基于 AADL 模型的静态故障树的自动生成

刘 玮,李蜀瑜
(陕西师范大学 计算机学院,陕西 西安 710062)

摘 要:在基于模型驱动的嵌入式软件开发中,为了避免安全问题可能产生的损失,需要对系统的安全性进行分析,找出可能发生错误的地方。在基于 AADL 系统模型的安全性分析上,利用 AADL 错误模型附件为系统故障和传播进行建模,通过追踪对象的可能故障源来提取系统的实例错误模型,然后将错误模型实例存储在有向图里,并根据有向图建立系统错误模型的故障树。通过故障树分析工具就可以高效地分析系统的安全问题。

关键词:结构化分析和设计语言;错误模型;有向图;故障树
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2013)10-0099-04
doi:10.3969/j.issn.1673-629X.2013.10.025

Automatic Generation of Static Fault Trees Based on AADL Model

LIU Wei, LI Shu-yu
(College of Computer, Shannxi Normal University, Xi'an 710062, China)

Abstract: In the embedded software development based on model-driven, in order to avoid losses the security problems caused, it is indispensable for system safety analysis, finding out the possibility of error. On the safety analysis of AADL system model, use AADL error model for modeling the system failure and propagation. The error model instance can be achieved by tracking the possible fault source of object, then store the error model instance into the directed graph, establish a system error model of fault tree on the basis of the directed graph. The fault tree analysis tool can be efficient to analyze system security problem.

Key words: AADL; error model; directed graph; fault tree

0 引 言

安全关键系统(例如航空电子、医疗器械、汽车领域等)[1]发生故障可能会造成难以估量的损失。系统安全分析是系统能够安全运行的重要保证[2]。传统的系统安全分析主要依靠工程师对设计文档和设计模型的研究判断,这种分析方法主要依赖于系统安全分析工程师的经验,难以准确地评估系统的安全性。为了解决这个问题,需要提供一种规范的系统安全分析方法,这种分析方法不仅能够分析系统各个构件的行为及整体性能,还必须能够考虑系统构件之间的交互接口。由于安全问题存在于整个系统的执行过程中,所以系统的硬件也是必须要考虑的问题。

结构化分析与设计语言(Architecture Analysis and Design Language, AADL)是一个 SAE 标准,它的核心语言能够为复杂实时系统建模[3]。AADL 还有两个重要的附件扩展:Behavior Annex 和 Error Model Annex。其中 Error Model Annex 能够用来描述故障和失效信息,它的主要特点是能够在 AADL 架构模型上描述系统的错误信息[4]。因此,通过分析系统构件的错误模型以及它们之间的交互就可以分析系统的安全性。

1 AADL 综述

1.1 AADL 模型综述

AADL 里最基本的元素是构件,构件声明定义了构件类型和构件实现。构件类型定义了构件的接口和外部可见的属性(例如,feature 描述构件的端口,properties 定义构件的内在特性)。构件实现定义了子程序、子程序执行顺序、属性等构件内部结构。AADL 将构件分为三类:

(1)应用软件构件(线程、进程、数据等);

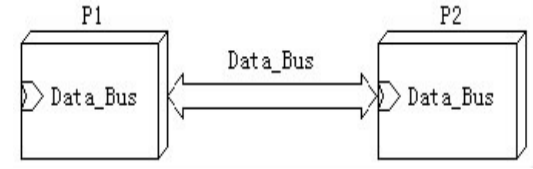
(2)执行平台构件(处理器、存储器、总线、外设等);

收稿日期:2012-12-18 **修回日期**:2013-03-25 **网络出版时间**:2013-07-24
基金项目:中央高校基本科研业务费专项资金(GK2010002011);教育部科学教育重点项目(107106)
作者简介:刘 玮(1984-),男,硕士研究生,研究方向为嵌入式软件;李蜀瑜,副教授,硕士生导师,研究方向为 Web 服务与组合、嵌入式系统。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20130724.0953.021.html

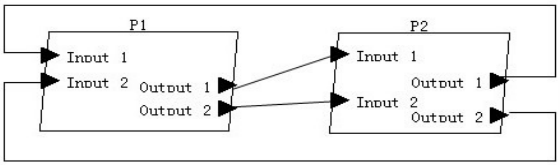
(3)复合构件(软件构件映射到执行平台的复合构件)。

AADL 用绑定属性来描述相关硬件构件、软件构件、连接之间的映射。AADL 系统实例模型是从模型定义中产生的,它以系统实现为系统实例的根,迭代实例化每个子构件。所有定义在构件里的绑定属性会结合于构件实例[5]。

图 1 所示的系统由两个处理器、两个进程以及连接它们的总线构成。这个系统(DualSystem)包括子构件 HW 和 SW,它们分别由 Duplex. Basic 和 Dual. Basic 实现,这两个子构件(HW 和 SW)也是复合构件。



HW



SW

图 1 DualSystem 构件

AADL 描述的部分 DualSystem 如下:

```
system Duplex
end      Duplex;
system implementation Duplex. Basic
  subcomponent
  P1:processor Proc. Basic;
  P2:processor Proc. Basic;
  Data_Bus:bus PCI. Basic;
connections
  bus access Data_Bus->P1. Data_Bus;
  bus access Data_Bus->P2. Data_Bus;
end Duplex. Basic;
system Dual
end      Dual;
system implementation Dual. Basic
   subcomponents
  P1:process SW_Proc. Basic;
  P2:process SW_Proc. Basic;
connections
  C12:data port P1. Output_1->P2. Input_1;
  C21:data port P2. Output_1->P1. Input_1;
  C12_2: data port P1. Output_2->P2. Input_2;
  C21_2: data port P2. Output_2->P1. Input_2;
end Dual. Basic;
system   DualSystem
end DualSystem;
system implementation DualSystem. Basic
  subcomponents
  HW:system Duplex. Basic;
  SW:system Dual. Basic;
  properties
      Actual_Processor_Binding=>reference HW. P1 applies to
SW. P1;
      Actual_Processor_Binding=>reference HW. P2 applies to
SW. P2;
      Actual_Connection_Binding=>reference HW. Data_Bus
applies to SW. C12;
      Actual_Connection_Binding=>reference HW. Data_Bus
applies to SW. C21;
      Actual_Connection_Binding=>reference HW. Data_Bus
applies to SW. C12_2;
      Actual_Connection_Binding=>reference HW. Data_Bus
applies to SW. C21_2;
  end DualSystem. Basic;
```

DualSystem 的实现也包括了两个进程到处理器的映射及连接两个处理器的通信属性,系统实例的创建需要实例化最顶层构件 DualSystem. Basic。

1.2 错误模型综述

AADL 是一个可扩展的,可对系统运行架构进行分析的建模语言。AADL 的 Error Model Annex 为 AADL 构件错误和构件通信错误提供了建模依据。它既可以为单个构件的错误建模,也可以为构件通信产生的错误传播建模,并且过滤不同构件类型之间的错误传播,分层组织构件的错误模型[6]。错误模型分为错误模型类型和错误模型实现。文中系统的错误模型描述如图 2 所示。

错误模型类型 Basic 定义了失效模态,它分为:初始错误状态、错误状态、错误事件、(input/output)错误传播。错误模型实现定义了错误状态迁移和属性,其中属性 occurrence 可以描述错误事件引起迁移的发生率。错误模型定义完成后,还需要在系统模型里把与错误模型相关的构件、构件通信通过附件子句关联起来,如图 3 所示。

系统错误模型由它所包含的各个构件错误模型复合而来,它与构件的错误模型类型和模型实现有关。系统错误模型可以通过分层来建模,它包括两种类型:一种是由用户定义的构件错误状态来决定,另一种是子构件错误模型复合而来的抽象表示。Model_Hierarchy 属性值为 derived,表示用户定义的子构件错误状态,Drived_State_Mapping 描述子构件的错误状态迁移,并从子构件的错误模型来进一步获取构件的错误模型,进而忽略子构件错误模型。而 Model_Hierarchy 属性值为 abstract 的错误模型,可设置错误传入和错误传出 Guard_In 和 Guard_Out 属性,Guard_In 属性能在

构件接收端口屏蔽错误传播。

```
Package Standard _Errors
Public annex error _model
{ * *
error model Basic
features
err _free ;initial error state;
loss _avail, loss_int ;error state;
fail_stop,fail_babble:error event;
loss_data,corrupt_data:in out error propagation;
end Basic;

error model implementation Basic. Hardware
transitions
err_free-[fail_stop,in loss_data]->loss_avail;
err_free-[fail_babble,in corrupt_data]->loss_int;
loss_avail-[fail_babble]->loss_int;
loss_avail-[in loss_data,out loss_data]->loss_avail;
loss_int-[in corrupt_data,out corrupt_data]->loss_int;
end Bsaic. Hardware;

error model implementation Basic. software
transitions
err_free-[in loss_data]->loss_avail;
err_free-[in corrupt_data]->loss_int;
loss_avail-[fail_babble]->loss_int;
loss_avail-[in loss_data,out loss_data]->loss_avail;
loss_int-[in corrupt_data,out corrupt_data]->loss_int;
end Basic. software;
* *};
end Standard_Errors
```

图 2　错误模型类型与实现

```
System implementation Dual. Basic
…
annex error_model { * *
model=>standard_Errors::Basic. Software;
model_hierarchy=>Derived;
   err_free when P1 or P2,
   loss_avail when P1[loss_avail] or P2[loss_avail],
   loss_int when others;
   report=>loss_avail,loss_int; * *};
end Dual. Basic;

process implementation SW_Proc. Basic
annex error_model { * *
   model=>Standard_Errors::Basic. software;
   occurrence=>fixed 1E=4 applies to error fail_stop;
   guard_in=>
      mask when Input_1[err_free] or Input_2 [err_free],
      corrupt_data when Input_1[loss_int] and Input_2[loss_
int],
      loss_data when others
applies to Input_1,Input_2; * *};
end SW_Proc. Basic
```

图 3　关联的错误模型

2　故障树的生成

故障树[7]是对系统进行安全分析的一种重要方法。从带有错误模型标识的 AADL 模型里提取故障树,通过将提取的故障树输入到故障树分析工具就可以自动化地分析系统模型的可靠性[8]。因此基于 AADL 错误模型来提取静态故障树是系统安全分析的关键。

故障树的生成是一个递归的过程,主要分成 3 个步骤:提取系统实例错误模型,生成中间故障树,优化中间故障树。

(1)系统错误模型实例提取:系统错误模型实例包括构件错误模型实例、通信错误模型实例以及系统实例。因此需要基于系统实例的各个构件错误模型来提取整个系统实例错误模型。在系统实例模型里有两种类型:构件实例和连接实例。构件声明和连接声明与错误模型密切相关,它们能够被直接用于获取实例。如果连接实例和错误模型没有直接联系,则错误模型的整个源应用于连接实例。对于构件实例,除了 model 属性,同样也需要获取 Model_Hierarchy、Derived_State_Mapping、Occurrence 和 Guard_In 的属性值;对于连接实例,只需获取 Occurrence 属性。标识错误传播源对于构件和连接实例是十分重要的,为了简化,文中将构件实例的错误传播源分为直接传播和间接传播。直接传播即错误传播发生在直接相连的端口上,间接传播即错误传播出现在不相邻的连接端口上。

以有向节点图来存储所有的信息,并且建立基于有向图的故障树模型[9]。对于 drived 错误模型的构件实例,节点指向其包含的下一层所有子构件。对于 abstract 错误模型的构件实例和连接实例,节点指向了其作为错误传播输入源的所有构件。然而,有些潜在的错误传播路径可能会在有向图里形成一个循环,要生成一个故障树就必须消除循环。

在文中事例里,SW 子构件 P1 有一个 abstract 错误模型,它的输入端口带有条件 Guard_In,错误传播能够直接从它的输入端口 Input_1 和 Input_2 进入,也可以间接从绑定的处理器 HW. P1 进入。而 HW 子构件 Data_Bus 也有一个 abstract 错误模型。基于错误传播规则,处理器 HW. P1 和 HW. P2 都能传播错误到 HW. Data_Bus 上。

(2)生成中间故障树:将系统实例错误模型以有向图的形式进行存储后,就进入到故障树的生成阶段。在 Report 属性里列出错误状态或传播的顶事件。基于图 3,Report 属性列出了两个错误状态 loss_of_availability 和 loss_of_integrity,它们被认为是生成故障树的顶事件。接着使用一些优化方法来对故障树的中间表示执行语法优化。例如去除多余的操作符、崩溃的门等等。更复杂的是在同一故障树里或者不同故障树之间共享子树。涉及简化冗余子树等优化故障树将在最后讨论。

(3)故障树生成:故障树以层次化的拓扑结构组

织系统故障模型。文中用系统故障树的子树 HW. Data_bus 的错误状态 loss_of_integrity 来说明这个过程，它的故障树如图 4 所示。

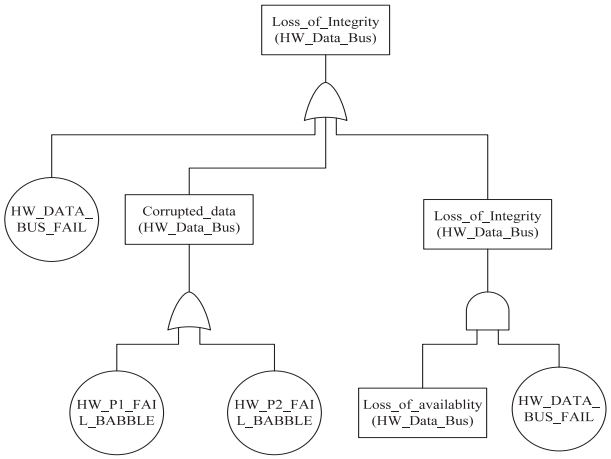


图 4 静态 HW. Data_bus 故障树

在 Basic. Hardware 错误模型实现里已经定义了错误迁移(如图 2)。HW. Data_bus 的错误状态可通过下面三种方式获得：

(1)构件在 error_free 状态，错误事件(fail_Babble)出现。

(2)构件在 error_free 状态，错误输入(corrupted_Data)发生。

(3)构件在 loss_of_availability 状态，错误事件(fail_Babble)发生。

图 4 是一个与三个输入有关的 OR 门。带有优先级的与门(priority－AND)可以获取错误迁移里的次序。这种有序的 AND 门在静态错误模型故障树里可用来做定量分析。例如，错误传播的路径总是从处理器 HW. P1 和 HW. P2 到总线上。以 HW. P1 的错误传播考虑，在错误状态 loss_of_integrity，它发射(out)错误传播。这里，总线也能够传播错误给 HW. P1，因此容易在故障树里产生一个环。要破坏这种错误的错误传播环。因此仅仅在内部错误事件 fail_Babble 下，才传播 corrupt_data，出现 fail_Babble 的两个处理器用 OR 门表示。

3 生成故障树的一点声明

即使最简单的系统架构，AADL 也能为它的各种故障和失败模式建立有效的模型。建立系统故障和失效模型的重要优点是能从总体上来把握系统模型的安全性。系统故障树是自顶向下分层建立的，不同故障子树里的同一错误事件在故障树里具有相同的名字，这便于共享失效的子树。门信息会帮助分析系统错误模型之间的联系，然后通过故障树分析工具就可以获取安全分析结果。不过目前也有一些挑战还需解决。一个潜在威胁是有些失效模式不能用错误模型描述。由于 AADL 误差模型附件的限制，有些特定失效模式类型应该单独分析。

故障树优化对自动生成故障树或故障树分析十分关键，目前已经有很多关于故障树的分析方法[10-12]。在这里仅仅讨论子树的共享和简化。以构件 SW 的实现 Dual. Basic 来说，由于 P1 和 P2 之间互为输入而彼此依赖，因此在系统架构中产生了循环。SW 有一个 derived 错误模型，它的错误状态和 P1、P2 子构件错误状态相关。如图 5 所示，P1 的左子树依赖 P2 的错误传播，而 P2 同样也依赖 P1 的错误传播，由此故障树里产生了一个循环。为了打破这个循环，需要记录下所有已经遍历过的节点。在该例中，当以 P1 作为错误传播源，如果祖先列表里包含(SW，P1，P2)，它就会探测到一个循环，然后忽略来自 P1 的错误传播。当然，也可以通过共享子树来判断特别的错误传播或者状态。在该例中，如果知道 P2 在右分支上，就不会在左分支上共享为 P2 节点所创建的同一子树，因为他们指向同一个错误状态或者传播，这是为了破坏右分支上的 P2 也是左分支所产生的循环。因此，需要为右边的 P2 重新生成一个故障树。

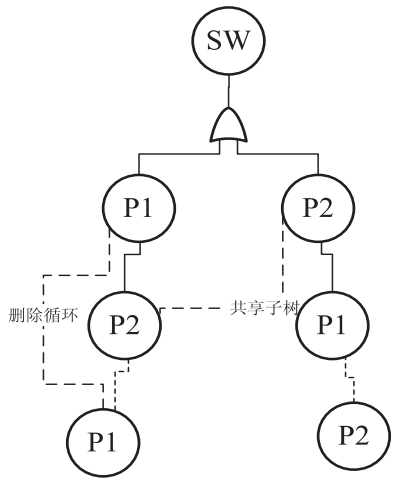


图 5 循环和共享子树

优化的目的是进一步简化和处理循环，让这些子树变成单调的。因此，必须在语言里引入一些运算符，由于不是所有的故障条件都能用单调故障树来表示，进一步的简化还需要很多工作要做。

4 结束语

文中介绍了基于 AADL 模型及其错误模型附件的自动产生故障树的生成方法。通过将生成故障树作为故障树分析工具输入可分析系统的安全性。这种方法对复杂实时系统设计初期的安全评估是十分重要的。同时，还提出自动生成这种安全模型的一些挑战。

(下转第 106 页)

便于测试,选择一些子网来执行[9]。为了监控整个子网的数据流,配置三层交换机,把三层交换机的主干接口镜像到另一个接口。然后,部署一台主机通过双绞线连接到交换机的镜像接口上面。在这台主机上运行钓鱼邮件监控系统后,就可以对网络中的邮件通信进行实时监控。经过数天的运行观测与数据分析,它能够捕获整个子网的数据流并且分析解码 SMTP 和 POP3 协议,然后告诉管理员网络中通信的钓鱼邮件情况。在实验中,截获了 70 多封邮件,检测到有 2 封是钓鱼邮件。图 4 显示了系统运行情况。总之,此监控系统在一定程度上保障了公共网络安全。

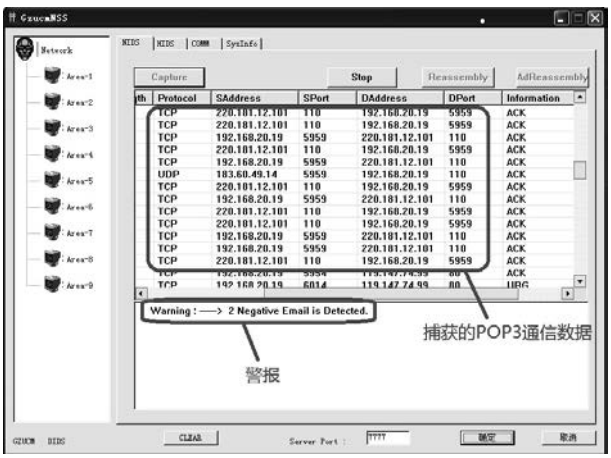


图 4 该系统在某校园网进行运行测试情况

4 结束语

由于互联网的发展和普通网民网络安全意识的亟待提高,在今后相当长的一段时间内,网络欺诈与网络钓鱼等欺骗技术会一直在互联网上发生,并且受到相关监控[10]。

文中设计实现了一个钓鱼邮件监控系统来保障公共网络安全。实验表明系统能检测到钓鱼邮件里面的敏感词并且有一个良好的警报效果。而如何检测邮件内容是图片且图片中存在钓鱼网站相关信息是下一步的工作。

参考文献:

[1] 吕述望,王昭顺,李 剑,等. 针对电子银行的网络钓鱼攻击及其防范策略[J]. 信息网络安全,2011(7):1-3.

[2] 朱 红,刘宝成,张 开. 规避网络钓鱼给证券行业带来的安全风险[J]. 信息安全与技术,2011(5):67-69.

[3] 丁岳伟. 基于 SMTP 协议电子邮件的还原[J]. 小型微型计算机系统,2002,23(3):290-293.

[4] 吴 勋,刘嘉勇. 基于网络数据包的邮件还原技术研究[J]. 通信技术,2011,44(4):124-126.

[5] 郑 魁,疏学明,袁宏永. 网络舆情热点信息自动发现方法[J]. 计算机工程,2010,36(3):4-6.

[6] 吴志强,马春波,敖发良. 基于 Winpcap 的邮件还原系统的实现[J]. 微型机与应用,2011,30(2):58-61.

[7] 何高辉,邹福泰,谭大礼,等. 基于 SVM 主动学习算法的网络钓鱼检测系统[J]. 计算机工程,2010,37(19):126-128.

[8] Liu Wenyin. An antiphishing strategy based on visual similarity assessment[J]. Internet Computing,2006,10(2):58-65.

[9] Fu A Y. Detecting phishing web pages with visual similarity assessment based on earth mover's distance[J]. IEEE Trans on Dependable and Secure Computing,2006,3(4):301-311.

[10] Raffetseder T. Building anti-phishing browser plug-ins:an experience report[C]//Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems. Austria:[s. n.],2007:1-7.

(上接第 102 页)

参考文献:

[1] 杨启亮,邢建春,王 平. 安全关键系统及其软件方法[J]. 计算机应用与软件,2011,28(2):129-138.

[2] 贾旭杰. 安全关键系统可靠性与安全性的研究与分析[M]. 北京:中国科学技术出版社,2011.

[3] SAE International. Architecture analysis and design references language(AADL)[S]. AS5506,2004.

[4] SAE-AS5506/1. Architecture analysis and design language annex volume 1[S]. 2006.

[5] 杨志斌,皮 磊,胡 凯,等. 复杂嵌入式实时系统体系结构设计与分析语言:AADL[J]. 软件学报,2010,21(5):899-915.

[6] ErrorModelAnnex-phf-JuneMtg2009[EB/OL]. 2009-04. http://www.aadl.info/aadl/documents/.

[7] 史定华,王松瑞. 故障树分析技术方法和理论[M]. 北京:北京师范大学出版社,1993.

[8] Li Yue,Zhu Yian,Ma Chunyan,et al. A method for constructing fault trees from AADL models[C]//Proc of the 8th International Conference on Autonomic and Trusted Computing. Berlin:Springer-Verlag,2011:245-258.

[9] 周建军. 基于有向图和故障树的导弹故障诊断系统研究[D]. 北京:航天工业总公司四部,2001.

[10] 李堂经,王新阁,杨 哲. 动态故障树的综合分析方法[J]. 装备制造技术,2009(8):22-23.

[11] 朱正福,李长福,何恩山,等. 基于马尔可夫链的动态故障树分析方法[J]. 兵工学报,2008,29(9):1104-1107.

[12] 高顺川. 动态故障树分析方法及其实现[D]. 长沙:国防科技大学,2005.

基于AADL模型的静态故障树的自动生成

作者： 刘玮， 李蜀瑜， LIU Wei， LI Shu-yu
作者单位： 陕西师范大学 计算机学院,陕西 西安,710062
刊名： 计算机技术与发展
ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2013(10)