequences of component failure such as death or personal injury, and environmental or financial losses about, according to a relative scale of safety. (Safety Criticality)

(c)We have developed an Eclipse plug-in to generate failure modes propagation paths automatically.

After conducting R-FMSEA based on AADL model for qualitative safety analysis, engineers can give special attention to these components at the design phase and iterate or refine the architectural model. Additionally, this analysis process is automatic and has improved the method of traditional R-FMSEA process to be more accuracy, more rapid and less error-prone.

Outline Section 2 briefly introduces AADL via the case study. Section 3 extends EMA with the R-FMSEA property to construct AADL safety model. Section 4 remarks qualitative safety method and give an algorithm to realize the R-FMSEA Eclipse plug-in. Section 5 exemplifies it with the case study to illustrate the efficiency of this method, and conclusion is drawn in Section 6.

2. Safety Analysis Framework Based on AADL Model

2.1. AADL Model and Error Model Annex

AADL is the society of automotive engineer's standard dedicated to modeling embedded real-time system architectures [6]. Through components and connections, AADL describes the structure of hardware and software of system. AADL component type can be defined as one of the three component categories: software application components (process, thread, thread group, subprogram, and data), execution platform components ((virtual) processor, memory, device, and (virtual) bus), and composite component (system). The software application components are allocated execution platform components.

Figure 1 illustrates the AADL graphical component architecture notation for the top-level system architecture for the Isolette. From the Figure, we can see that Isolette is

consisted of four subcomponents which are temperature_sensor, thermostat, operator_interface and heat_source. For these subcomponents, thermostat and operator_interface are composite components, temperature_sensor and heat_source are execution platform components.

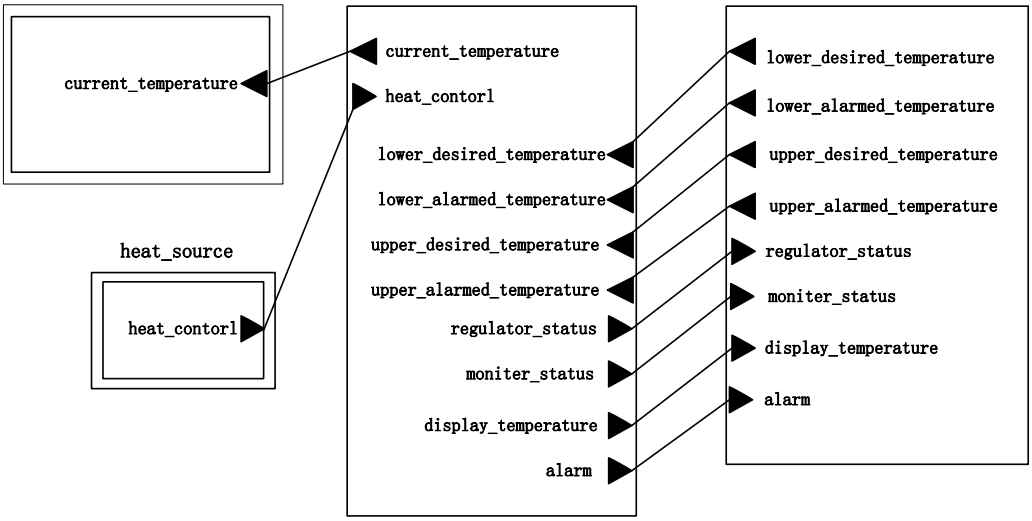


Figure 1. AADL Graphical Component Architecture of Isolette

In order to connect these components, AADL provides connections. In the Isolette system, port connections and flows are used to connect these four subcomponents. Triangles represent component ports, and lines represent data flow connections between ports.

The Error Model Annex is a sublanguage annex which extends the AADL core language and is included in AADL standard. An error model is a state machine that can be associated with an AADL component or connection in order to describe its behavior in terms of logical error states in the presence of faults [7].

```
annex EMV2{**
error types
 HeatControlError: type;  --heater on or off inappropriately
 AlarmError : type;   --the class of alarm errors
 FalseAlarm : type extends AlarmError;--alarm erroneously
                                      sounded
 MissedAlarm : type extends AlarmError;  --alarm missed
end types;
error behavior FailStop
events
 fail: error event;
 completed: error event;
states
 working: initial state;
 failed : state;
 done : state;
 transitions
    working -[fail]-> failed;
    working -[completed]-> done;
 end behavior;
**};
```

Figure 2. A Simple Example of Error Model Annex

An error model definition is divided into an error model type and an error model implementation. Elements declared in the error model type can be customized through component-specific properties, when an error model is associated with a component as an error model instance. Several error model implementations can correspond to the same error model type. Figure 2 shows an simple Error Model Annex which includes error types declaration and error behavior declaration.

The error model type declares error types (*i.e*., HeatControlError, AlarmError, FalseAlarm and MissedAlarm) error states (*i.e*., working and failed), error events (*i.e*., fail and completed). One error state (working) is the initial state. The error model instance is initially in the state working. If a fail error event comes, it becomes failed. Then, if comes error event completed, it becomes done.

2.2. Safety Analysis Framework

R-FMSEA method is a mature industrial safety analysis method, which contributes to safety effect of failure modes. AADL can be used to construct architecture model of embedded systems, and the EMA can model error properties. But EMV cannot analysis error types.by itself. So, we define R-FMSEA property to extend EMA to analysis the safety effect of failure modes. For realizing the safety analysis, we propose a safety analysis framework based on AADL model and extended EMA.

Figure 3 gives the framework, we can see that AADL and extended EMA can model the system architecture and error types from the requirements. The safety model which is AADL model combining extended EMA can be used to make R-FMSE analysis.

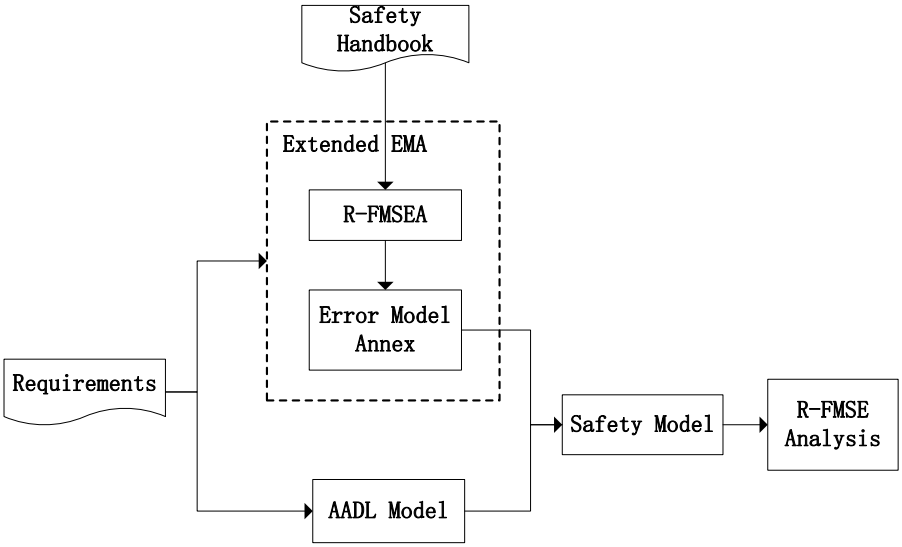


Figure 3. Safety Analysis Framework Based on AADL Model and Extended EMA

3. AADL Safety Model

3.1. Overview of Risk-Based FMSEA

Risk-based Failure Modes and Safety Effects Analysis is a design discipline that examines and records the safety consequences of a system through safety criticality analysis [3].

In safety critical field, the determination of safety criticality is essentially an expansion

of risk analysis in which focus is placed upon the importance of safety critical component early in the engineering design stage.

Any significant effect on the operational performance of critical component as a result of changes in designing for safety will inevitably have an influence on the performance of the total process. In effect, risk-based safety criticality analysis quantifies these impacts on the total process performance, whereby preventive maintenance tasks are scheduled according to required frequencies.

Safety criticality in process engineering is complex, and basically depends upon the reliability of component subject to a variety of failure risks. The interaction between the various risks of failure leads to this complexity. These risks are defined as the result of multiplying the consequence of failure by the probability of its occurrence.

Thus, when R-FMSEA method is applied into AADL's EMA, we can provide not only formal modeling for errors but also tool supported for automation of failure effects safety analysis for embedded systems.

3.2. Error Model Annex Extension

The EMA can be used to annotate the AADL model of an embedded system to support a number of the methods. An architecture specification containing error models may be subjected to a variety of analysis methods. For example, FMEA can be generated from specifications to assess failure effect, or Markov analyses can be applied to assess reliability and availability. Similarly, EMA can be extended with R-FMSEA property to assess failure safety effect. EMV2(Error Model Version 2.0) enables modeling of different types of errors, error behavior of individual system components, modeling of error propagation affecting related components in terms of peer to peer interactions and deployment relationships between software components and their execution platform, as well as modeling of aggregation of error behavior and propagation in terms of the component hierarchy [8].

In order to analyses the safety effect of failure modes, we define "R-FMSEA" property in EMV2 property set. The "R-FMSEA" property includes function description of component, failure cause, severity of failure, likelihood of failure occurrence and failure rate of component, which is shown in Figure 4. Function description and failure cause are defined as *aadlstring*. And failure rate and likelihood are defined as *aadlreal*. At last, the type of severity is defined as *aadlinteger*. A short explanation for some properties is shown below.

```
RFMSEA:record
(
Function: aadlstring;
Cause: aadlstring;
Severity: aadlinteger;
Likelihood: aadlreal;
FailureRate: aadlreal;
)applies to (all);
```

Figure 4. Definition of R-FMSEA Property

Severity

The use of qualitative assessment scales for determining the severity of a failure consequence is common in risk analysis, where severity criteria are designated a value ranging from 10 to 1. The most severe consequence is valued at 10 (disabling injury—life risk), whereas no safety risk is valued at 1, or 0, as indicated in Table 1 Our severity value is determined by this industry standard in R-FMSEA.

Likelihood

Many different scales have been developed for determining the likelihood of failure occurrence. One commonly used scale is expressed in terms of 'probability qualifiers' given as: Actual occurrence = 0.95 to 1.00, Probable occurrence = 0.50 to 0.95, and Possible occurrence = less than 0.50 .

Failure Rate

The overall failure rate of the component in its operational mode and environment. Where appropriate, application and environmental factors may be applied to adjust for the difference between the conditions associated with the generic failure rate data and operating stresses under which the item is to be used.

If the failure rate for the item cannot be determined from available data, a representative estimation for failure rate is shown in Table 2

Table 1. Severity Classification Standard in Industry

Type	Severity	Value
Disabling injury	Life risk	10
	Loss risk	9
	Health risk	8
Reported accident	People risk	7
	Process risk	6
	Product risk	5
Physical condition	Damage risk	4
	Defects risk	3
	Loss risk	2
No safety	No safety risk	1

Table 2. Failure Rate Classification Standard in Industry

Qualification	Failure rate($\times 10^{-4}$)
Very low	<100
Low	100 to 500
Medium	500 to 1,000
High	1,000 to 5000
Very high	>5,000

Risk can be quantified as the product of the probability of occurrence (chance), with the level of severity of the risk (disaster or loss). Risk is an indication of the degree of safety. Thus, risk can be defined as the result of multiplying the consequence of the failure mode (*i.e*. its severity) by the probability of failure (*i.e*. its likelihood):

$$\text{Risk}(R) = \text{Severity} \times \text{Probability (or Likelihood)} \quad (1)$$

Once an overall total and an overall average value of risk has been assessed according to the risk assessment scale, a criticality rating can be defined for each failure mode, using the following expression:

$$\text{Criticality}(C) = \text{Risk} \times \text{Failure rate} \quad (2)$$

3.3. Safety Model

AADL safety model is consists of AADL model and Error Model Annex. In AADL safety model, AADL model is used to construct system architecture and EMV is used to construct error model. Every component should have an initial state and a state transition occurs when an event is fired. The data failure mode is treated as a state and does an event

failure mode as an event. Different error types inside component are recognized as failure modes. Then these errors propagate to other component through the port. Additionally, in the safety model, transitions, propagations, port connections, execution platform bindings, error propagations and error flows are used to compose the failure mode effect propagation graph and we can find the failure modes effect paths through this graph.

In error model, components failure modes can be added to describe behaviors that may invade safety requirement of the whole system. The error model of one component can be regarded as a stochastic automaton, in which data failure mode and event failure mode is regarded as state and event respectively.

In this paper, the safety model is an AADL model with extended EMV2 which contributes to qualitative safety analysis. The simple example of safety model is shown in Figure 5.

```
device heat_source_hs
features
  heat_control : in data port
Iso_Variables::on_off
   {BLESS::Assertion => "<<HEAT_CONTROL(x)>>";};
  regulator_mode: out data port
Iso_Variables::regulator_mode;
annex EMV2
  {**
  use types Isolette;
  error propagations
  heat_control: in propagation {HeatControlError};
  regulator_mode: out propagation{
RegulatorModeError};
  flows
    hc:error sink heat_control{HeatControlError};-
-detects heat control
    rmhc:error path
regulator_mode{RegulatorModeError}
          -> heat_control{ HeatControlError};
  end propagations;
    properties
       EMV2::RFMSEA => ([
            Function=>"manage_heat_source_mhs is
        used to control temperature and manage
        heat source.";
                    Cause=>"bad heat control!";
            Severity=>9.5;
            FailureRate=> 0.03;
            Likelihood=> 0.03;
    ])applies to hc.HeatControlError;
  **};
end heat_source_hs;
```

Figure 5. A Simple Example of Safety Model Based on AADL

4. Qualitative Safety Analysis

4.1. Failure Modes

For R-FMSEA method, firstly we need to determine failure modes. For Error Model Annex, it can define error events and error states. An error model is a state machine that

can be associated with an AADL component or connection in order to describe its behavior in terms of logical error states in the presence of faults.

To apply R-FMSEA to AADL model, event failure mode and data failure mode are analyzed for each component. Therefore, when event and state are as failure modes, the R-FMESA property of error model should be specified for them. In R-FMSEA process, failure modes can be found automatically.

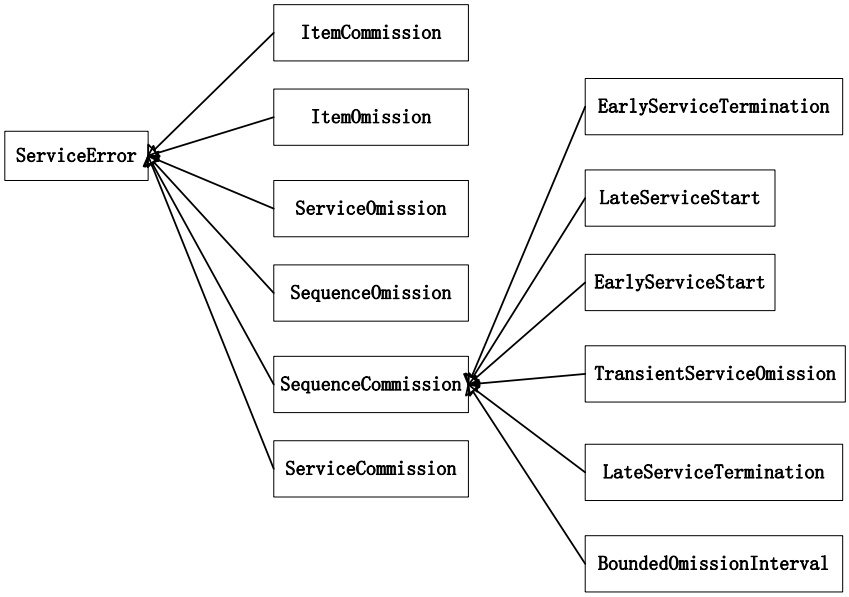


Figure 6. ServiceError Hierarchy

Failure mode is the basis of the R-FMSEA and the definition of failure modes is one of the most important parts in the process of R-FMSEA. The EMV2 has given some general error types, it includes service errors, timing errors, value errors, replication errors, concurrency errors. In order to reuse the error type knowledge, as well as be convenient to check the inconsistencies, we build ontology for these error types. For example, service errors ontology can be seen in Figure 6.

4.2. Failure Causes

After determining the failure modes for R-FMSEA, the most probable causes for each failure mode should be identified and described. Failure causes can lead to failure modes. Failure modes come from inherent design defects in the AADL model. Thus, when the causes of postulated failure modes are found, the design process should be examined and improved. SAE-J1739 [9] provides a list of appropriate failure causes for failure modes. IEC 60812 [10] suggests that failure causes may be identified by eliciting the opinion of experts, while the design is new and without precedent. We use these two standards, failure causes are defined in the R-FMSEA property of the error model for each component. When R-FMSEA is carried out, failures causes can be obtained automatically.

4.3. Risk-Based Safety Critical Analysis

In safety critical area, the determination of safety criticality is essentially an expansion of risk analysis in which focus is placed upon the importance of safety critical component early in the engineering design stage.

Safety criticality in process engineering is complex, and basically depends upon the reliability of component subject to a variety of failure risks. This complexity is due to the

interaction between the various risks of failure. These risks are defined as the result of multiplying the consequence of failure by the probability of its occurrence. In effect, risk-based safety criticality analysis quantifies these impacts on the total process performance. So, in the process of R-FMSEA, risk analysis is basic and necessary.

The consequence of a component failure mode is a failure effect. Severity assesses the significance of the component failure mode's final effect on component operation. In the safety analysis, risk is an indication of the degree of safety. It can be quantified as the product of the probability of occurrence (chance), with the level of severity of the risk (disaster or loss) and its expression can be seen at Section 3. From the definition, the measure of severity can be quantified in two events: accidents and incidents. The measure of probability can be quantified in the form of appropriate statistical probability distributions or measures of statistical likelihood. In this regard, an accident is an undesired event that results in disastrous physical harm to a person. An incident is an undesired event that could result in a loss. In the context of safety, this loss is in the form of an asset loss, which implies damage to equipment or property. Therefore, risk is an indication of the degree of safety, determined on the basis of two considerations, the first according to design criteria, and the second according to operational performance.

In the AADL safety model, components are organized in a hierarchy, a component failure mode is represented in the form of an error type, the error type can be propagated to other component through failure effect propagation path, and the failure effect propagation stops in the error sink at last.

In component error behavior of the Error Model subclause, component failure modes will be propagated through the port or binding. Firstly, the source of failure mode is located in error source of flow specification the combination of this error type and corresponding port or binding is regarded as first effect. Then, the error type is propagated to the next adjacent component through port connection or execution platform binding. The port or binding of next adjacent component combines with error types specified for port and binding as second effect. And then, the error type may propagate through failure effect propagation path continually until it reaches error sink of flow specification. In the same way, the propagation path from error source to error sink combines with error types specified for ports and bindings as final effect. The combination of component interactions and propagations composes error propagation paths. Every component can be regarded as a node. Therefore, these paths and nodes compose the error propagation graph. The composite component that includes subcomponents may have error propagation sub graphs. First effect, second effect and third effect (if has), *etc.* are only dependent on this error propagation path in the safety model. Failure effects can be found automatically in the safety model by searching the graph, so that there is no need to manually set specific failure effect propagation path for each failure mode ahead of time. Before R-FMSEA is carried out for AADL model, the risk rank of final effect is defined in the R-FMSEA property for the error type of error sink. Then, failure effect path and R-FMSEA property can be found automatically through failure effect propagation path in AADL safety model. So, combining R-FMSEA property, we can get the value of risk and criticality. Through the value of risk and failure propagation paths, we can find the most critical component. Then, engineer can pay special attention to this component and refine the architecture model to control component hazards in an acceptance level.

4.4. The Realization of R-FMSEA Eclipse Plug-in

In the AADL safety model, components are organized in a hierarchy, a component failure mode is represented in the form of an error type, the error type can be propagated to other component through failure effect propagation path, and the failure effect propagation stops in the error sink at last.

In order to apply this method into industry, it is necessary for us to realize it in Eclipse environment. Our plug-in development is based on Eclipse IDE (Integrated Development

Environment, IDE). The AADL meta model is implemented using the EMF (Eclipse Modeling Framework, EMF). The model is split across seven EMF Ecore packages: core, component, connection, feature, flow, instance, and property [11]. The development process can be as follows:

The first step: create a new plug-in project

The second step: extract failure mode information

To get the model's statistics, we must traverse the AADL specification or system instance and get information of the various model elements. We create a subclass of *AadlProcessingSwitch* for this purpose. This class provides model traversal methods and leverages the EMF switch class concept to allow the specification of processing methods for each class of object in the AADL model. Then, we use these statistics (these statistics are called key word in algorithm) to create graph.

The third step: create failure mode propagation graph

After extracting the failure modes information, we use the algorithm to create failure mode propagation graph. The algorithm can be seen at Figure 8.

The last step: get Failure Mode Propagation Paths (FMPP)

The source and end of mode propagation are labeled in graph, so that we can use traversal algorithm for finding all routes between two points to traverse the graph, then we can get FMPP.

From the development process, we can see that the algorithm of creating graph is the most critical step. The main idea of this algorithm is that: in AADL model, failure mode information is implicated in error propagations and flow specification. In this algorithm, each error propagation can be seen as node and every error path in flow specification can be seen as the transformation of failure modes. When the failure modes of two ports are the same, and the types of two ports are different, they can be matched (*i.e.*, one kind of failure mode can be propagated from out port to in port). When the failure modes of two ports are different and the types of two ports are different, they can be matched unless there is an error path between the two ports. For example, in Figure 4, there are two error propagations: heat_control and regulator_mode. The failure modes of these two ports are different, but there is an error path rmhc. It means the failure mode of RegulatorModeError in regulator_mode port can be transformed into failure mode of HeatControlError in heat_control port. Thus, we create two nodes of heat_control and regulator_mode by the two propagations and then create connection between these two nodes by the error path rmhc.

In order to make the algorithm clearly, we define the data structure for the nodes of the graph. In Figure 7, there are seven elements in this structure. One failure mode can be uniquely determined by ComponentName, Feature and FailureMode (*i.e.*, specific failure mode likes "HeatControlError"). The direction determines whether two components can be matched, which out is denoted as 1 and in is denoted as 0. And the location is labeled whether the failure mode is the source or destination of failure mode propagation path, which the source is denoted as 0 and the destination is denoted as 1. At last, the array next is used to create connections between one node and the next nodes which is at least one node.


```
class Node {
ComponentName String;
Feature String;
FailureMode String;
Direction int;
Location int;
Property R-FMSEA;//here, R-FMSEA is a kind of datatype which
          // can represent all the properties of the R-FMSEA method.
next[]  node *;//here, next[] is a array which is used to point to nodes
         // which are suited for this node.
}
```

Figure 7. The Data Structure of the Node

The input of the algorithm is the AADL source code and the out is Nodesource. The procedure of this algorithm can be divided into four steps. The first step is to initialize the variables, then create the vertex, thirdly, create lines to connect the vertexes. At this time, we have created a graph which implies failure modes information. Lastly, we use the mature traversal algorithm for finding all routes between two points to get FMPP. The time complexity of this algorithm is consisted of two parts: one is creating graph, another is getting FMPP based on graph. The time complexity of first part is O(n2), and the time complexity of the second part is O(n) [12].

```
Input: AADL project source code;
   Output: Nodesource .// Nodesource are graphs which are created based on //AADL
                          project source code.
Procedure:get_FailureModeGraph
//1.Initialize the variables
   ArrayList<Node> Nodes=new ArrayList<Node>(); //Nodes is used to save //the failure
                                        mode  propagation node
Nodes=null;
//2.create vertex
Search all components in source code
For each component:
//The first step: create nodes
if (key word== “error propagations”)   //in error propagation specification
      For  each  error  propagation      //ComponentName,  Feature,  FailureMode  //and
                                    Direction are determined
         {    Node node=new Node();
              node.ComponentName=componentName;
              node.Feature=feature;
              node.FailureMode=errorType;
              node.Direction=direction; }
Nodes.add(node); //let the node add to Nodes list.
if( key word ==“flows”)
         For each flow specification:                 //  created in the first step
          if( key word == “error source”)       // specification and the nodes
            While  (Features  and  FailureModes  are  matched)  //  Features  //and
                                    FailureModes are between flows
                 Node.location = “error source”;
if (key word == “error sink”)
```

```
          While ( Features and FailureModes are matched)
         Node.location = “error sink”;
if (key word == “error path”)
          While (Feature and FailureModes are matched)
            Node.next[i]=Node1;//Node is header and Node1 is the tail in this //“error path”
 // 3.create lines to connect the nodes
int count=0;//count is used to record the number of lines for one node.
    for(i=0;i< Nodesource.Size();i++):// Nodesource is a set for Node whose //location
                                        property equals “error source”;
{    count=0;
          for(j=0;j< Nodenon-source.Size();j++)// Nodenon-source is a set for //Node
                whose location property doesn’t equal “error source”
           if (Direction, Feature and FailureMode are all matched)
                 Nodesource[i].next[count++]= Nodenon-source[j];
      }
for(k=0;k< Nodenon-source.Size()&&k!=j;k++)
{   count=0;
          for(l=0;l< Nodenon-source.Size()&&l!=k;l++)
         if (Direction, Feature and FailureMode are all matched)
                   Nodenon-source[k].next[count++]= Nodenon-source[l] ;
      }
 Return  Nodesource ;
```

Figure 8. The Algorithm for Creating Failure Modes Graphs

5. Case Study

This section demonstrates the efficiency of the qualitative safety analysis method with the example of an infant incubator called “Isolette” [13]. In this paper, we use Isolette as the primary illustration, because it is relatively simple while still rich enough to illustrate a number of dimensions in safety analysis.

The Isolette mainly consists of four units: temperature_sensor, thermostat, operator_interface and heat_source. Among these units, thermostat and operator_interface are composite components, temperature_sensor and heat_source are execution platform components. The Isolette thermostat takes as input an air temperature value from a temperature sensor and controls a heat sources to produce an air temperature within a target range specified by the clinician through the operator interface. Safety concerns include ensuring that infant is not harmed by air temperature inside the isolette being too hot or too cool. The Isolette uses a subsystem separate from the operational thermostat to sound an alarm if hazardous temperatures are detected. Figure 9 presents a diagram of the Isolette’s primary system components and environment interactions.

Figure 10 gives the AADL description of part of the Isolette system. The automatic qualitative safety analysis has been implemented as an Eclipse plugin, and the R-FMSEA check list can be automatically generated from above safety model, as shown in Table 3 The heads of check list are “No”, “Component”, ”Cause”, ”Error Propagation Path”, ”Severity”, “Likelihood”, ”Failure Rate”, ”Risk”, ”Criticality”.

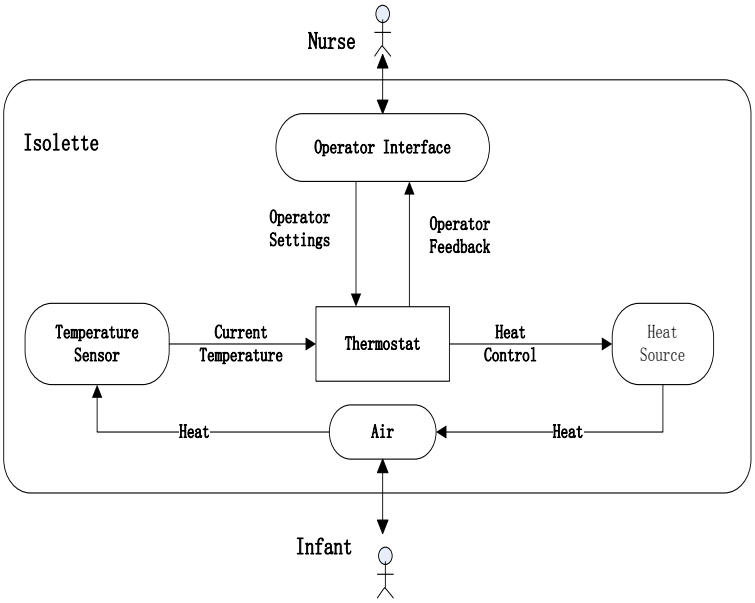


Figure 9. Operational Context for Isolette Thermostat

With a combination of severity, failure rate and likelihood, the Criticality may be set for action to mitigate effect of certain failure modes. In order to make this qualitative analysis more reasonable and efficient, we divide the degrees of severity into three levels by combining the standard severity category of the industry (see Table 3). From level 1 to 3, the severity is gradually decreased. Severity 8 to 10 is considered as first level, and then severity 5 to 7 is second level, finally, severity 1 to 4 is third level. The priority of failure mode in first level is definitely higher than that in second or third level. Similarly, the priority of failure mode in second level is definitely higher than that in third level.

From the part of the result, we can see that we just give three records, because the table is a litter bit large, so it is not convenient for us to show all the records. And we firmly believe that these three records are reasonable enough to demonstrate the advantage of R-FMSEA method. R-FMSEA is a kind of safety qualitative analysis which aims at helping engineers to get components failure information and instructing them to refine the system. For example, for the first, 7th and 15th R-FMSEA records of table, temperature_sensor is a device component which function is to detect temperature and manage_regulator_interface is a thread component which function is to manage the interface of thermostat's regulator.

Firstly, the severity values of these failure mode belong to first level, so we can discuss the three records at the same time; Then, when comparing first record and 15th record, we find that the severity values of failure modes are both 10, but the likelihood and failure rate are different, and by comparing the criticality of the failure modes, we suppose that engineer should pay attention to temperature_sensor' ServiceOmission failure mode first. While, when comparing the 7th and 10th records, we find that even though the severity value of the failure mode of OutOfRange in 7th record is less than the severity value of the failure mode of SubtleValueError in 15th record, temperature_sensor's OutOfRange failure mode is considered as more important failure mode by comparing the criticality of these two failure modes.

```
system implementation isolette.impl
subcomponents
  thermostat : system thermostat_th.impl;
  temperature_sensor : device temperature_sensor_ts.impl;
  heat_source : device heat_source_hs.impl;
  operator_interface : system operator_interface_oi.impl;
connections
  ct : port temperature_sensor.current_temperature ->
thermostat.current_temperature;
  hc : port thermostat.heat_control -> heat_source.heat_control;
  ldt : port operator_interface.lower_desired_temperature ->
thermostat.lower_desired_temperature;
  udt : port operator_interface.upper_desired_temperature ->
thermostat.upper_desired_temperature;
  lat : port operator_interface.lower_alarm_temperature ->
thermostat.lower_alarm_temperature;
  uat : port operator_interface.upper_alarm_temperature ->
thermostat.upper_alarm_temperature;
  rs : port thermostat.regulator_status ->
operator_interface.regulator_status;
  ms : port thermostat.monitor_status -> operator_interface.monitor_status;
  dt : port thermostat.display_temperature ->
operator_interface.display_temperature;
  al : port thermostat.alarm -> operator_interface.alarm;
end isolette.impl;
```

Figure 10. The Architecture Model of Isolette

The R-FMSEA method not only can instruct the engineers to refine the architecture model to eliminate or control hazards in acceptance levels, but also can give the error propagation paths of one failure mode of component. The error propagation paths can analyses the influence of one kind of failure mode in detail. For example, in 15th record of the result, the error propagation path can be seen as below:

```
temperature_sensor{current_temperature=SubtleValueError}-
->
thermostat.regulate_temperature.manage_regulator_mode
{current_temperature=SubtleValueError}-->
thermostat.regulate_temperature.manage_heat_source
{regulator_mode=RegulatorModeError}-->
heat_source{heat_control=HeatControlError}
```

Figure 11. The Description of Error Propagation in 15th Record of Table

The R-FMSEA method not only can instruct the engineers to refine the architecture model to eliminate or control hazards in acceptance levels, but also can give the error propagation paths of one failure mode of component. The error propagation paths can analyses the influence of one kind of failure mode in detail. For example, in 15th record of the result, the error propagation path can be seen as below:

Figure 11 tells us that the error source is component temperature_sensor and error sink (terminal point of the error propagation path) is the component heat_source. The HeatControlError of the feature heat_control in heat_source is originated from the SubtleValueError of feature current_temperature in temperature_sensor. The first level effect is that source failure mode results in SubtleValueError of current_temperature feature in manage_regulator_mode component which belong to thermostat. And then, it propagates to manage_heat_source, which result in RegulatorModeError of

regulator_mode feature. Finally, it leads to HeatControlError in heat_source. In a word, the error propagation path tells us that bad temperature causes bad mode and bad mode causes bad heat control.

Through these analyses, so engineers can know the influence of one kind of failure mode in detail and give special attention to one component to refine the architecture model.

6. Conclusion

In this paper, we introduce a qualitative safety analysis method based on AADL model. The proposed method can instruct engineers to iterate or refine the architectural model by giving a table of R-FMSEA. In order to generate the table automatically, we extend the Error Model Annex with R-FMSEA property, and then design the algorithm for finding FMPP. Finally, we develop an Eclipse plug-in to make it possible to generate the R-FMSEA table automatically. Comparing with the conventional safety analysis method, Informal manual our method mainly holds two advantages: formal specification of failure mode and automatically safety analysis by using R-FMSEA method. Even though our method helps engineers to refine the architecture design and solve the traditional problems of informal and manual in a way, there are many items in the table. So, in the future, we plan to find the critical path to filter many unuseful paths to instruct engineers more accurately.

Table 3. The Part Results of R-FMSEA, where (1) = Severity of the Consequence , (2)= Likelihood of Occurrence, (3)= Failure Rate,(4)=Risk (Probability×Severity), (5)=Criticality (Risk×Failure Rate)

#	Component	Cause	Error Propagation Path	(1)	(2)	(3)	(4)	(5)
1	manage_regulator_ interface	parameter invalid	temperature_sensor{ interface_failure= ServiceOmission}--> thermostat.regulate_temperature.manage_regulator_mode {interface_failure=ServiceOmission}--> thermostat.regulate_temperature.manage_heat_source{ regulator_modeRegulatorModeError}--> thermostat.regulate_temperature.manage_heat_source {heat_control=HeatControlError}--> heat_source{heat_control= HeatControlError}	10	0.45	0.05	4.5	0.225
7	temperatur-_sensor	detect the temperature of thermostat	temperature_sensor{current_temperature=OutOfRange}--> thermostat.regulate_temperature.manage_heat_source {current_temperature= OutOfRange }	8	0.25	0.05	2	0.1
15	temperatur-_sensor	detect the temperature of thermostat	temperature_sensor{current_temperature=SubtleValueError}--> thermostat.regulate_temperature.manage_regulator_mode{ current_temperature=SubtleValueError}--> thermostat.regulate_temperature.manage_heat_source{ regulator_mode=RegulatorModeError}--> heat_source{heat_control=HeatControlError}	10	0.25	0.01	2.5	0.025

Acknowledgements

This work was supported by the Fundamental Research Funds for the Central Universities of China (Grant No. NS2015093).

References

[1] "SAE Aerospace, SAE AS5506B: Architecture Analysis & Design Language (AADL) standard document", SAE International, **(2012)**.
[2] J. Delange and P. Feiler, "Architecture Fault Modeling with the AADL Error-Model Annex", Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on. IEEE, **(2014)**, pp. 361-368.
[3] R. F. Stapelberg, "Handbook of reliability, availability, maintainability and safety in engineering design", London: Springer,.K. Elissa, "Title of paper if known," unpublished, **(2009)**.
[4] Z. Q. Huang, B. Xu, S. Kan, J. Hu and Z. Chen, "Survey on Embedded Software Safety Analysis Methods for Airborne System", Journal of softeare, vol. 2, **(2014)**, pp. 200-218.
[5] B. Gu, Y. Dong, X. Wei, "A Qualitative Safety Analysis Method for AADL Model", Software Security and Reliability-Companion (SERE-C), 2014 IEEE Eighth International Conference on. IEEE, **(2014)**, pp. 213-217.

[6] P. H. Feiler, D. P. Gluch and J. J. Hudak, "The architecture analysis & design language (AADL): An introduction", Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, **(2006)**.
[7] P. Feiler and A. Rugina, "Dependability modeling with the architecture analysis & design language (AADL)", Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Institute, **(2007)**.
[8] P. Feiler, "Architecture Analysis and Design Language (AADL) Annex: Annex E: Error Model V2 Annex. Number SAE AS5506/3 (Draft) in SAE Aerospace Standard", SAE International, vol. 3, **(2013)**.
[9] SAE. SAE-J1739 JAN2009: (R) "Potential Failure Mode and Effects Analysis in Design (Design FMEA), Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA)", SAE Internaltional, **(2009)**.
[10] IEC 61025. IEC (Intern. Elect. Commission) "Fault-Tree-Analysis", (FTA), **(1990)**.
[11] Carnegie Mellon Software Engineering Institute, "Open Source AADL Tool Environment", http://www.aadl.info," Tech. Rep., **(2006)**.
[12] Z. Pei-de, "A Fast Algorithm for Finding the Shortest Path Between Arbitrary Two Points in a Traffic Road Net", Computer Engineering and Science, vol. 4, **(2002)** , pp. 35-37.
[13] D. Lempia and S. Miller, "DOT/FAA/AR-08/32", Requirements Engineering Management Handbook, **(2009)**.

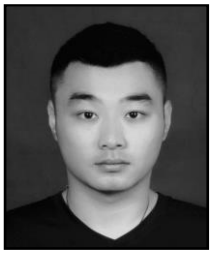
Authors

Yinling Liu, (1989), Male, Postgraduate, Major: Software engineering, Safety analysis, Model driven engineering.

Guohua Shen, (1976), Male, associate professor, Major: Software engineering, Safety analysis, Model driven engineering, software metrics, semantic web ,description logic and ontology.

Fei Wang, (1990), Male, Postgraduate, Major: Software engineering, Safety analysis, semantic web and ontology.

Jia Si, (1992), Male, Postgraduate, Major: Software engineering, Safety analysis, Model driven engineering.



Zi Wang, (1992), Male, Postgraduate, Major: Software engineering, Safety analysis, Model driven engineering.