

代 号 10701

学 号 0821421373

分类号 TP311.5

密 级 公开

西安电子科技大学

硕士学位论文



题 (中、英文) 目 基于 AADL 的嵌入式软件可靠性建模与评估

Reliability Modeling and Evaluation of the Embedded

Software based on AADL

作者姓名 高志伟 指导教师姓名、职称 李青山 教授

学科门类 工学 学科、专业 计算机软件与理论

提交论文日期 二〇一一年一月

西安电子科技大学 学位论文创新性声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：高志伟

日期 2011.3.4

西安电子科技大学 关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安电子科技大学。本人保证毕业后离校后，发表论文或使用论文工作成果时署名单位仍然为西安电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。（保密的论文在解密后遵守此规定）

本人签名：高志伟
导师签名：李丁

日期 2011.3.4

日期 2011.3.4

摘要

随着嵌入式软件的规模不断增大，复杂度迅速增加，嵌入式软件可靠性模型的建立越来越困难，模型不能很好地反映构件间的依赖关系。如何尽早建立软件可靠性模型，降低建模的复杂度，使模型反映出故障在构件之间的传播关系，是软件可靠性研究工作中的一个重要问题。AADL(Architecture Analysis and Design Language)作为一种针对嵌入式软件的体系结构建模语言，其概念框架为解决上述问题提供了途径。

本文在对软件可靠性理论和 AADL 进行深入分析和研究的基础上，提出了基于 AADL 的嵌入式软件可靠性建模方法。文中首先以 AADL 结构模型为基础，通过 AADL 流信息确定构件间的故障传播路径，刻画构件发生故障后对其他构件的影响，同时使用流终点构件确定系统的失效，刻画不同构件对系统可靠性的影响，之后采用 AADL 错误模型机制建立可靠性模型的形式化描述。在此基础上，通过将 AADL 可靠性模型转换为广义随机 Petri 网可靠性模型，实现了软件可靠性的自动化定量评估。最后，采用本文设计并开发的 AADL 可靠性评估器对一个典型实例进行可靠性评估与分析，验证本文提出的可靠性建模与评估方法的可行性。

关键词： 嵌入式软件 AADL 软件可靠性建模 广义随机 Petri 网

ABSTRACT

With the increasing size and complexity of embedded software, construction of embedded software reliability model is more difficult, which cannot reflect the dependencies between components. The issues of constructing software reliability model as soon as possible, reducing the complexity of modeling and making the model reflect the relationship of fault propagation between components in system are important. AADL (Architecture Analysis & Design Language) is an architecture modeling language that aims at embedded software. Its conceptual framework provides a way to solve these problems.

In this paper, the software reliability theory and AADL is analyzed and researched deeply and a method of reliability modeling embedded software is proposed. Firstly, based on the AADL architecture model, fault propagation path is determined through AADL flow information to describe the impact of the failure of component on other components and the failure of system is determined through the component that is the sink of flow to describe different impacts of each component on system. In addition, the paper uses AADL error model mechanism to describe reliability model, which can support formal description. On this basis, through the transformation of AADL reliability model to generalized stochastic Petri-Net reliability model, the automated quantitative evaluation of software reliability is available. Finally, the paper presents a case of reliability evaluation and analysis by the AADL reliability evaluation system proposed, which verifies the feasibility of the method.

**Keyword: Embedded Software AADL Software Reliability Modeling
Generalized Stochastic Petri Net**

目录

第一章 绪论.....	1
1.1 论文背景.....	1
1.2 研究内容与意义.....	2
1.3 国内外研究现状分析.....	4
1.3.1 软件可靠性模型研究现状.....	4
1.3.2 AADL 研究现状.....	5
1.4 本文的组织结构.....	6
第二章 软件可靠性与 AADL.....	7
2.1 软件可靠性理论.....	7
2.1.1 软件可靠性的定义.....	7
2.1.2 软件失效机理.....	8
2.1.3 软件可靠性特点.....	9
2.1.4 软件可靠性工程.....	9
2.1.5 软件可靠性度量参数.....	10
2.1.6 软件可靠性模型.....	11
2.2 AADL 及其工具.....	12
2.2.1 AADL 概念框架.....	12
2.2.2 AADL 附件.....	14
2.2.3 AADL 开发环境.....	14
第三章 基于 AADL 的嵌入式软件可靠性建模评估框架.....	17
3.1 可靠性建模框架.....	17
3.2 可靠性评估框架.....	18
3.3 可靠性评估工具.....	19
第四章 基于 AADL 的嵌入式软件可靠性建模.....	21
4.1 错误模型建立框架.....	21
4.2 单独构件模型.....	22
4.3 故障传播.....	24
4.3.1 故障传播的过程.....	24
4.3.2 错误模型的故障传播描述.....	25
4.3.3 故障传播路径.....	26
4.4 系统失效模型.....	27

第五章 基于AADL的嵌入式软件可靠性评估	31
5.1 广义随机Petri网	31
5.1.1 Petri网模型	31
5.1.2 GSPN	32
5.2 GSPN可靠性模型	33
5.2.1 建模方法	33
5.2.2 可靠性评估原理	35
5.3 AADL可靠性模型到GSPN可靠性模型的转换	36
5.3.1 单独构件的模型转换	36
5.3.2 故障传播的转换	37
5.4 可靠性评估器的设计与实现	38
5.4.1 体系结构设计	38
5.4.2 评估自动化	39
5.4.4 可靠性分析方法	41
第六章 实例验证	43
6.1 飞行管理系统	43
6.2 可靠性建模	44
6.3 模型转换	45
6.4 评估器运行结果	46
第七章 总结与展望	49
致谢	51
参考文献	53
在研期间研究成果	57

第一章 绪论

1.1 论文背景

本论文的工作来源于航空科学基金项目“×××领域的软件可靠性评估方法研究”。随着半导体技术的快速发展，处理器计算能力得到了飞速提高，嵌入式计算机系统的应用日渐广泛，在航空、航天、武器装备、通讯、工业控制、金融等各个领域中都发挥了关键的作用，但是，嵌入式计算机系统中存在的可靠性问题也引起了人们的关注，在诸如核电站控制系统、医疗系统、航空电子系统中的嵌入式计算机系统一旦发生失效，就有可能造成财产损失，甚至人员伤亡的重大事故。

由于人们需求的与日俱增和信息技术的发展，嵌入式计算机系统的功能更多地由软件来实现，这导致了软件系统的规模越来越庞大，复杂程度超过了目前软件开发和维护的能力。而相对于硬件可靠性技术，软件的可靠性研究还相对落后。这些原因使得软件产品的可靠性无法得到有效的保证，软件成为了计算机系统发生失效的主要元凶。

1991年海湾战争中，由于爱国者导弹中的软件产生了时间误差而未能成功拦截伊拉克发射的“飞毛腿”导弹，最终造成28名美国士兵丧命，98人受伤。1992年，因为计算机辅助派遣系统中的软件发生失效，世界上最大的救护服务系统完全瘫痪，上千人受到影响。1996年，欧洲航天局发射阿丽亚娜5号火箭失败，直接损失5亿美元，更使耗资超过80亿美元的开发计划推迟了近三年。经过调查，事故的原因是火箭控制系统中的软件发生失效。1999年在卡那维拉尔角发射通讯卫星时，由于软件问题火箭在起飞后偏离了预定航向，导致上面级火箭“非正常点火”，卫星提前打开进入了错误的轨道。在地面控制工作试图纠正卫星位置时，又由于火箭燃料耗尽而没有成功。最终卫星未能进入预定的地球同步轨道，造成发射失败。一次又一次的惨痛经验教训使人们更加认识到了软件可靠性的重要性，各国都加大了对软件可靠性相关技术的研究力度。经过数十年的研究，软件可靠性的概念已经深入人心，以应用为目标的软件可靠性工程也逐渐受到了认可和重视。

与此同时，为解决“软件危机”，业界不断尝试提出新的软件开发方法。其中模型驱动工程^[1,2](Model-Driven Engineering, MDE)方法的提出引起了广泛的注意，得到了大力提倡。在MDE中，模型是软件开发过程中各个阶段的主要产品，开发人员集中于问题域而非解决域，即模型的建立。而开发中的各个阶段通过模型转

换来衔接，最终产品的实现通过代码的自动生成技术完成。实质上，模型是一个概念框架，能够对系统的各种特征进行抽象，并通过严格的语义和直观的符号进行描述。这使得参与软件开发的开发人员都能够通过模型来理解待开发的系统，尽可能消除沟通中的理解歧义，保证了软件产品的设计与实现的一致性。

AADL^[3](Architectural Analysis and Design Language)是专门针对安全关键、挑战资源限制、强实时响应的嵌入式实时系统领域的体系结构建模语言，它的前身是 MetaH 语言，由霍尼韦尔实验室研发并成功应用在航空电子系统中软件系统的研发上，后来经过其他研究机构的研究和扩充和美国自动机协会的推动，于 2004 年 11 月发布了第一个标准版本^[4]。AADL 的特点是对嵌入式系统的各种关键性能属性有较强的描述能力，支持对体系结构的早期分析。AADL 以构件为单位，遵从迭代开发原则，在软件开发周期中的不同阶段对应不同精度的 AADL 模型。随着开发的不断推进，通过模型中构件的演化和精化来不断细化模型，最终得到待开发系统的相关设计决策，包括构件属性、动态模式、可靠性设计等。AADL 在非功能属性方面，能够对时间约束、调度性、安全性等进行精确建模。这些关键特性使得 AADL 受到了卡内基梅隆大学、空中客车公司、欧洲宇航局等学术界和工业界的关注和重视。

1.2 研究内容与意义

在软件工程中，越早发现软件开发中潜在的问题并解决，项目的风险就越小。目前的软件可靠性评估活动主要集中在软件测试阶段，根据测试中收集到的可靠性数据进行建模和评估，当发现软件可靠性无法达到用户的需求时，通过更多的测试进一步减少软件中存在的缺陷来提高系统整体的可靠性。这种方法无法对软件的设计进行评估，使软件的缺陷一直保留到了软件产品实现后，加大了软件可靠性保证工作的难度。

体系结构是在软件开发过程之初产生的，是对系统的高层抽象，因此设计优质的体系结构可以减少和避免软件错误的产生和维护阶段的高昂代价^[5]。如何在早期的设计阶段对系统的体系结构进行评估和分析是目前软件可靠性研究中亟待解决的问题。MDE 中模型提供的概念框架为解决该问题提供了可能。

模型是对系统的抽象描述和刻画。通过模型，可以捕获、验证、实现整个系统，这就为在软件生命周期的早期分析、发现隐患和错误提供了可能^[6]。但是对于模型驱动工程的研究主要集中于建模语言的表达能力、模型之间的转换、代码自动生成等内容上。模型的早期分析与验证这一重要课题却较少涉及，而 AADL 的开发目标之一就是能够对系统的非功能属性如可调度性和可靠性进行分析和验证^[7]。从软件可靠性角度对 AADL 体系结构模型进行评估和分析，可以尽早对软

件的可靠性状态进行评价，发现软件体系结构设计中存在的隐患，为选择和优化体系结构设计提供依据。

另一方面，随着嵌入式软件规模和复杂度的增大，利用现有的软件可靠性模型技术进行建模是一件难度大、易出错的工作。同时，建立的嵌入式软件可靠性模型不能对构件之间的依赖关系引发的故障传播进行清晰地刻画。AADL 错误模型能够更好地描述系统中故障在构件之间的传播关系，反映单独构件的故障对其他构件的影响。

综上所述，在早期对软件的可靠性进行建模，反映故障传播关系，估测系统目前的可靠性状态，分析并发现软件体系结构设计中的可靠性隐患是软件开发的要求；而基于 AADL 的模型驱动工程方法又为软件可靠性的早期建模和评估提供了可能。基于此，本文做了以下的工作：

(1)研究软件可靠性的相关理论技术和基于 AADL 的模型驱动工程方法。对软件可靠性工程中的基本概念，软件可靠性模型进行了分析和总结；对基于 AADL 的建模方法、模型转换、体系结构分析等理论进行了研究和总结。

(2)研究基于 AADL 的嵌入式软件系统的可靠性建模方法。为反映嵌入式软件的失效行为，错误模型的建立以单独构件模型为基础，通过 AADL 流信息确定构件间的故障传播路径，以流终点构件定义系统失效，最终和 AADL 结构模型共同构成嵌入式软件的可靠性模型。同时使用 AADL 对可靠性模型进行形式化描述，为自动化可靠性工具提供了支持。

(3)研究基于 AADL 的嵌入式软件系统的可靠性评估方法。与建模方法结合，通过模型转换，将 AADL 可靠性模型转换为广义随机 Petri 网可靠性模型，对嵌入式软件进行可靠性评估和分析。以此判断软件目前的可靠性水平和发现未来可能影响系统可靠性的重要模块，为体系结构设计的选择与优化提供依据。

(4)设计并开发基于 AADL 的可靠性评估器。结合项目需要，利用本文的方法设计并实现了 AADL 可靠性评估器，该评估器通过对 AADL 模型文件进行解析，获取结构模型和错误模型中的相应信息并转换成广义随机 Petri 网模型，继而对系统的可靠性进行定量评估与分析，为模型驱动工程提供自动化工具支持。

本文的工作为基于 AADL 嵌入式软件开发方法中的可靠性保证工作提供了方法和工具的支持。该方法降低了可靠性评估工作的复杂性，建立的可靠性模型也能更好地反映构件之间的故障传播关系。在工具的帮助下可以实现可靠性评估工作的自动化，提高了软件开发的效率，降低了出错率。

1.3 国内外研究现状分析

1.3.1 软件可靠性模型研究现状

早期的软件可靠性研究集中于黑盒软件可靠性模型，研究人员试图通过统计分析工具从本质上理解软件可靠性行为和特点。第一个发表的软件可靠性模型是 1967 年 Hudson 的生灭过程模型^[8]。该模型将软件开发过程看成是一个生灭过程，假设软件错误的纠正率与剩余错误数量及其时间的某个正次幂成正比，且软件中的错误产生是延期的，错误纠正是死亡期，任意时刻软件中存在的错误个数是过程的状态参数，由此该模型导出了故障间隔的威布尔(Weibull)分布。之后在 1972 年，B. Littlewood 和 J. L. Verrall 发表了第一个贝叶斯模型^[9]，他们假设故障间隔时间服从含参数的指数分布，且服从先验分布。这样就可以由标准贝叶斯过程获得下一次软件的故障时间。1975 年，首先提出软件可靠性工程概念的 Musa 认为执行时间应分为软件测试时间和软件交付运行后的执行时间，而计时时间不能说明程序在运行或测试中的不同用途，称为 Musa 执行时间模型^[10]。同年，A. K. Trividi 和 M. L. Shooman 发表了第一个马尔可夫过程模型^[11]，将系统状态区分为正常和非正常两个，输出结果为一个概率集合，参数是一个时间段内发现的错误数。N. F. Schneidewind 发表了第一个非齐次泊松过程模型^[12]，他使用离散的时间段来构造模型，并对指数函数等进行了研究，提出应根据项目的实际开发情况，选用适合的可靠性函数，以估测软件的可靠度。1976 年，M. L. Shooman 和 S. Natarajan 发表了第一个不完美排错模型^[13]，提出了错误率、纠错率以及新错误引入率等概念。

随着软件产业的发展，软件体系结构设计得到了广泛应用，人们希望利用已知的软件结构来更精确地评价软件的可靠性，相继不断有白盒软件可靠性模型发表。早期由 Cheung 提出的模型^[14]认为软件的可靠性行为符合离散时间的马尔可夫链，通过软件构件之间的控制流可以生成有向图继而建立模型。Shooman 提出了一种基于路径的模型^[15]，认为一组构件的执行序列就是一条路径，根据不同路径的执行概率及路径上所有构件的失效率，给出系统的可靠性评价。Xie 和 Wohlin 提出了一种累加模型^[16]。该模型假设系统是由 n 个构件组成的，且每个构件都是单独开发和测试的，构件的失效率使用非齐次泊松过程建模，每一次构件的失效都认为是一次系统失效，最后每个构件的失效率函数相加就是系统的失效率函数。

国内的软件可靠性研究始于 80 年代初。武汉大学的徐仁佐教授在对软件可靠性建模进行深入研究的基础上开发了软件可靠性专家系统 SRES^[17]。北京航空航天大学的蔡开元教授创立并发展了模糊可靠性理论，被称为“蔡氏模型”，受到了国外的较高评价，并应用于“主动控制技术验证机”(ACT)上^[18,19]。陆民燕对软件可

可靠性的实用性进行了研究，如软件可靠性增长模型的组合应用^[20]和基于失效数据的软件可靠性度量框架^[21]。

随着一些新技术的出现和逐渐成熟，软件可靠性的研究也借助这些工具进一步推进软件可靠性研究的发展。N.Karunanith, Y.Malaiya 和 D.Whitley 将神经网络理论应用于软件可靠性的预测中^[22]。实验结果显示其一致性良好，精度较传统模型显著提高。文献[23]将随机 Petri 网应用于软件可靠性建模中，模型直观，同时利用随机 Petri 网相关理论分析了软件中的可靠性隐患。

综上所述，软件可靠性的研究经过数十年的发展取得了丰硕的成果，已经从早期的理论研究逐步转向工程实用。但这也暴露了软件可靠性研究存在的问题。一方面软件可靠性模型的建立非常复杂，使软件可靠性评估工作成为了一项难度大，出错率高的工作。另一方面软件可靠性的建立没有和软件工程的新方法结合，在软件设计的早期进行软件可靠性建模和评估，尽早发现问题。

1.3.2 AADL 研究现状

在模型驱动领域，AADL 还是一个较前沿的规范标准，国内外研究均处于萌芽状态。但是 AADL 已经得到了如卡耐基梅隆大学软件工程研究所、空中客车公司、霍尼韦尔公司、欧洲宇航局等权威机构的重视，日后很有可能成为嵌入式领域建模语言的事实标准。

一种建模语言不可能面面俱到，每种模型都有其优点和缺点。为了充分发挥各种模型的优势，应用于不同的场合，模型转换是模型驱动领域内一个重要的研究课题。Delanote 等人在文献中提出了从 UML 模型到 AADL 模型的转换规则^[24]，目的是将 UML 模型作为需求模型。而 STOOD 软件^[25]实现了 HOOD 模型到 AADL 模型的自动化转换。法国的 INRIA 团队开发了 ATL(Atlas Transformation Language)^[26]，它是一种基于规则的模型转换语言，使用 Eclipse 建模框架 EMF (Eclipse Modeling Framework)来描述其元模型及模型，在 AADL 模型转换的研究中比较常用。

在模型验证与分析方面，目前主要是通过仿真和形式化分析方法来实现。ADeS(Architecture Description Simulation)^[27]是一个基于事件驱动的行为仿真工具，支持对 AADL 模型的执行时间、调度策略和模式的分析。法国 Brest 大学 LISYC 小组对 AADL 模型可调度性分析方法进行了研究，并研发了 Cheddar 工具^[28]。Cheddar 通过仿真方法，根据经典调度算法，实现了对实时调度、多处理器、端到端任务和资源共享等情况下的可调度分析。在可靠性验证方面，也有一些工作成果。Rugina 研究了 AADL 模型到随机 Petri 网模型的转换规则以进行可靠性分析，^[29,30]。Boudali 等人提出了 Arcade(Architectural Dependability Evaluation)^[31]，它是

一种可扩展的可靠性评价框架，可以将包括 AADL 和 UML 在内的多种建模语言作为输入模型转换到 IO-IMC(input/output interactive Markov chains)模型，并通过 CADP 工具进行可靠性分析。文献[32]将故障树作为可靠性、安全性分析的工具，研究了 AADL 模型到故障树的转换。

从对目前的国内外研究现状的分析中可以看出，AADL 领域内的许多研究工作都是利用已有的较为成熟的技术，通过模型转换方法实现的。在可靠性验证分析方面也是如此，研究的重点是相关的模型转换规则。本文的重点是研究如何建立合适的 AADL 可靠性模型以反映系统的失效行为特点，降低可靠性建模的复杂度和难度，并通过工具实现可靠性评估的自动化。

1.4 本文的组织结构

本文对基于 AADL 的嵌入式软件的可靠性建模与评估方法进行了研究，其组织结构如下。

本章为绪论，在对本文的研究背景进行阐述的基础上引出本文的研究内容和意义。继而阐述和分析了国内外研究现状以说明本文所做工作的创新性。

第二章，基于本文的研究内容介绍了软件可靠性的相关理论和 AADL 语言的相关概念和工具集，为本文后续研究工作的阐述提供理论基础和技术储备。

第三章，给出了基于 AADL 的嵌入式软件可靠性建模、评估框架，介绍了支持自动化评估的可靠性评估工具。

第四章，给出了建立嵌入式软件 AADL 可靠性模型的方法，从单独构件、故障传播以及系统失效三个方面阐述 AADL 可靠性模型的建立方法。

第五章，给出了 AADL 可靠性模型的定量评估方法。AADL 可靠性模型仅是对系统可靠性行为的描述，而广义随机 Petri 网具有直观的描述手段和良好的分析能力，通过从 AADL 可靠性模型到广义随机 Petri 网可靠性模型的转换，可以对 AADL 可靠性模型进行定量评估。在此基础上，对 AADL 可靠性评估器的设计与实现进行了介绍。

第六章，通过使用 AADL 可靠性评估器对一个实例进行可靠性评估和分析，验证了本文提出的方法的可行性，降低了可靠性建模工作的复杂度。

最后一章对本文阐述的工作进行了总体概括，指出了其中的不足，对下一步的工作进行了展望。

第二章 软件可靠性与AADL

2.1 软件可靠性理论

2.1.1 软件可靠性的定义

软件可靠性作为软件产品固有的特性之一^[33]，是软件质量因素中最基本、最重要的因素^[34]。但是人们并不是一开始就认识到了这一点。随着因软件可靠性而引起的灾难性后果不断出现，人们加强了对该问题的研究。许多国家都投入了大量资源在这一领域的研究和实践上。如欧洲信息技术研究与发展战略(ESPRIT)计划和尤里卡(EUREKA)计划都对软件可靠性进行了大量研究。在1983年，IEEE计算机学会对软件可靠性给出了专门的定义：

软件可靠性是指1) 在规定的条件下，在规定的时间内，软件不引起系统失效的概率，该概率是系统输入和输出的函数，也是软件中存在的缺陷的函数；系统输入将确定是否会遇到已存在的错误(如果错误存在的话)；2) 在规定的周期内，在规定的条件下软件执行规定功能的能力。

该定义从定量和定性两个方面对软件可靠性进行了解释。这两方面的定义都是从三个要素出发来规定软件可靠性，即时间，条件和功能。

- 时间

时间是指软件被实际运行的时间。软件的可靠性只在其实际运行的阶段才会体现，所以时间的理想度量单位是“执行时间”。但是有时软件的执行时间较为困难，所以也可用日历时间来作为软件可靠性的度量时间。在软件可靠性的研究中，执行时间是一种随机变量。

- 条件

条件是指软件运行时所处的环境。它包括支持软件系统运行的所有要素，比如硬件环境、操作系统、其他相关联的软件、输入的数据以及操作流程等。在不同的运行条件下，软件所表现出的可靠性是不同的。“规定的条件”是指对软件系统所运行的硬件配置和输入数据等的要求。

- 功能

功能是人们使用软件要完成的任务。软件的操作流程随着完成任务的不同而有所区别，所调用的子模块或者说程序的执行路径就有所不同。显而易见，不同

的程序执行路径导致软件的可靠性发生改变。所以欲评价一个软件的可靠性，首先要定义软件系统所要完成的功能。

2.1.2 软件失效机理

由于在早期对软件可靠性进行研究时，硬件可靠性技术的研究已经取得了一定的成功，形成了较为完善的体系。所以人们在研究软件可靠性时拿硬件可靠性进行类比，试图借鉴硬件可靠性中的相关理论和技术。但是结果却总是与人们的期望大相径庭。在不断深入的研究后，人们发现软件失效的机理与硬件截然不同的，所以不能照搬硬件可靠性技术。要了解软件可靠性特点，必须先理解软件的失效机理。软件的失效过程包括软件错误的引入、软件缺陷的存在、软件故障的发生和软件失效的感知。

软件错误指的是在软件的寿命周期内的一种人为引起的人们不希望的或不可接受的差错。相对于软件本身，软件错误是一种外部行为，其结果是软件缺陷的产生。

软件缺陷是指隐含在于软件(文档、数据、程序)之中的那些不希望或不可接受的偏差，如语义的错误或语句执行路径的错误等。软件缺陷可能被激活，即当软件运行到某一特定状态而执行软件缺陷部分时出现软件故障。

软件故障是指在运行过程中，软件所表现出来的一种不希望的或不可接受的内部状态。例如，当软件处于执行一个死循环而无法输出结果时，称软件出现了故障，软件故障是软件的一种动态行为。

软件失效是指软件运行过程中发生的一种不希望或不可接受的外部行为结果。当软件发生故障后，若无适当的措施即时进行处理，用户感知到了软件内部的故障状态，这时便称软件发生了失效。

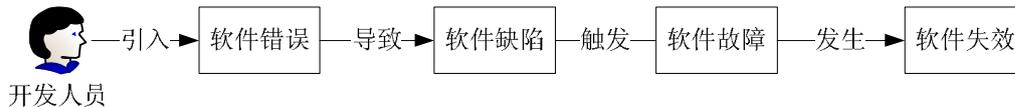


图 2.1 软件的失效机理

如图 2.1 展示了上述的软件失效机理。从错误的引入到缺陷的触发，每一个过程都具有随机性，所以软件的失效行为具有随机性^[35]。所谓随机性是指不能确定变量的取值。这种情况通常发生于当影响变量取值的因素过多时，以至于预测是不实际的。但是可以确定每个变量可能的值的集合及其出现概率。在软件可靠性研究中，虽然无法准确预知下一次失效在何时发生，但是可以知道所有可能的失效发生时间，以及每个失效时间出现的概率。软件失效表现出随机性有两个原因。第一是开发人员在开发软件的过程中发生错误是一个很复杂不可预测的过程。因此程序中错误的位置是未知的。第二，程序的执行环境条件一般是不可预测的。

尽管理论上可以知道代码路径之间的关系，但是实际中却因为太复杂而无法确定。由于失效依赖于代码中存在的错误，以及是否执行了错误代码，因此，软件的失效过程是随机的。

2.1.3 软件可靠性特点

根据软件失效机理，可以总结软件可靠性的特点和硬件可靠性的不同之处。弄清软件可靠性的特点是研究软件可靠性的基础。

与硬件相比，软件是人类脑力的产物，只能记录在纸张或存储介质中。软件失效是由于程序运行过程中软件缺陷被触发所致。软件缺陷是在进行软件设计工作时因疏忽等人为因素而引入的。硬件是建立在物质的基础上，由于受到物理和化学的作用，物质的性质和结构会发生变化，致使硬件发生损耗性失效。

在软件失效时，软件缺陷可以被发现出来并被排除掉，因此软件的可靠性会在软件测试的过程中不断提高，并趋于稳定。虽然软件在经过一段时间的使用之后，也会被废弃，但是主要原因是该版本软件的功能已不能满足用户新的需求。硬件报废的主要原因则是经过长时间的使用，产品已经进入了损耗失效期，无法继续正常稳定地工作。

在使用环境方面，影响软件失效的环境因素是指软件开发环境与使用环境的差异，包括软件平台和硬件平台。而软件的可靠性不受外部自然环境的影响。硬件失效与使用现场的自然环境或人为环境密切相关，因此硬件可靠性的高低更依赖于环境因素。

因此，硬件的可靠性，除了设计因素之外，受到物质老化过程的影响更大。而这种物理化学变化是无法避免的。相比之下，影响软件可靠性的因素主要是设计质量。这里所指的软件设计包括了从需求分析开始，直至实现软件，软件维护的软件生命周期全过程。

2.1.4 软件可靠性工程

软件可靠性工程的概念是穆萨在 1991 年的第十三届国际软件工程会议上正式提出的，它是指为有效实现软件可靠性目标而采取的系统化技术、方法，管理等活动。经过十余年的发展，逐渐被业界接受，并明确了软件可靠性工程不仅包含软件可靠性模型和软件可靠性度量，还包括可靠性的需求及其可行性，设计和实现，测试和验证，交付后等活动。但是到目前为止，尚无一个被业界广泛认同的软件可靠性工程定义，还有待权威的标准化机构作出定义。

图 2.2^[34]展示了软件可靠性工程活动在软件生命周期中的相应活动。在需求分

析阶段，首先要明确软件可靠性的参数，继而为每个模块分配指标，明确软件可靠性的目标，并对提出的可靠性目标进行可行性分析验证；在设计开发阶段需要采用软件可靠性分析技术确定潜在的可能导致软件失效的隐患、缺陷、薄弱环节，针对这些环节要进行可靠性设计，提高系统的整体可靠性；在测试阶段则需要通过软件可靠性测试技术有效地排除软件中的缺陷并收集可靠性度量数据，提高软件的可靠性，并对软件可靠性是否达到可靠性要求进行评价、验证，以确定是否可以交付，是否满足使用方的要求。

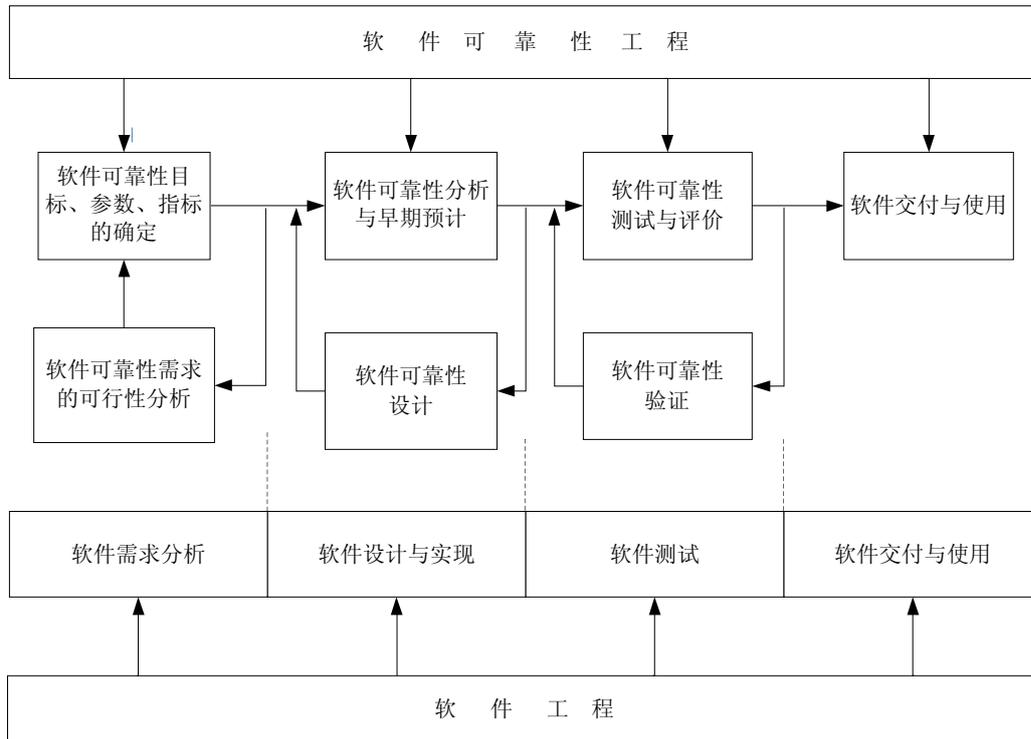


图 2.2 软件可靠性工程活动

2.1.5 软件可靠性度量参数

软件可靠性度量是指对软件产品具有的可靠性程度的定量评价。软件可靠性度量参数(常简称软件可靠性参数)是描述软件可靠性的依据，是验证、分析软件可靠性的必要步骤。

- 可靠度

根据软件可靠性定义，软件可靠度指软件在规定的运行环境中和时间，不发生失效的概率。

$$R(t) = P(T \leq t)$$

式 2-1

- 失效概率

失效概率是失效时间少于或等于 t 的概率，根据其定义可知它和可靠度 $R(t)$ 之间存在如下联系

$$F(t) = 1 - R(t) \quad \text{式 2-2}$$

- 失效强度

失效强度是失效概率的密度函数，如果 $F(t)$ 是可微分的，失效强度 $f(t)$ 是 $F(t)$ 关于时间的一阶导数，即

$$f(t) = \frac{dF(t)}{dt} = -\frac{dR(t)}{dt} \quad \text{式 2-3}$$

- 失效率

失效率是指在 t 时刻尚未发生失效的条件下，在 t 时刻之后的单位时间内发生失效的概率。失效率是失效概率 $F(t)$ 的条件概率密度，又称条件失效强度。即

$$\lambda(t) = \frac{f(t)}{R(t)} = \frac{dR(t)/dt}{R(t)} \quad \text{式 2-4}$$

- 平均失效前时间

平均失效前时间 (Mean Time To Failure, MTTF) 是指当前时间到下一次失效时间的均值，按照可靠度的定义，有

$$MTTF = \int_0^{\infty} R(t) dt \quad \text{式 2-5}$$

- 平均失效间隔时间

平均失效间隔时间 (Mean Time Between Failure, MTBF) 是指两次相邻失效时间间隔的均值，若软件从 T_1 时刻到 T_2 时刻，发生了 n 次失效，则有

$$MTBF = \frac{T_1 - T_2}{n + 1} \quad \text{式 2-6}$$

- 可用度

产品在任一时刻 t 时，需要和开始执行任务时，处于可工作或可使用状态的概率。

$$A(t) = P(T = t) \quad \text{式 2-7}$$

针对不同系统的特点，需要使用不同的软件可靠性度量参数。本文针对嵌入式软件系统，特别是高可靠性嵌入式软件可恢复以及需要和硬件系统的可靠性度量参数统一的特点，选择可用度作为度量参数。

2.1.6 软件可靠性模型

软件可靠性模型是软件可靠性工程研究的基础，它试图从本质上理解软件的

可靠性行为。在软件可靠性领域中，软件可靠性模型的研究一直占据重要地位，也是成果最丰富的领域。

前面已经提过软件可靠性研究中的随机性。软件可靠性模型是以数理统计理论为数学工具，以软件的失效数据，内部结构等信息为依据建立数学模型，对软件目前的可靠性性能进行度量，进而对软件将来的可靠性状态和行为进行预计和推断，如未来可能发生失效的时间段，剩余的软件缺陷等。

通常，软件可靠性建模在软件开发的需求分析阶段^[35]，与软件可靠性指标体系的建立同步开展，协调进行，支持将可靠性需求分配给拟开发软件及其运行剖面。

目前发表在文献上的软件可靠性模型多达上百种，可以称之为模型大战。文献[17]中提出了选择软件可靠性模型的一般原则是模型中理论假设的相对合理性；数学上的易处理性；成熟度高，使用频繁；典型代表性，能概括不同的建模特点和工具易处理性。

2.2 AADL 及其工具

2.2.1 AADL 概念框架

在 AADL 提供的概念框架中，构件是最基本的元素。与传统意义上的构件概念不同，它代表了嵌入式系统中的不同元素，以此抽象出整个系统的体系结构。在 AADL 嵌入式系统模型中，软件构件分为线程(thread)，线程组(thread group)，进程(process)，数据(data)，子程序(subprogram)五种，分别代表了应用软件中的不同实体。应用软件的执行和数据的存储需要硬件的支持，即软件的执行平台。AADL 将执行平台分为处理器(processor)，存储器(memory)，设备(device)，总线(bus)。最后，为了体现层次关系，方便对大型复杂系统的分析，AADL 提供了组合类构件，即系统(system)。

和面向对象的实现方法类似，在 AADL 中构件的表达也是由类型及其对应的实现共同描述。其中在构件类型中需要说明一个构件从外部可见的属性、特性和接口。而在构件实现中是对构件的内部结构进行说明，例如表现不同运行条件下系统不同状态的模式、内部隐含的属性、其所包含的子构件、子构件间的交互特性、各子构件之间流动的数据。在现代软件设计中，一般构件的接口较为稳定，而内部实现则可以有很多版本。因此在 AADL 中，构件类型可以对应多个构件实现。

在 AADL 中，软件构件之间的交互使用连接(connections)来表达；一个连接将

一个构件的 out 端口和另一个构件的 in 端口相连，表示组件间存在控制流或数据流。这种连接有四种方式，包括端口连接(port connections)，构件访问连接(component access connections)，子程序调用(subprogram calls)，参数连接(parameter connections)。AADL 流(flows) 能够表示系统中实际的信息流动，用于对系统中抽象的信息路径进行详细描述和分析。针对嵌入式系统，AADL 中的另一种构件间的交互是软件构件到执行平台构件的绑定(bindings)，一个应用程序构件必须和执行平台构件绑定，来表示系统中硬件资源的使用情况。例如，一个线程必须绑定到一个处理器。

属性(properties)提供了关于构件的描述信息，如线程的优先级等。在 AADL 中内建了各种不同类型构件的属性，也允许自定义的属性来描述构件。属性集是被命名的有共同特性的一组属性。

模式(mode)表现了一个系统或者构件的可选的操作状态,如飞机的起飞、降落、巡航等阶段都可定义为飞行系统的模式。在不同的模式下，系统的软硬件运行状态都会发生变化，借由模式概念，可以表达不同状态下系统的软硬件组成。系统可以定义多个模式以及触发模式转换的事件，以此对系统运行时的动态变化情况进行建模。

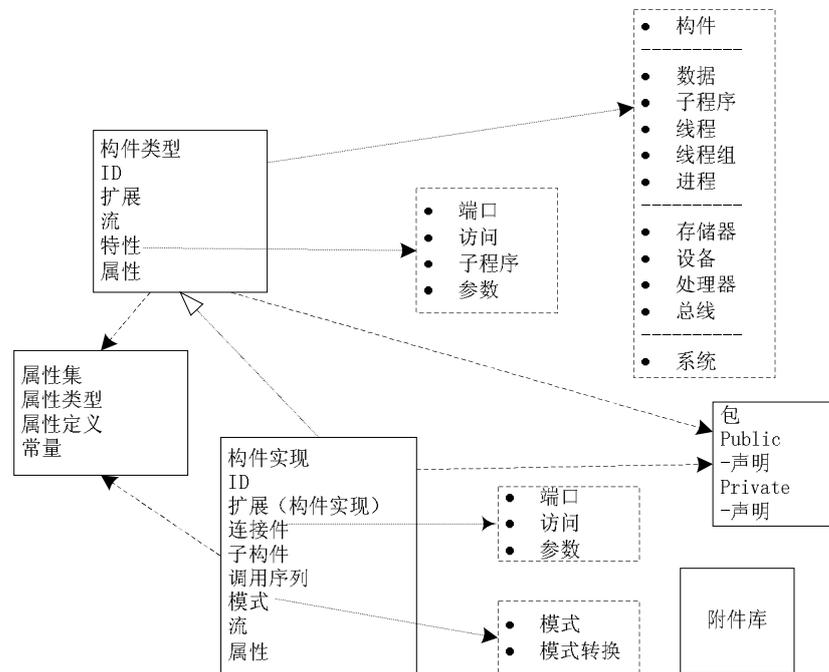


图 2.3 AADL 概念框架

AADL 不仅支持构件实现继承构件类型，也支持类型之间、实现之间的继承关系，包括构件类型扩展、构件实现扩展、构件实现的精化(refines)几种方式。通过以上机制，AADL 模型的重用度大幅提高，逐步精化的迭代设计方法也得到了良好的支持。

在 AADL 中,包(package)的概念允许一组构件被组织为拥有独立命名空间的单独单元。拥有共性的元素可以构成一个包,并且可以通过包名来引用。包所提供的名称空间可以帮助我们进行大规模系统的分解开发。图 2.3^[7]展示了 AADL 中的全部语言要素。

2.2.2 AADL 附件

AADL 通过引入附件(annex)机制来提供扩展性。当系统内建的属性和语义不能满足用户对构件的描述需求时,附件概念可以帮助解决这个问题。它拥有独立的语法和语义,但必须与 AADL 核心标准保持语义的一致。

为了不断增强 AADL 的描述和分析能力,AADL 委员会已核准了一些标准附件补充到 AADL 标准中。如错误模型附件(Error Model Annex)、AADL 元模型和 XMI/XML 附件(AADL Meta Model and XMI/XML Annex)、图形化 AADL 符号附件(Graphical AADL Notation Annex)等。还有如行为附件(Behavior Annex)和 ARINC653 附件(ARINC653 Annex)正在制定和审核中。

目前研究较多的有错误模型附件、行为附件。错误模型附件支持对构件、连接的故障事件、故障概率等属性的建模;行为附件增强了 AADL 对构件实际功能行为的详细描述能力,以更好地支持功能行为验证和自动代码生成。

2.2.3 AADL 开发环境

前文已经提过,模型驱动工程中的一个重要课题就是软件开发过程的自动化。在目前关于基于 AADL 开发方法的支持工具研究中,已经有丰硕的成果。AADL 工具集涵盖了 MDE 中的建模、分析、代码生成的全过程。开源工具和商业工具皆有其一席之地。

- OSATE

OSATE(Open Source AADL Tool Environment)是由卡耐基梅隆大学软件工程研究所开发的一款 AADL 开源开发环境。本文设计开发的评估器也选择了 OSATE 作为 AADL 的模型编辑环境。它基于 Eclipse 平台,以插件形式实现,包含有建模工具、验证工具和分析工具。建模工具提供了文本编辑器、文本解析器、AADL 产生器、图形编辑器,支持 AADL 文本、XML 和图形三种视图的同步建立、更新与转换;其验证分析工具包括语法、语义检查器和多种分析器。目前提供的分析器有资源预算和分配分析(Resource Budget and Allocation Analysis)、端口连接检查(Required Connection Checking, Stream Miss Rate Checking)、安全级别检验(Security Level Checking, Safety Criticality Level Analysis)、流延迟分析(Flow Latency

Analysis)、优先级倒置检验(Priority Inversion Checker)和应用程序绑定与调度分析(Application Binding & Scheduling Analysis)。图 2.4 是 OSATE 开发环境的工作窗口截图。

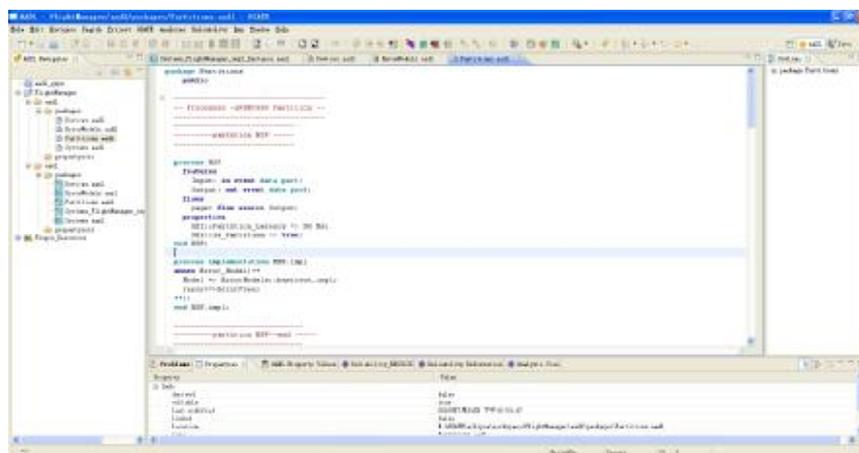


图 2.4 OSATE 窗口

● TOPCASED

TOPCASED(Toolkit in Open Source for Critical Applications & Systems Development)是由欧洲空中客车公司开发的另一个基于 Eclipse 平台的开发环境。TOPCASED 的特点是严格面向模型，对建模语言进行了深入描述，支持不同的建模语言，包括 Ecore、UML、SAM、AADL 和 SysML。它不仅提供了 OSATE 中的模型编辑器和模型检验器，还提供了不同类型的模型转换器。TOPCASED 项目还对自动化文档生成工具进行了研究，给用户提供了一种可定制的使用方式，用于创建新的文档模板。除此之外，TOPCASED 还提供了多个横向活动管理，包括异常管理、版本控制和需求追踪等。可以说，TOPCASED 是一个面向工程化实践应用，支持模型驱动工程所有活动的工具集。

● ADeS

ADeS (Architecture Description Simulation)是由 SEI 开发的一种仿真工具。它基于 Eclipse 平台,可单独使用,也可作为 OSATE 和 TOPCASED 的插件。AADL 的体系结构模型是在软件开发的过程中逐步被细化和精确化的,ADeS 既可以在开发初期对上层系统的全局行为进行仿真,也可以用于在开发后期对底层系统的精确行为进行评估。它首先解析出模型中关于行为的描述信息,然后据此仿真系统行为。

● OCARINA

Ocarina 是由法国 TELECOM ParisTech 采用 Ada 语言开发的一个 AADL 模型处理工具。Telecom Paris 将 AADL 作为分布式实时嵌入式系统开发的必要工具。为此, Ocarina 开发了自己的模型解析器,以支持 AADL 模型的各种操作。Ocarina

用一个基于 Dia 图像处理的模型转换器来支持对图形的处理。Ocarina 还提供了一个自动化生产工具，它基于 Ada Ravenscar Profile，针对高集成度分布式应用程序的运行时软件，能自动生成运行于 PolyORB 之上的 Ada2005 和 C 代码应用程序。

- Cheddar

Cheddar 是由法国 Brest 大学的 LISyC 团队开发的用于检验实时系统中的时间约束的工具，提供了基本的实时调度分析功能。Cheddar 并不局限于由 AADL 描述的系统体系结构模型，也接受其他模型作为输入。Cheddar 采用仿真方法，其调度仿真器由两个部分组成：a)图形编辑器，使用 GTK Ada 作为实现框架，用于描述实时应用程序或系统；b)框架，由多种经典的实时调度算法和可行性分析算法组成，使用灵活。既可以自动查找任务约束属性，通过计算理论进行调度分析，也可以不经过计算步骤，直接进行调度可行性测试。

- STOOD

STOOD 是由欧洲 Ellidiss 技术中心提供的一种基于构件的软件建模工具。它与 TOPCASED 一样，试图涵盖软件开发的各个阶段，而且达到了 DO178B 的要求。STOOD 支持已有多种代码的自动生成和重用，包括 C、Ada 和 C++。在模型驱动工具支持方面，STOOD 支持系统的早期性能分析。为重用已有的 STOOD 技术，STOOD 不仅提供了一个 AADL 图形编辑器，能够导入和导出 AADL 文本以及解析和产生模型，还支持 AADL、UML2.0、HOOD 三种模型之间的等价转换。

第三章 基于AADL的嵌入式软件可靠性建模评估框架

软件可靠性评估的目的是在对软件的可靠性因素加以考量的基础上，估计出目前软件系统的可靠性状态。为了能更清晰和更明确地反映系统的当前状态，一般采用定量评估的方法，得出一个量化指标。

对软件可靠性进行定量评估的最简便明确的方法是在软件开发完成并投入运行后，在真实的运行环境中检验软件失效的情况。但这种方法并不能满足人们的要求。它要求的时间周期长，而且评估的结果无法对后续的软件开发提供依据。目前，在开发阶段的软件可靠性评估方法是基于软件可靠性建模的方法，软件可靠性建模是软件可靠性评估的基础。本章对软件建模、评估框架以及工具支持进行阐述。

3.1 可靠性建模框架

在基于AADL的嵌入式软件开发中，体系结构模型是整个工程的核心，体系结构的验证、分析与AADL体系结构模型的建立同步展开。图3.1展现了使用AADL建立的体系结构模型的特点，通过一个模型，就可以表达系统的各种非功能属性。这就为从不同非功能属性角度对体系结构的设计决策进行验证与分析提供了平台，最大限度地确保设计满足用户的需求，减小后期修改的代价。

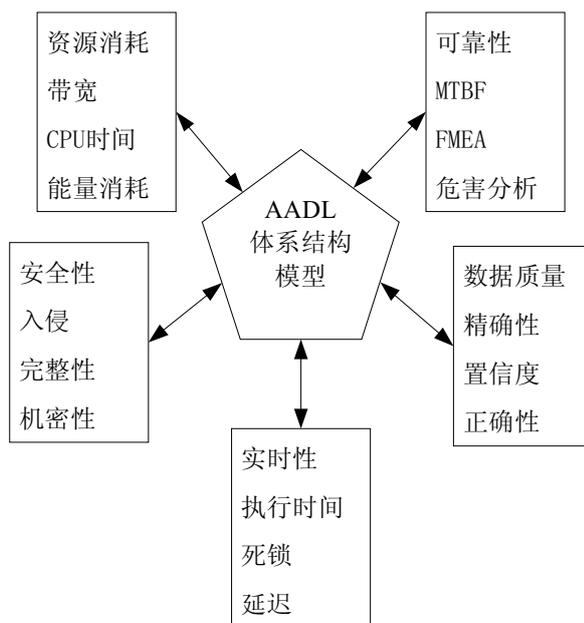


图 3.1 AADL 模型的非功能属性

在AADL标准中,部分简单的非功能属性(如实时性)是可以通过内建的或者自定义的构件属性来表达的;而复杂的非功能属性(如安全性)就需要AADL附件机制的支持才能表达。AADL错误模型附件提供了对系统的可靠性属性进行建模的支持。在本文中,为了更明确地表达AADL可靠性模型的建立方法,使用“AADL结构模型”来代表未添加可靠性信息的AADL体系结构模型。

图3.2是AADL可靠性模型的建立框架,AADL错误模型的作用是对构件的各种可靠性信息包括故障行为、故障传播等进行描述,然后添加到AADL结构模型中,与嵌入式软件中的各种构件相结合,即错误模型与相应的构件绑定,最终形成完整的嵌入式软件可靠性模型。该可靠性建模框架实现了故障行为建模和软件结构建模的分离,提高了可靠性模型的重用性。而错误模型与结构模型绑定的方式,使故障行为模型和结构模型成为了一个有机的整体,更易于理解系统的可靠性行为。

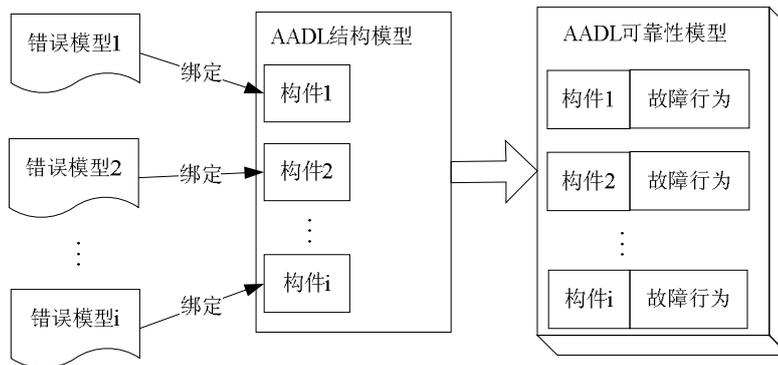


图 3.2 AADL 可靠性模型建立框架

3.2 可靠性评估框架

不同于前面介绍的软件可靠性模型,在本文中的AADL可靠性模型仅是对嵌入式软件的故障行为的刻画和各种可靠性特征的抽象描述。在AADL可靠性模型的基础上,还需要系统定量性能评价工具。

图3.3展示了基于AADL的可靠性评估框架。以AADL可靠性模型为基础,根据模型转换规则,生成广义随机Petri网(Generalized Stochastic Petri-Nets, GSPN)模型。不仅GSPN模型中的元素与AADL可靠性模型中的元素一一对应,不会造成模型转换后的信息缺失,而且为软件系统可靠性的定量评价提供了手段。基于GSPN理论,可以对软件系统目前的可靠性状态做出定量评价。评价结果与软件需求阶段建立的软件可靠性目标进行比较后,决定下一步的软件开发工作。如果可靠性评价结果满足可靠性目标,就进入下一阶段的软件开发工作。否则就需要进

行可靠性分析，找出影响系统可靠性的缺陷和关键环节，通过加入可靠性设计方案、调整体系结构来提高系统的可靠性。

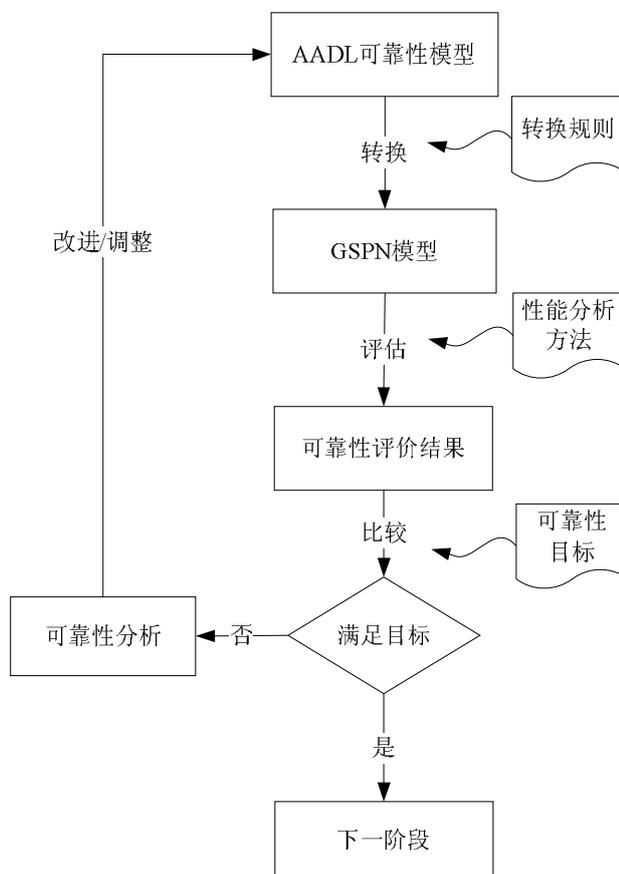


图 3.3 AADL 可靠性评估框架

3.3 可靠性评估工具

在 MDE 的思想中，解决软件开发复杂度问题的一个途径就是在软件的开发过程中提供部分的或全部的自动化支持。在基于 MDE 的软件开发过程中，对模型的建立、验证、分析提供相关工具的支持，可以减轻开发人员的工作量。

图 3.4 展示了基于本文的 AADL 可靠性评估方法的自动化评估工具的总体流程。首先通过模型编辑器生成符合 AADL 语法、语义规则的可靠性模型及其文件；之后通过模型转换工具生成对应的存储 GSPN 模型的文件；最后将 GSPN 模型输入到 GSPN 性能指标计算工具中，得到系统的可靠性评估和分析结果。图中虚线框内为本文的 AADL 可靠性评估器完成的工作。在工具的支持下，AADL 可靠性模型到 GSPN 模型的转换和相应性能指标的计算工作对用户透明化了，简化了软件可靠性的评估工作，提高了效率。

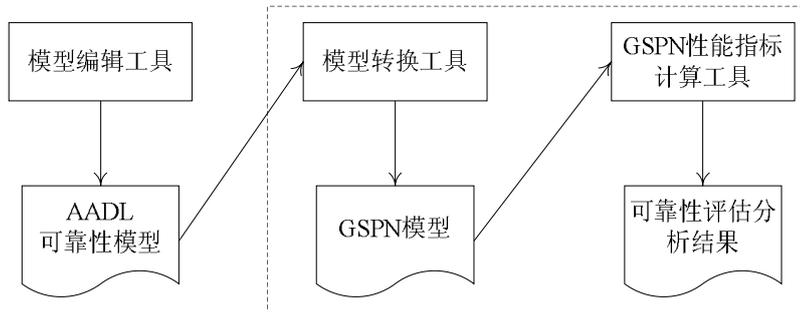


图 3.4 AADL 可靠性评估工具总体流程

在第二章中已对常见的AADL相关工具进行了介绍，OSATE由于其环境代码开源、文档丰富、平台开放等特性而被广泛使用。根据项目的需求和OSATE的优势，本文实现的AADL可靠性评估器亦选择OSATE作为模型编辑工具。图3.5以用户视角展示了本文的AADL可靠性评估器。评估器以插件形式实现，它的使用需要依赖于OSATE插件和错误模型附件插件。错误模型附件插件提供了生成以XML格式存储的错误模型文件的功能，方便模型的转换。用户要完成可靠性的评估和分析工作，只需要在Eclipse环境下进行操作就可以完成。

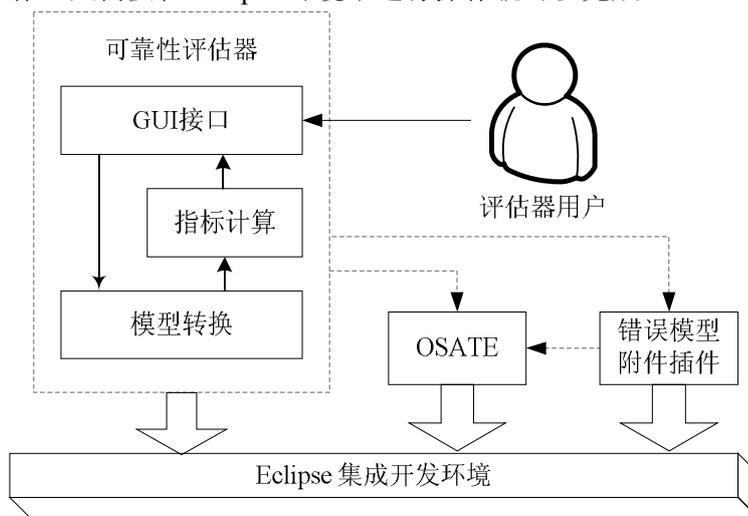


图 3.5 AADL 可靠性评估器的用户视角

第四章 基于AADL的嵌入式软件可靠性建模

第三章中已经给出了基于AADL的嵌入式软件的可靠性建模和评估框架。AADL可靠性模型是嵌入式软件可靠性评估、分析等工作的基础，它的目标是建立能够准确反映系统可靠性行为的抽象模型，并用形式化的语义进行描述。本章详细阐述结合AADL结构模型，使用AADL错误模型机制和语句建立嵌入式软件可靠性模型的具体方法。

4.1 错误模型建立框架

AADL错误模型提供了嵌入式软件构件的故障行为的描述信息，将它与AADL结构模型相结合就可以构成完整的AADL可靠性模型，最终实现软件可靠性行为的形式化描述。基于体系结构的软件可靠性建模方法以软件的内部结构为基础，错误模型的建立同样需要利用结构模型中提供的相关信息，才能描述软件系统的可靠性行为。

为降低庞大复杂系统的建模困难，应遵循AADL的迭代开发思想。首先根据AADL结构模型中的构件组成信息，选择合适的建模粒度，如果粒度过大则不能精确反映系统的可靠性行为，准确定位系统中的关键构件；如果粒度过小则会使建立的模型过于庞大，无法进行可靠性评估。可以在系统开发的早期选择较大的粒度如子系统等，而在后期可以选择较小的粒度如线程等。

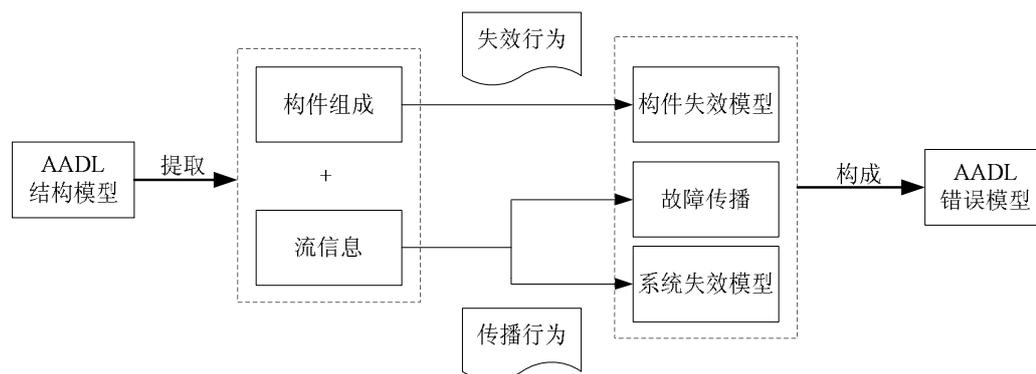


图 4.1 错误模型建立框架

图 4.1 给出了AADL错误模型建立的具体步骤。首先确定建模的粒度，为结构模型中同一层次上的所有构件建立单独构件失效模型，以反映单个构件在不受其他构件影响下的失效行为。

因为系统中的构件之间要进行相关信息的交互，所以相应构件之间存在着依

赖关系，一个构件的失效会影响其他构件。根据AADL结构模型中的流信息，可以确定构件之间的故障传播路径，利用错误模型中的机制在单独构件模型中添加错误传播描述，以此反映单个构件的失效行为对其他构件的影响。

由于故障传播的存在，不同构件的失效对系统的影响不同，在建模时需要考虑该因素。根据AADL结构模型中的流信息，可以确定系统信息的输出构件，以此建立系统的失效模型，反映不同构件的失效行为对系统的影响。

4.2 单独构件模型

单独构件的错误模型是AADL错误模型的基础。概括地讲，AADL错误模型使用随机状态机描述构件的故障行为。通过将故障行为随机状态机和系统中对应的构件绑定的方式，就可以描述每个构件自身的故障行为。

在具体实现上，单独构件的AADL错误模型和结构模型的实现相似，由错误模型类型和错误模型实现共同构成，这提高了错误模型的重用性。错误模型类型中声明了所有可能的故障状态和故障事件。其中故障事件的触发会导致故障状态的转移。错误模型实现中则定义了故障事件及其随机参数。

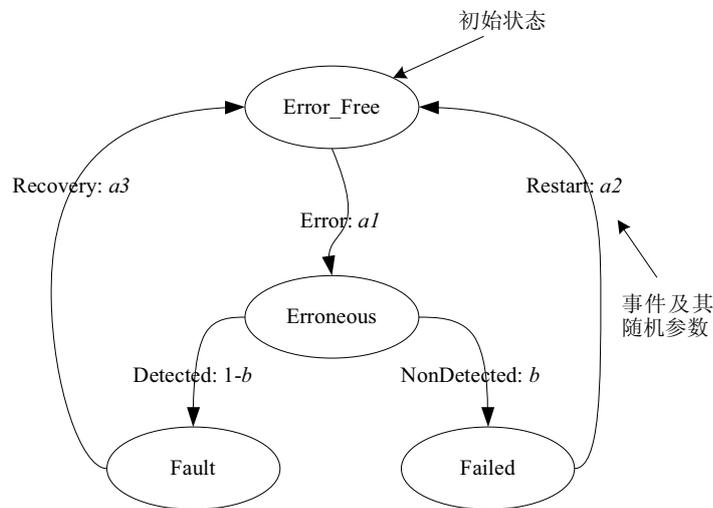


图 4.2 单独构件错误模型

在错误模型的随机状态机中，构件以初始状态开始，初始状态有且仅有一个，事件的到达会触发状态之间的转移。单独构件模型的建立应针对嵌入式软件如航空航天、武器装备、通讯领域内的软件具有故障恢复、容错能力的特点加入相应的状态转移。如图 4.2 所示是一个基本的错误模型，构件以 Error_Free 为初始状态，Error 事件的到达会触发从 Error_Free 状态到 Erroneous 状态的转移。而当构件处于 Erroneous 状态时，有可能会被系统侦测到，也有可能不会。如被侦测到，构件即

进入 Fault 状态并有可能因 Recovery 事件而恢复到 Error_Free 状态。当未被侦测时，进入 Failed 状态，此时需要 Restart 事件才能重启进入 Error_Free 状态。

在 AADL 错误模型中。事件到达的概率特性有两种描述。一种是发生概率，它以开关分布的形式表示一个事件发生的概率，参数取值为 0 到 1 的实数。另一种是到达速率，它表示故障事件的到达符合泊松过程，参数 λ 取正实数，表示平均到达速率。在上述例子中，Detected 和 NonDetected 事件的可能性则由 $1-b$ 和 b 的概率表达。Error、Restart 和 Recovery 事件到达的概率由参数为 $a1$ 、 $a2$ 和 $a3$ 的泊松分布描述。

下面给出了图 4.2 的 AADL 错误模型的形式化语句定义。feature 关键字表明以下是错误模型状态及事件的声明。transitions 关键字表明以下是事件引起的状态转移路径的声明。properties 关键字则表示了对事件随机参数的声明。形式化语句定义为 AADL 开发环境中的自动化可靠性工具的实现奠定了基础。

```
error model Example //错误模型类型
features //错误模型特性
ErrorFree: initial error state; //初始状态
Faulse, Failed: error state; //故障状态
Restart, Recovery: error event; //故障事件
end Example; //类型声明结束

error model implementation Example.impl // 错误模型实现
transitions //状态转移声明
ErrorFree-[Fail]->Fault;
Fault-[Detected]-> Faulse;
Fault-[NonDetected]-> Failed;
Failed-[ Restart]->ErrorFree;
Faulse-[ Recovery]->ErrorFree;
Properties //故障事件的随机属性声明
Occurrence => poisson a1 applies to Error;
Occurrence => fixed b applies to NonDetected;
Occurrence => fixed 1-b applies to Detected;
Occurrence => poisson a2 applies to Restart;
Occurrence => poisson a3 applies to Recovery;
end Example.impl; //实现声明结束
```

4.3 故障传播

4.3.1 故障传播的过程

为降低复杂的大规模软件的开发难度，现代软件一般采用模块化设计，软件系统可以看作是若干个模块相互作用而成，它们通过参数传递、内存共享、消息传递等方式相互协作，共同影响系统功能的完成。在现在的软件体系结构中，构件的封装度越来越高，重用度越来越大，对运行在同一环境中的不同构件发生的故障对其他构件的影响进行研究有着现实的价值。在体系结构层面，软件故障传播研究的内容就是不同构件在故障状态下是如何影响其他构件的，哪一个构件是最容易受到其他构件的影响。特别是在当今的大型嵌入式软件领域中，构件的交互影响异常频繁，在可靠性建模时必须考虑构件之间的故障传播因素。

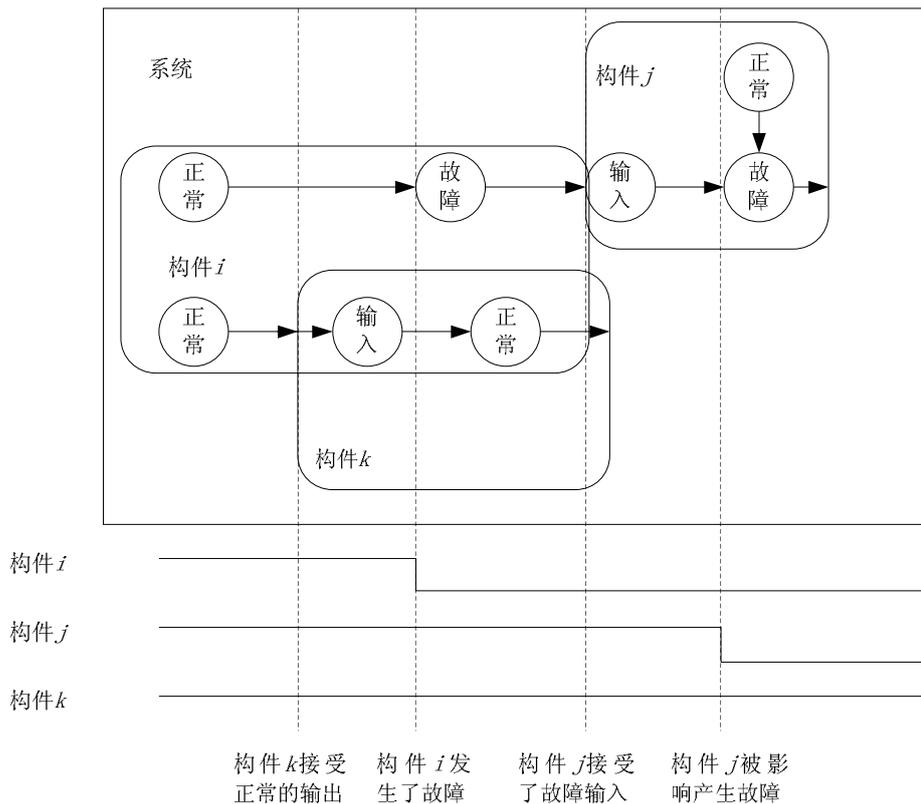


图 4.3 故障传播过程

图 4.3 是系统内的一次故障传播过程。其中构件 i 有两个接口分别连接构件 j 和构件 k 。当构件 i 正常时，构件 k 接受了构件 i 的输入，因此，构件 k 未受到影响；构件 i 随后发生了故障，而构件 j 接受了构件 i 故障之后的错误输入，最终导

入构件 j 也发生了故障。因此可以总结故障传播的过程如下，首先在一个构件内产生了故障，这个构件称为源构件；当故障未能被隔离，从源构件内传出，一个或多个构件接受了故障输入；一个或多个构件被影响而发生故障，称为目标构件。

4.3.2 错误模型的故障传播描述

在故障传播过程中，故障是否会从源构件中传出，有其随机性。例如当源构件发生故障后产生了错误的的数据并输出到了共享数据区，但在目标构件接受错误的的数据之前，源构件在共享数据区内更新了正确的数据，则故障未传入到目标构件中。本文定义故障传播概率如下：

定义 4.1 $AEP[A(s), B]$ 表示系统中构件 B 所处的状态受到构件 A 处于故障状态 s 时影响的概率。即构件 A 进入状态 s 时产生的故障传播到构件 B 的概率。

该定义是与 AADL 错误模型中对于故障传播的描述方式相结合的， $AEP[A(s), B]$ 是一个条件概率，即在构件 A 已经发生故障而处于故障状态 s 的条件下，影响构件 B 所处状态的概率。

在 AADL 错误模型中对于上述故障传播过程的描述是在单独构件模型的基础上，加入故障传播事件的概念来描述故障传播过程。故障传播事件分为传出故障传播事件和传入故障传播事件两种，其中传出故障传播事件具有随机参数，即故障传播概率。在源构件的单独错误模型中，需要定义可以产生传出故障传播事件的故障状态，当源构件处于这种故障状态时，就可能触发传出故障传播事件的产生。这相当于描述了故障传播过程的第一步和第二步。故障传播过程的第三步通过传入故障传播事件和名字匹配机制来描述。当产生传出故障传播事件时，该事件就会直接影响一个或多个目标构件，使目标构件转移至故障状态。其中目标构件的识别通过名字匹配实现，同名的传出、传入故障传播事件表示了故障传播的起点和终点。

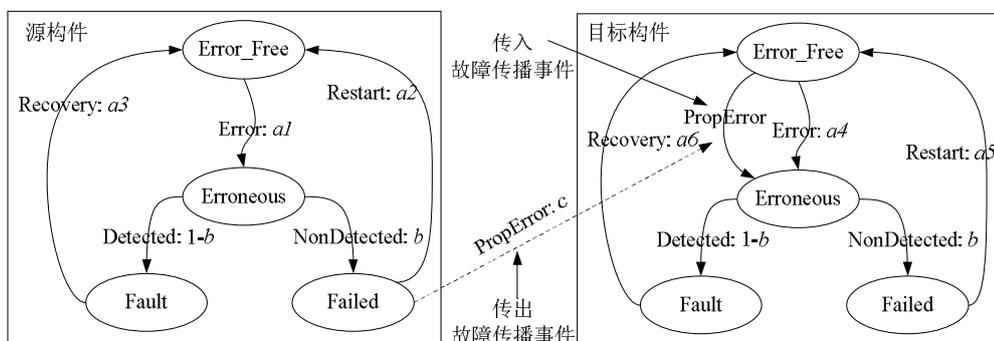


图 4.4 故障传播的 AADL 错误模型描述

源构件的单独错误模型、故障传播事件和目标构件的单独错误模型共同构成了 AADL 故障传播模型。图 4.4 给出了一个 AADL 故障传播模型的例子，当源构

件处于 Failed 状态时，就会以 c 的概率产生名为 PropError 的传出故障传播事件。而目标构件中也存在一个同名的传入故障传播事件。一旦源构件产生了 PropError 传出故障传播事件，且目标构件处于 Error_Free 状态，则目标构件被影响进入 Erroneous 故障状态。这里对故障传播概率的表达仅在传出故障传播事件中，而不在传入故障传播事件中，所以一旦产生了传出故障传播事件，就会直接导致目标构件进入故障状态。

下面是图 4.4 中错误模型的形式化描述的片段。在 AADL 错误模型中，通过传入和传出故障传播事件的名字匹配来判断源构件和目标构件。在示例中，PropError 事件是 Source 中的传出故障传播事件。而在 Target 中声明了一个同名的传入故障传播事件，只要 Source 进入 Failed 状态，就会以 c 的概率产生 PropError 事件，而一旦 PropError 事件产生，且 Target 处于 ErrorFree 状态，就会直接转移至 Fault 状态。

```

error model Source //源构件的错误模型类型声明
features //特性声明
PropError: out error propagation; //传出故障传播事件
end Source;
error model implementation Source.impl //源构件的错误模型实现声明
transitions
Failed-[out PropError]->Failed; //触发传出故障传播事件
end Source.impl;
error model Target //目标构件的错误模型类型声明
features //特性声明
PropError: in error propagation; //传入故障传播事件
end Target;
error model implementation Target.impl //目标构件的错误模型实现声明
transitions
ErrorFree-[in Error]->Fault; //传入故障传播事件引起的状态转移
end Target.impl;

```

4.3.3 故障传播路径

在体系结构层面，代码和语法信息均不可见，只有模块内和模块之间的数据流和控制流是可以获得的^[36]，通过这些信息可以确定故障传播的路径。在 AADL

体系结构模型中，流声明是对流经系统中所有构件的抽象信息的详细描述，以提供对信息流的分析能力。如流延迟分析是嵌入式软件调度性分析中的一个重要方面，必须在设计阶段就进行描述。AADL 流信息分为两种，数据流信息和控制流信息，分别反映了系统中不同构件之间的数据的交互关系和控制关系。由于嵌入式系统内构件交互的实质是数据的交互，同时系统或子系统输出的正确性以输出的数据为依据，因此本文选取 AADL 数据流作为确定故障传播路径的依据，而忽略构件之间的控制流的影响。

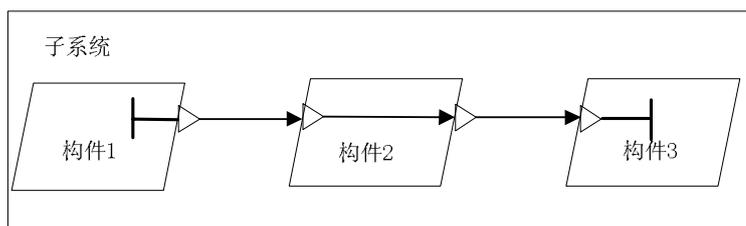


图 4.5 AADL 流声明

图 4.5 是 AADL 结构模型中一条完整的流声明，表明了数据如何在该子系统内的各个构件之间流动。其中构件 1 中的 T 型线表示该构件是流路径的起点，而构件 3 中相反的 T 型线表示了该构件为终点。整条流声明表示了数据从构件 1 开始并流经构件 2，最终在构件 3 终止。据此可以确定构件之间的故障传播路径为从构件 1 中的故障只会传播到构件 2，而构件 2 中的故障只会传播到构件 3。

4.4 系统失效模型

嵌入式软件系统是由多个相对独立的构件和子系统自上而下组成的，系统的失效是由所有构件或子系统所处状态共同决定的。本文在第二章中对软件的失效机理进行了阐述，即人为引入的软件错误最终导致了软件失效的发生。但是，不是所有的软件错误均会导致软件失效。同理，不是每个构件的失效都会导致整个子系统或系统发生失效。传统的可靠性理论一般认为任何一个构件的失效均会导致整个系统发生失效，这种观点未考虑不同构件以及故障传播的影响，仅关注了单个构件的故障行为，导致高估了系统的失效率。

在软件的失效机理中，仅被外界感知的故障或者说将故障输出到了外界才称为软件失效。在系统中，只有与外界交互的构件发生失效时，外界才会感知系统的失效状态。如图 4.6 所示，构件 3 的故障输出到外界时，以该单独构件为界限，就认为是构件失效了，但这不代表系统发生了失效，仅当与外界交互的接口构件 4 发生了故障并输出时，才认为系统发生失效。

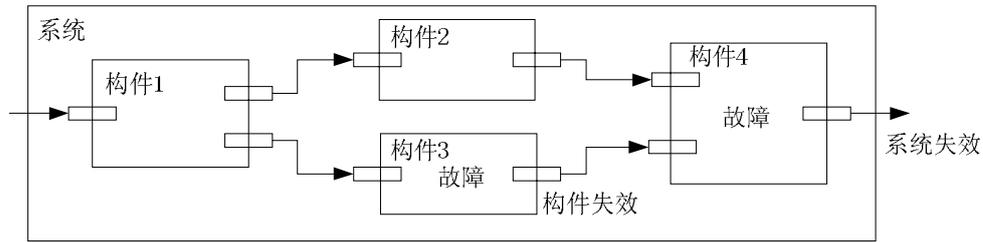


图 4.6 系统失效模型

在嵌入式软件中，不是子系统内所有的构件处理的数据都会传送到其他子系统，仅有一个或几个构件的数据会传送出去。如在图 4.5 中，从子系统内部看，构件 3 是数据传输的终点，但是一个子系统不会是封闭的，所以构件 3 的数据最终会传输给其他子系统或者用户。本文提出，以 AADL 结构模型中的 AADL 流终点构件代表系统中的交互构件，作为系统失效的判断依据。

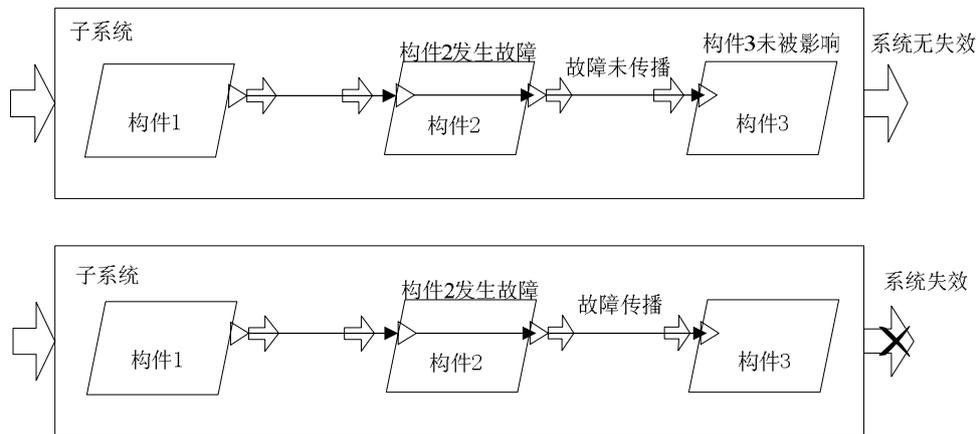


图 4.7 故障传播对系统失效的影响

在本文的系统失效模型中，考虑了故障传播因素，能更精确地对当前系统的可靠性状态进行评估。如图 4.7 所示，虽然构件 2 发生了故障，但因其故障传播的随机性，仅当故障传播到了接口构件--构件 3 时，子系统才发生失效，否则认为子系统未发生失效。

基于此，根据 AADL 结构模型中单独构件错误模型、AADL 流信息确定的故障传播路径、接口构件共同组成系统的 AADL 嵌入式软件可靠性模型。图 4.8 所示是由三个构件组成的子系统的 AADL 可靠性模型。其中构件 3 的 Failed 状态即为系统的失效状态。

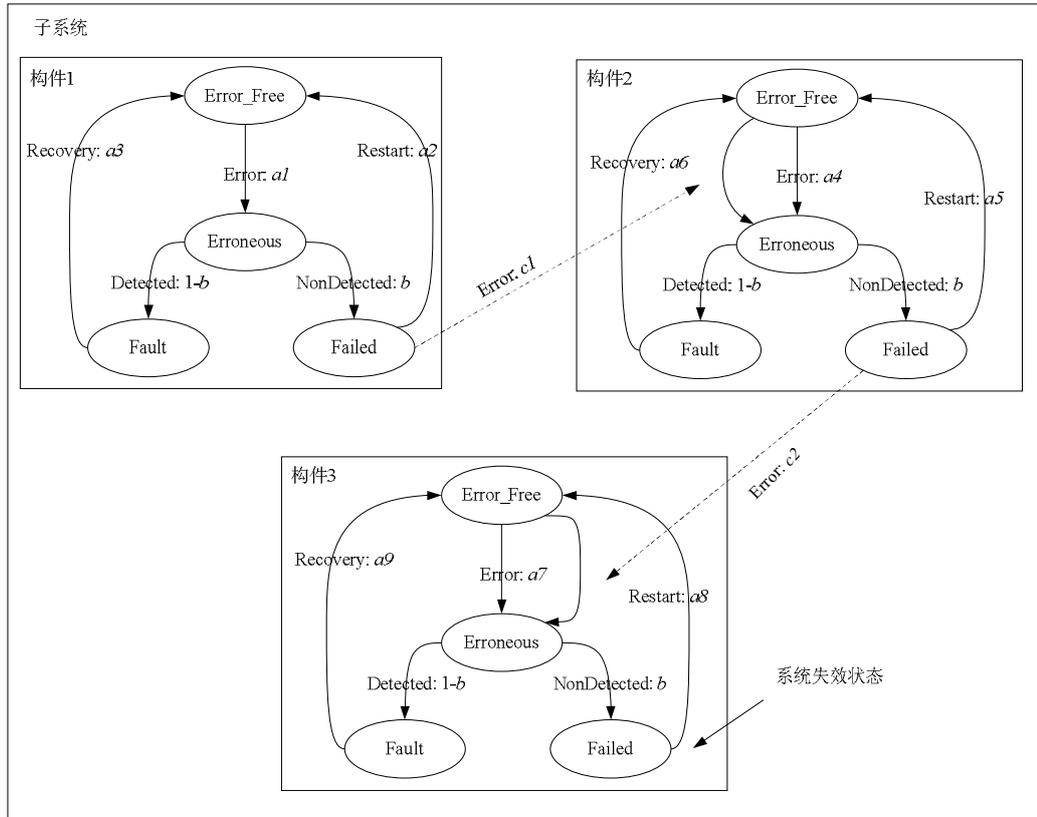


图 4.8 AADL 可靠性模型

第五章 基于 AADL 的嵌入式软件可靠性评估

上一章阐述了使用 AADL 建立嵌入式软件可靠性模型的方法。但是 AADL 可靠性模型仅仅是定性和定量地描述了软件的故障行为，得出软件可靠性定量评估指标，还需要数学工具的支持。

Petri 网是 1962 年由西德波恩大学的 C.A.Petri 在他的博士论文中首次提出的，论文中定义了一种通用的数学模型，可以对条件与事件间的关系进行描述。之后经过不断的发展和完善，Petri 网成为了一种重要的动态系统建模工具，其应用研究主要集中在通信协议、计算机组织结构以及计算机软硬件分析方面。

随机 Petri 网为系统性能模型的建立提供了良好的描述手段；而其状态空间又与随机马尔可夫链同构，为性能评价提供了坚实的数学基础^[37]。因此将随机 Petri 网应用于软件性能度量可以取得良好的效果。

5.1 广义随机 Petri 网

5.1.1 Petri 网模型

一个 Petri 网模型包含库所、变迁、有向弧和托肯四种元素。库所用圆圈表示，可以容纳托肯，并可以设置容纳数量的上限。托肯代表了系统中的某种资源的数量，可以在不同库所之间流动，用黑点表示。变迁用矩形表示，它有不可实施、可实施和实施三种状态。当变迁的输入库所中至少含有一个托肯时，成为可实施状态；当变迁满足实施规则表示的条件时，进入实施状态，导致变迁所连接的输入库所到输出库所的托肯发生流动。有向弧用来连接库所和变迁，表示了托肯流动的方向，它划分了输入库所、变迁和输出库所。

一个 Petri 网系统是指带有初始标识的 Petri 网模型。所谓标识是指托肯在 Petri 网中的不同分布状况，初始标识代表了系统的初始状态。在 Petri 网系统中，以初始标识为起点，一旦满足变迁的实施条件，变迁的实施会使托肯沿着弧在库所之间流动。托肯的流动导致了标识的改变，以此描述了系统在一定条件下的动态变化过程。

图 5.1 是一个基本的 Petri 网系统，包含有三个库所和四个变迁，初始标识是库所 P0 和 P3 中各含有一个托肯。在处于初始标识时，变迁 T2 处于可实施状态。T2 实施后，P0 和 P3 中的托肯会同时流动到库所 P2 中。变迁 T3 实施后，又会流

回到 P0 和 P3 中。当变迁 T1 实施时，托肯从 P0 流动到 P1，而变迁 T0 的实施会导致托肯又流回到库所 P0 中。

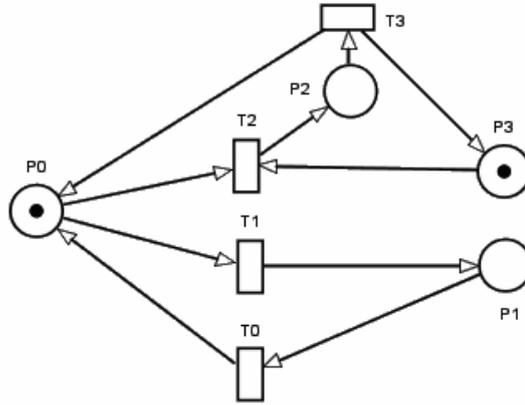


图 5.1 Petri 网系统示例

在实际建模中，有向弧连接库所和变迁的不同方式描述了系统的组织结构；标识描述了系统在不断的动态变化中目前所处的状态；变迁代表系统中真实的事件，变迁的实施代表事件对系统的状态的改变。

5.1.2 GSPN

在早期的 Petri 网研究工作中，时间概念并未被考虑，因为时间参数可能破坏 Petri 网结构能够表示真实系统中所有可能的行为的构想。但是，在许多应用领域，特别是系统性能的定量分析中，需要时间的概念，而相关的数学工具却不尽如人意。于是时间参数被加入到 Petri 网系统中，扩展了 Petri 网的应用范围。

在每个变迁的可实施与实施之间联系一个随机的延迟时间，即一个变迁 T 从变成可实施的时刻到它的实施时刻之间被看成是一个连续随机变量 X_t (取正实数)，且服从一个分布函数

$$F_t(x) = P\{x_t \leq x\} \quad \text{式 5-1}$$

如果所有变迁的分布函数都定义成指数分布函数，即

$$\forall t \in T : F_t = 1 - e^{-\lambda_t x} \quad \text{式 5-2}$$

其中变量 $x \geq 0$ ，参数 $\lambda_t > 0$ 是变迁 T 的平均实施速率。这样的变迁称为随机变迁，拥有这样变迁的 Petri 网称为随机 Petri 网 (Stochastic Petri-Nets, SPN)。当 SPN 中的指数时间变迁从可实施的变为不可实施的，再一次变为可实施的情况下，如果重置其延迟时间为初值，则称随机变迁所联系的时钟是无记忆的。亦即，每当随机变迁所联系的时钟启动时，它总是设置为初值。已经证明，这样的随机 Petri 网的状态空间同构于一个齐次马尔可夫链。

在实际建模中， λ_t 表示在可实施的情况下单位时间内的平均实施次数，它的

值是根据某种要求从对所建模系统实际测量中获得的，有其实际的物理意义的。平均实施速率的倒数 $\tau_i = 1/\lambda_i$ 称为变迁 T 的平均实施延时。

广义随机 Petri 网(Generalized Stochastic Petri-Nets, GSPN)是将 SPN 中的变迁扩充为瞬时变迁和指数时间变迁两种，从而扩展了 SPN 的应用范围。其中指数时间变迁就是上述的随机变迁。瞬时变迁没有时间延迟，一旦处于可实施条件便立即实施。定义 5.1 是 GSPN 的形式化定义。

定义 5.1 一个 GSPN = $(P, T; I, O, H, \Pi, W, M_0, \lambda)$ ，其中

- P 是库所的集合， $P=(p_1, p_2, \dots, p_m)$
- T 是变迁的集合，包括瞬时变迁和指数时间变迁。 $T=(t_1, t_2, \dots, t_m)=T_{im} \cup T_{exp}$
- I 是输入弧集合 $I \subseteq P \times T$
- O 是输出弧集合 $O \subseteq T \times P$
- H 是禁止弧集合 $H \subseteq P \times T$
- Π 是瞬时变迁的优先实施函数
- M_0 是初始标识
- λ 是瞬时变迁的实施速率
- W 是瞬时变迁的权值函数

其中瞬时变迁的优先实施函数决定了当多个相同权值的瞬时变迁同时是可实施的时，哪一个变迁会被实施。而两个或多个瞬时变迁发生冲突时，权值函数代表了一个开关分布，决定每一个变迁会被实施的概率。

在 GSPN 中引入的禁止弧仅存在于库所到变迁之间。它连接变迁后，变迁的原可实施条件会变为不可实施条件，原不可实施条件会变为可实施条件，且在相连的变迁实施时，没有托肯从相连的库所中移出。

5.2 GSPN 可靠性模型

5.2.1 建模方法

在本文的 GSPN 可靠性建模中，每一个库所都代表一种故障状态，托肯的流动代表了构件状态的变化，容纳托肯的库所代表目前所处的状态。而不同类型的变迁代表了状态之间的转移时间符合哪一种随机分布。为了能与 AADL 可靠性模型对应，以下从单独构件模型、故障传播模型两方面对 GSPN 可靠性模型进行阐述。

- 单独构件 GSPN 模型

在单独构件 GSPN 可靠性模型中，库所代表状态，托肯代表构件所处的状态。

因为每个构件在一个时刻内只会处于一种状态，所以在单独构件 GSPN 模型中只存在一个托肯。图 5.2 是一个 GSPN 可靠性模型。其中有 4 个库所，分别代表 4 种状态，Error 变迁是指数时间变迁，用空心矩形表示，代表了构件的故障率符合指数分布。Detected 变迁和 NonDetected 变迁是瞬时变迁，用实心矩形表示，它们之间的开关分布代表了故障可被检测到的概率。变迁的实施速率是由系统中实际测量的结果得出的。

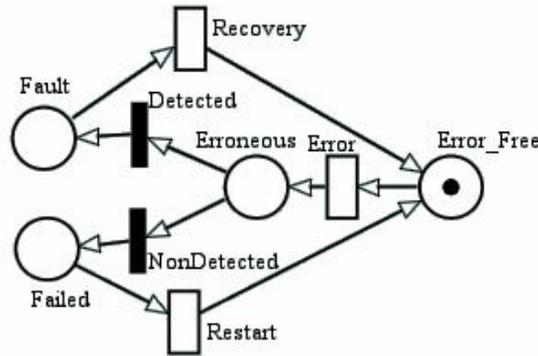


图 5.2 单独构件 GSPN 可靠性模型

● 故障传播 GSPN 模型

对于故障传播的建模，要符合故障传播的发生过程。当构件进入失效状态，使用两个冲突的瞬时变迁对故障传播的随机性进行刻画，两个变迁的开关分布代表了其发生概率。图 5.3 是故障传出的 GSPN 模型。假设当构件处于故障状态时，即当托肯在 Failed 库所内，有传出故障的可能。两个瞬时变迁 Prop 和 NoProp 表达了故障传出的概率。Prop 变迁的实施表示构件的故障已传出，而 NoProp 变迁的实施表示故障未传出。当变迁 Prop 实施后，库所 Failed 和 PropPlace 内皆含有托肯，表明构件处于可以传出故障的状态。当构件离开 Failed 状态后，变迁 Eliminate0 和 Eliminate1 可以清除 PropPlace 和 NoPropPlace 中的托肯。使模型回到不能传出故障的状态。

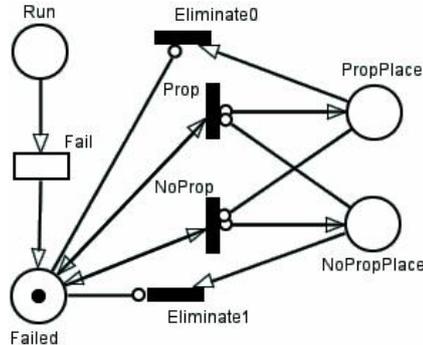


图 5.3 故障传出 GSPN 模型

图 5.4 是故障传入的 GSPN 模型，当源构件传出了故障后，且目标构件处于正

常的运行状态，即 PropPlace 库所和 Run2 库所中都含有托肯时，InProp 瞬时变迁满足可实施条件而立即实施，目标构件即从正常运行状态进入失效状态。这就对源构件传出故障并导致目标构件故障的过程进行了建模。

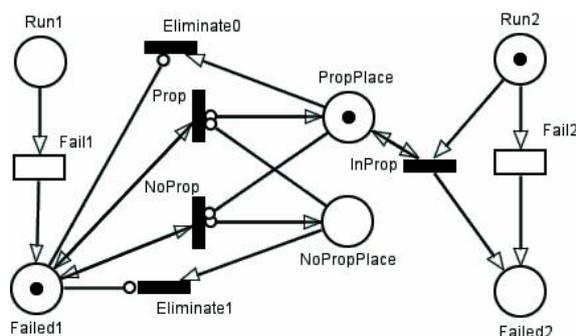


图 5.4 故障传入 GSPN 模型

使用上述方法对嵌入式软件中的构件，构件之间的故障传播建立的 GSPN 模型，能够反映软件系统中的各种故障状态及其状态变化，为嵌入式软件进行可靠性定量评估奠定了基础。

5.2.2 可靠性评估原理

在软件可靠性理论中，认为软件构件的寿命符合指数分布^[17]，所以软件系统的运行过程符合马尔可夫随机过程。根据马尔可夫链的遍历性，经过有限步的状态转移后，各个状态转移的概率会接近一个极限概率，称为稳态概率。如果将软件运行状态和故障状态以及它们之间的转移抽象为马尔可夫链，当系统经过长期稳定的运行后，即状态的有限步转移后，可以求出系统进入失效状态的稳定概率，这个概率是对系统长期运行后发生失效的概率的估测。前文已经介绍过，GSPN 的状态空间和马尔可夫链同构，即通过 GSPN 建立的软件可靠性模型可以转化为同构的马尔可夫链，继而求出稳态概率，得出对软件系统的可靠性评价。

在 GSPN 理论中，稳态概率的获得是性能指标分析中重要的一点。所以对系统的可靠性状态的定量评估与 GSPN 模型的性能指标分析方法一致。它的分析方法大体分为以下四个过程^[36]。首先建立系统的 GSPN 模型，在本文中即 GSPN 可靠性模型；其次根据模型穷举出所有可能的状态及状态转移，据此构造出与 GSPN 状态空间对应的可达图；然后根据可达图构造出与 GSPN 同构的马尔可夫链；最后根据马尔可夫随机过程理论，计算链中代表系统或构件的失效状态的稳态概率，获得对系统可靠性的定量评估结果。

5.3 AADL 可靠性模型到 GSPN 可靠性模型的转换

在第三章对评估框架已经进行了阐述，AADL 可靠性模型对系统的可靠性行为进行了描述，而 GSPN 可靠性模型为定量评估提供了手段，只需要将前者转换到后者，就实现了嵌入式软件的可靠性评估。

5.3.1 单独构件的模型转换

表 5.1 是单独构件的 AADL 错误模型到 GSPN 模型的转换规则^[30]，每一个错误模型中的元素都对应 GSPN 中的一个元素。AADL 错误模型中的状态，对应 GSPN 模型中的库所。错误模型中的初始状态决定了 GSPN 的初始标识；错误模型中事件的发生会导致故障状态的转移，所以事件对应 GSPN 中的变迁。变迁的类型根据事件的随机参数类型决定。因为错误模型中的到达速率是指参数为 a 的指数分布，所以对应 GSPN 中的实施速率为 a 的随机时间变迁。而发生概率对应瞬时变迁中的开关分布。一次状态的转移就对应相应输入库所到变迁，变迁到输出库所的弧。

表 5.1 错误模型到 GSPN 模型的转换规则

AADL 错误模型	GSPN
状态	库所
初始状态	初始标识
事件	变迁
随机参数-到达速率 (a)	时间变迁-参数 a
随机参数-发生概率 (b)	瞬时变迁-开关分布
状态转移 (Src_State-[event]->Dest_State)	从库所 Src_State 到变迁 event 的弧和从变迁 event 到库所 Dest_State 的弧

图 5.5 中展示了根据表 5.1 中的转换规则将单独构件的错误模型转换到 GSPN 模型的结果。ErrorFree、Erroneous 等故障状态都转换为 GSPN 模型中的一个同名库所。Recovery 等以到达速率为随机参数的故障事件，转换为同名的指数时间变迁。而以发生概率为随机参数的故障事件，分别转换为一组同名的瞬时变迁。变迁的权值是故障事件的随机参数，一组变迁代表了一个开关分布状态。

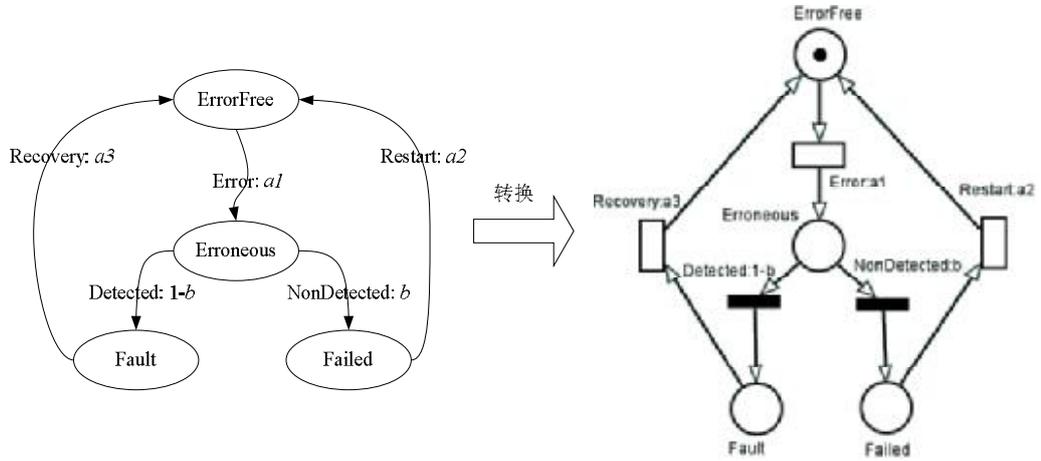


图 5.5 单独构件的转换

5.3.2 故障传播的转换

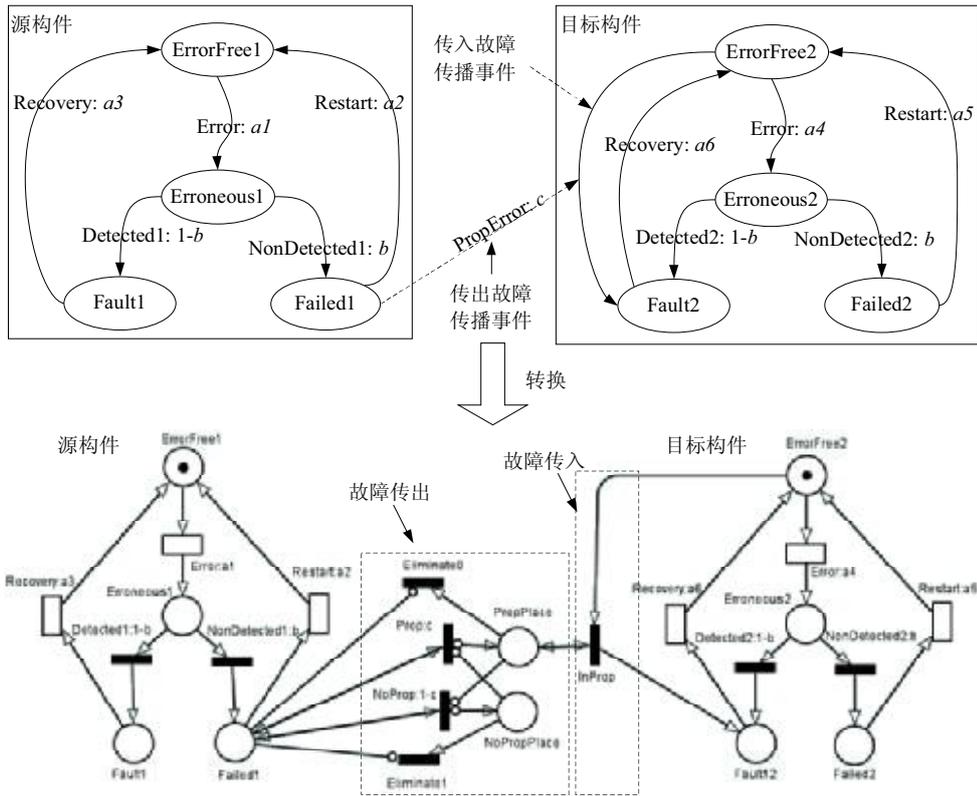


图 5.6 故障传播的转换

故障传播的转换是在源构件和目标构件的 GSPN 模型之间加入故障传播 GSPN 模型，以反映AADL 错误模型中描述的故障传播过程。图 5.6 是一个故障传播转换的示例。转换的过程是首选对源构件的故障传出进行转换，将产生传出故

障传播事件的故障状态 Failed1 与故障传出的相关变迁通过连接弧、抑制弧连接起来；然后将目标构件中受故障传播事件影响的状态转移和代表故障传出的库所 PropPlace 连接起来，完成故障传入的 GSPN 模型转换。

5.4 可靠性评估器的设计与实现

为减轻开发人员的负担，提高开发效率，软件工具的支持是软件开发中重要的组成部分。在模型驱动工程中，模型的形式化语义描述，为模型的自动化验证和分析奠定了基础。AADL 可靠性评估器是对本文嵌入式软件可靠性建模评估方法的实现，它提供了自动化的评估功能和基本的分析功能，能有效降低可靠性评估工作的复杂性。

5.4.1 体系结构设计

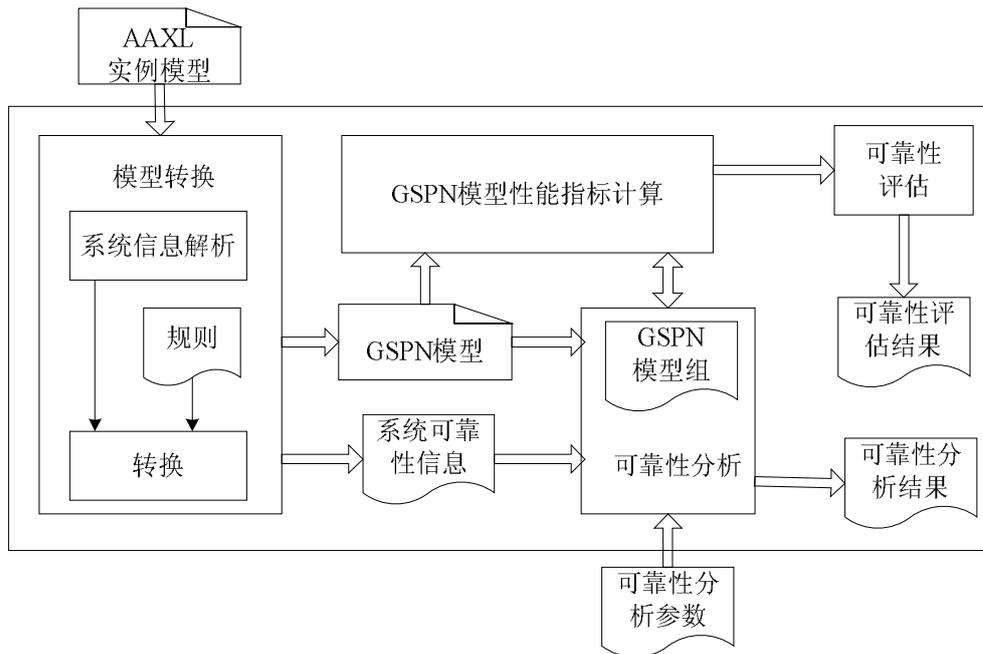


图 5.7 可靠性评估器体系结构

图 5.7 是 AADL 软件可靠性评估器的体系结构。从总体上看，评估器分为模型转换、GSPN 模型性能指标计算、可靠性评估、可靠性分析几个部分。评估器以 AAXL 文件作为输入。AAXL 实例模型是 AADL 模型的 XML 视图，它将 AADL 模型中的系统层次结构、系统的构件组成，构件的详细信息等内容以 XML 格式存储在一个文件中，方便了对模型的自动化验证和分析工作。在 OSATE 环境中，提供了生成 AAXL 实例模型文件的功能。

当进行可靠性评估时，首先对 AAXL 实例模型文件进行解析，输出系统的可靠性信息。同时，使用 5.3 节中的转换规则，将得到的可靠性模型进行转换，生成相应的 GSPN 模型。在对生成的 GSPN 模型进行分析后，得出目标状态的性能指标评价，即可靠性评估结果。当嵌入式软件可靠性评估结果不能满足需求阶段建立的可靠性目标时，需要进行可靠性分析，定位影响系统可靠性的关键构件。当进行可靠性分析时，首先要输入可靠性分析参数，根据参数，在原有 GSPN 模型的基础上改变变迁的实施速率，得到一组新的 GSPN 模型。最后对 GSPN 模型组逐个进行性能指标计算后，得出分析结果，并以友好的形式展现给用户。

5.4.2 评估自动化

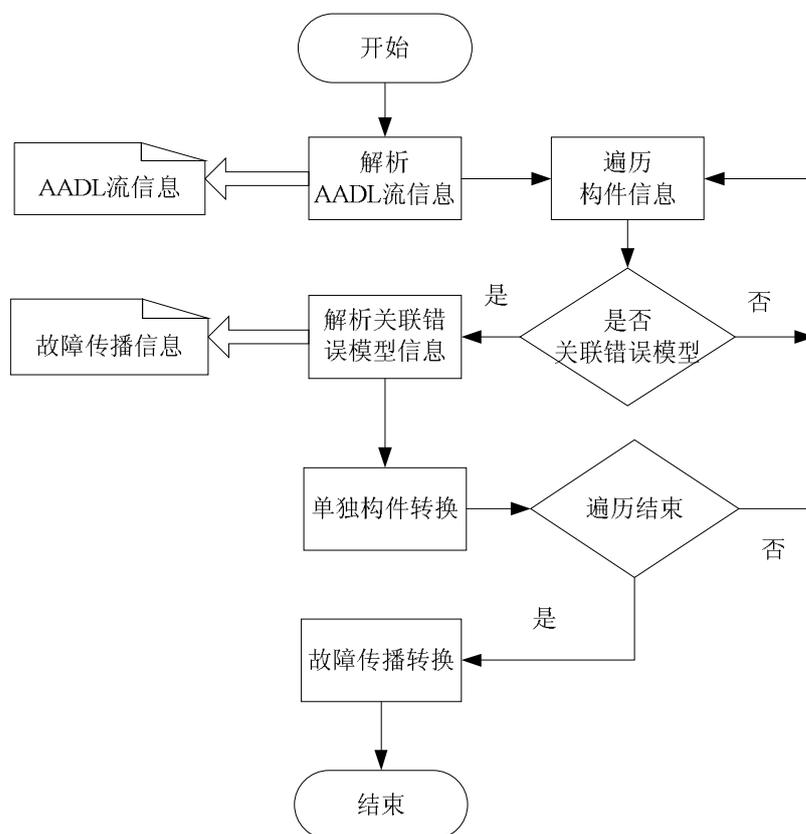


图 5.8 模型自动化转换流程

前面已经阐述 AADL 可靠性模型的定量评估以 GSPN 模型为基础，所以实现评估自动化最关键的步骤是根据转换规则将 AADL 可靠性模型自动转换为 GSPN 模型。模型的自动化转换流程大体上分为两个阶段，第一个阶段是可靠性信息的解析和单独构件模型的转换，第二个阶段是对故障传播的转换。具体的转换流程

如图 5.8 所示，首先从模型中解析出 AADL 流信息，为之后的故障传播转换服务。之后遍历模型中的构件，解析构件信息，检查其是否关联有错误模型，如果关联就获取相关联的错误模型信息，解析出其故障状态和故障事件。如果是故障状态或非故障传播事件，就采取边解析边转换的策略，逐步转换出单独构件的 GSPN 模型。如果属于故障传播事件，对传播事件引起的状态转移进行记录。当所有的构件都遍历完毕后，根据 AADL 流信息和记录的故障传播信息判断换构件之间是否存在故障传播，进行故障传播的转换。综合 AADL 流信息和故障传播事件的名字匹配机制共同判断构件间的故障传播，可以提高 AADL 错误模型的重用度和开发人员的工作效率。

算法 5.1 是单独构件模型的转换算法，它的输入是错误模型中的故障状态或故障事件。输出是 GSPN 模型。对输入的元素首先判断属于哪一种状态或事件，据此决定转换的 GSPN 元素类型。如果属于错误传播类型，需要记录相应的信息。

```

Begin
  Do
    If (errItem is Error State) //元素是故障状态
      GeneratePlace(); //生成 GSPN 库所
    Else If (errItem is Error Event) //元素是非故障传播事件
      GenerateTransition(); //生成 GSPN 变迁
    Else If (errItem is ErrorTransition) //元素是状态转移
      GenerateArc(); //生成库所变迁的连接弧
    Else If (errItem is OutProp) //元素是故障传出
      WriteSourceID(); //记录源构件 ID
      WriteSourceEvent(); //记录传出事件
    Else If (errItem is InProp) //元素是故障传入
      If(MatchName()); //查询同名故障传出事件
      WriteTargetID(); //记录目标构件 ID
      WriteTransition(); //记录目标构件状态转移
  End

```

算法 5.1 单独构件转换算法

算法 5.2 是对故障传播转换算法的描述。它以 AADL 流信息为基础，如果构件之间存在流，就去故障传播信息中查询两个构件之间是否存在故障传播的建模信息，如果存在，按照规则进行故障传播转换。

```
Begin
  Do
    If (Comp is Source) //构件是源构件
      GenerateOutTransition(); //生成 OutError 的几个变迁
      GenerateOutPlace(); //生成 OutError 的几个库所
      GenerateOutArc(); //生成 OutError 的连接弧和抑制弧
    Else If (Comp is Target) //构件是目标构件
      GenerateInTransition(); //生成 InError 的变迁
      GenerateInArc(); //生成 InError 的连接弧
  End
```

算法 5.2 故障传播转换算法

在 5.4 节中已经对 GSPN 的可靠性评估原理进行了阐述,可靠性评估结果实质是分析 GSPN 模型得出的性能指标--稳态概率。在自动化转换中,对哪一个状态进行指标分析是通过 AADL 可靠性模型中的 Report 关键字表明的,需要用户在建模时标注。

由于已经存在大量功能强大且实用的随机 Petri 网模型分析平台,本文没有从头设计一个 GSPN 工具,而是选择了 TimeNet^[38]工具作为 GSPN 模型的性能指标分析工具。TimeNet 由柏林大学多届博士生共同完成的,积累了大量文档以及相关的学术论文,目前第四版已经面世。其功能强大,不光可以对扩展的 Petri 网进行稳态分析、暂态分析、仿真,还可以对高级有色 Petri 网进行分析。这提高了可靠性评估器的可扩展性,为下一步工作提供支持。

在 AADL 可靠性评估器中集成 TimeNet 工具后,通过后台调用 TimeNet 就实现了对 GSPN 模型的性能指标计算,将相应的可靠性评估结果展现给用户后,就实现了对 AADL 可靠性模型的自动化评估。

5.4.4 可靠性分析方法

在本文的 AADL 可靠性评估器中,提供了基于事件的敏感性分析方法,即以 AADL 错误模型中的故障事件为单位,得出每个事件随机参数的变化对于系统可靠性的影响程度,以此确定影响较大的故障事件。根据影响程度的不同,确定哪些故障事件对系统的可靠性影响最大,对后面的开发工作提供指导。例如通过有针对性的测试降低构件的失效率,替换其他软件构件,加入容错设计等方法提高系统的可靠性。

因为 AADL 错模型中的故障事件对应 GSPN 模型中的变迁，所以改变故障事件的随机参数后需要重新设置变迁的实施速率或开关分布概率。变迁随机参数的改变会导致 GSPN 状态空间的改变，生成的马尔可夫链会发生变化。所以需要在一组具有不同实施速率变迁的 GSPN 模型分别进行性能指标评价，得出相应的指标结果，才能实现对故障事件的敏感性分析。

```
Begin
  Do
    SelectFaultEvent(Event); //选择关心的故障事件
    InputScope(Parameter); //输入故障随机参数的范围
    GenerateGSPNAnalysisModel; //生成与参数对应的 GSPN 模型
  While() //选择完毕
  Do
    Analyzes(GSPN); //分析 GSPN 模型，得出该稳态概率结果
    WriteResult(); //记录结果
  While() //遍历所有的 GSPN 模型
    DrawCoordinate (); //绘制坐标图
End
```

算法 5.3 基于故障事件的敏感性分析

基于故障事件的敏感性分析算法如算法 5.3 所述。用户首先要选择关心故障事件，并输入该事件的随机参数范围。根据随机参数范围，生成一组 GSPN 模型。当选择完关心的故障事件后，对该组 GSPN 模型进行分析，得出随机参数对应的系统可靠性评估结果。最后绘制坐标图，横坐标为随机参数，纵坐标为系统可用度，以此将不同故障事件对系统可靠性的影响程度清晰地展现出来。

第六章 实例验证

6.1 飞行管理系统

本章结合飞行管理系统，给出一个基于 AADL 的嵌入式软件的可靠性建模、评估与分析的应用实例。飞行管理系统^[39](Flight Management System, FMS)是现代飞机中数字化系统的核心，它集飞行规划、性能优化、综合导航与制导、控制显示等功能为一体，通过飞行计划的制定和实施来实现飞行任务的自动控制，减轻了飞行员的负担。

根据功能划分，FMS 中有 6 个线程组成，分别是导航传感器处理线程(Navigation Sensor Processing, NSP)、综合导航线程(Integrated Navigation, INav)、制导处理线程(Guidance Processing, GP)、飞行性能优化线程(Aircraft Performance Calculation, APC)、飞行规划线程(Flight Plan Processing, FPP)和数据传输(Period IO, PIO)。图 6.1 中展示了 FMS 的 AADL 结构模型，所有线程构件通过共享数据区相互传输数据，最后通过 PIO 线程构件传送给其他子系统。

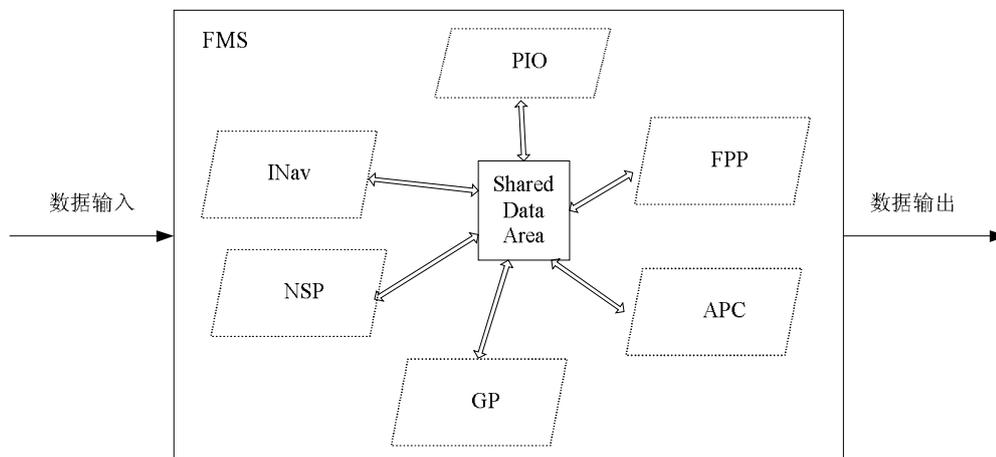


图 6.1 FMS 构件组成

在确定 FMS 的构件组成后，还需要确定各个线程构件之间如何传输数据。FMS 的数据输入首先经过 NSP 线程构件的基本处理，传输给 INav 线程构件并产生综合导航数据传给 GP、APC 和 PIO 三个线程构件，经过 GP 和 APC 线程构件处理后的制导和性能优化数据交给 FPP 线程构件，得出飞行计划数据。最终，所有输出给其他子系统的都经过 PIO 线程构件传给其他子系统。图 6.2 中的 AADL 结构模型中加入了流信息，以 FMS 的内部为视角，NSP 线程构件是流的起点，而线

程构件 PIO 是终点。

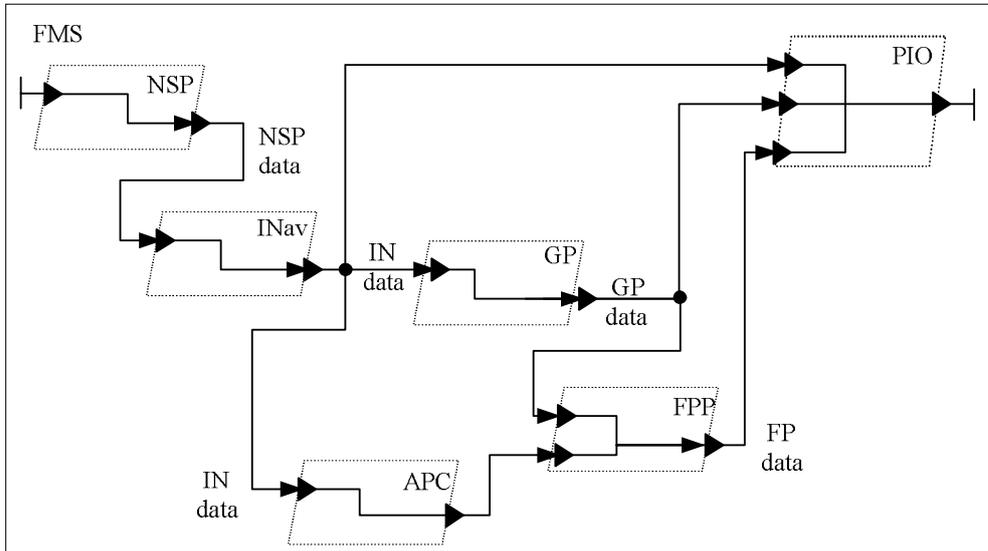


图 6.2 飞行管理器 AADL 结构模型

6.2 可靠性建模

6.1 节中阐述了 FMS 的背景及其 AADL 结构模型。当建立具备以上细节的 AADL 结构模型后，就可以进行可靠性建模工作。首先建立每个线程构件的单独构件错误模型，反映其自身的失效行为。为更明晰地表明本文所述的系统失效模型，线程构件的错误模型如图 6.3 所示，有正常(Error_Free)和失效(Failed)两种状态，其中正常状态为初始状态，失效事件(Fail)可以使构件从正常状态进入失效状态，而重启事件(Restart)则相反。

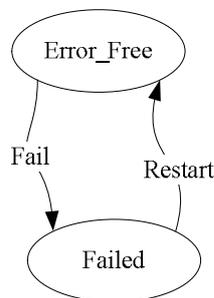


图 6.3 线程构件的单独构件错误模型

之后，根据 AADL 流信息确定系统的故障传播模型和系统失效模型。FMS 的失效模型如图 6.4 所示，每个线程构件在进入失效状态后，均有可能产生故障传播传出事件 CorruptedData，沿流路径影响其他线程。流路径确定了故障传播的路径，

而流的终点 PIO 线程构件的失效状态代表了系统的失效状态。

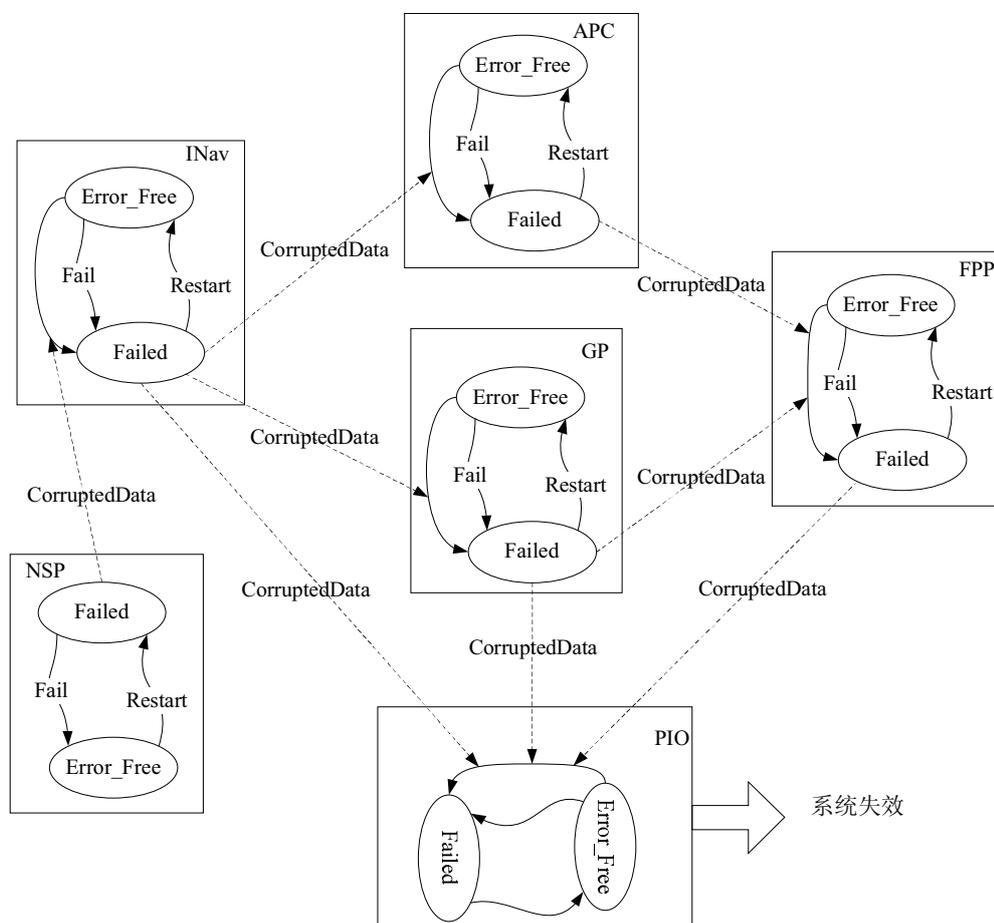


图 6.4 FMS 的 AADL 可靠性模型

最后，根据 FMS 的 AADL 可靠性模型，需要建立形式化的 AADL 错误模型，并将其绑定到 FMS 的每一个线程构件中。

6.3 模型转换

建立好可靠性模型后，需要进行模型转换工作。根据第五章中所述转换方法，首先将 AADL 可靠性模型中的每个线程构件转化成相应的 GSPN 模型，之后再为各个构件之间加入故障传播的 GSPN 模型。在 AADL 可靠性评估器中，模型转换过程对用户是透明化的。为了能更清楚地展示本文的评估过程，图 6.5 是模型转换后的 FMS 的 GSPN 可靠性模型。由于模型较大，以 INav 线程构件为界分两张图展示。其中 GSPN 模型中库所和变迁以构件名加状态名或事件名的形式命名。INav 线程构件的传出故障传播会影响三个线程构件，分别是 GP、APC 和 PIO 线程构件。

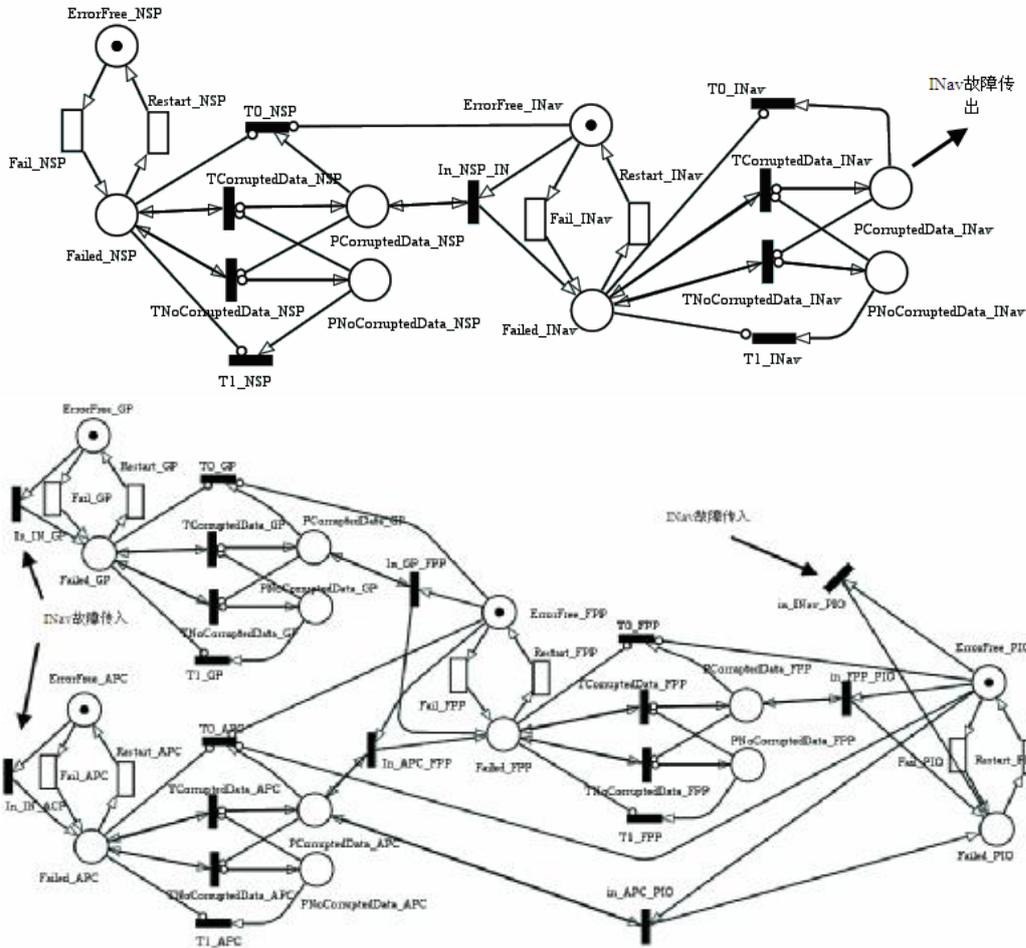


图 6.5 FMS 的 GSPN 可靠性模型

6.4 评估器运行结果

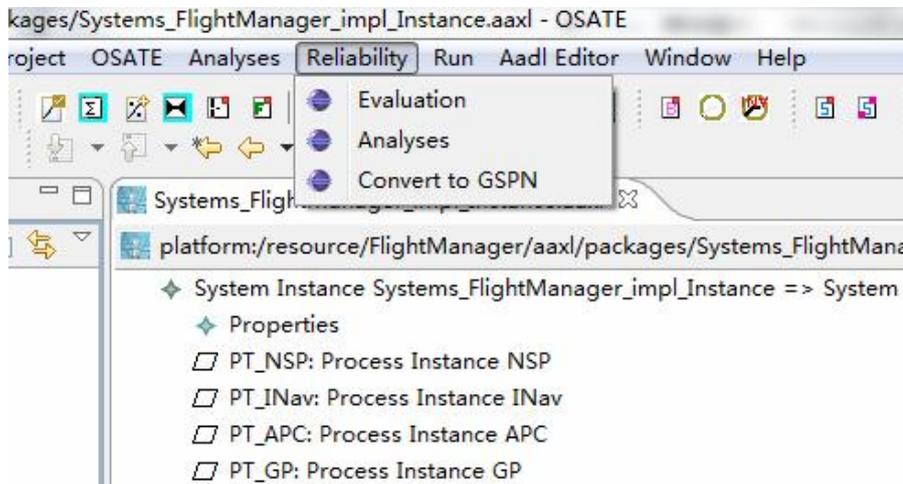


图 6.6 AADL 可靠性评估器菜单

将本项目的 AADL 可靠性评估器插件拷贝到 OSATE 插件包中，OSATE 环境界面出现图 6.6 中的菜单项，表明评估器安装成功。使用 OSATE 环境编辑好本章前面所述的 AADL 可靠性模型文件后，执行模型实例化，生成 AAXL 实例模型文件，就可以执行可靠性评估功能。图 6.7 是根据上述飞行管理器建立的 AADL 结构模型得出的 AAXL 实例。

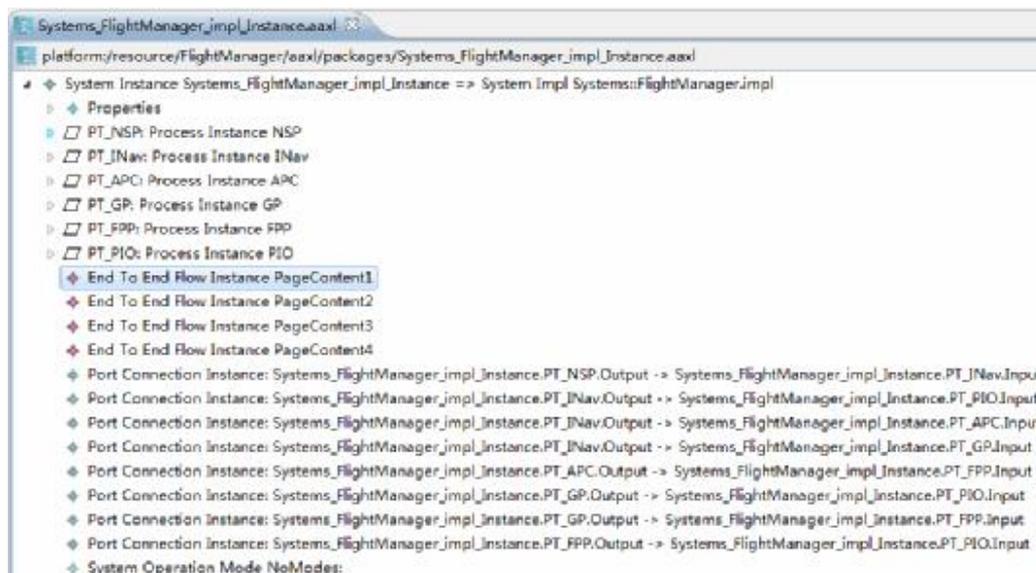


图 6.7 实例化 AADL 模型

假设各个线程构件以及故障传播的可靠性参数如表 6.1 和表 6.2 所示，执行 AADL 可靠性评估器评估功能后，给出的评估结果如图 6.8 所示。在评估结果中显示了每个线程构件的可用度以及系统的可用度。

表6.1 构件可靠性参数

事件	随机参数	事件	随机参数	事件	随机参数
NSP Fail	Poisson 5.0e-4	GP Fail	Poisson 5.0e-4	FPP Fail	Poisson 2.0e-4
NSP Restart	Poisson 1.0e-1	GP Restart	Poisson 1.0e-1	FPP Restart	Poisson 1.0e-1
INav Fail	Poisson 1.0e-3	APC Fail	Poisson 1.0e-3	PIO Fail	Poisson 5.0e-4
INav Restart	Poisson 1.0e-1	APC Restart	Poisson 1.0e-1	PIO Restart	Poisson 1.0e-1

表6.2 故障传播可靠性参数

路径	随机参数	事件	随机参数
EP (NSP, INav)	0.80	EP (APC, FPP)	0.65
EP (INav, GP)	0.75	EP (GP, PIO)	0.72
EP (INav, APC)	0.75	EP (INav, PIO)	0.75
EP (GP, FPP)	0.70	EP (FPP, PIO)	0.5

Compositon	Reliability
NSP	0.9950249
INav	0.9822720
GP	0.9763780
APC	0.9715944
FPP	0.9512654
PIO	0.9310102
SYSTEM	0.9310102

图 6.8 评估器评估结果

图 6.9 是使用 AADL 可靠性评估器对 INav 线程构件和 APC 线程构件的 Failed 事件进行敏感性分析得出的结果，不同事件的曲线由颜色进行区分，横坐标是故障事件的延迟时间，纵坐标是系统的可用度。可以看出，当 Failed 事件的延迟时间参数从 1000 变为 5000 时，即构件的无失效运行时间由 1000 变为 5000 时，相比于 APC 构件，INav 构件自身可靠性的提高对于 FMS 可靠性的提高影响更大。APC 构件的无故障运行时间变为 3000 后，已经接近了对系统可靠性提高的极限。

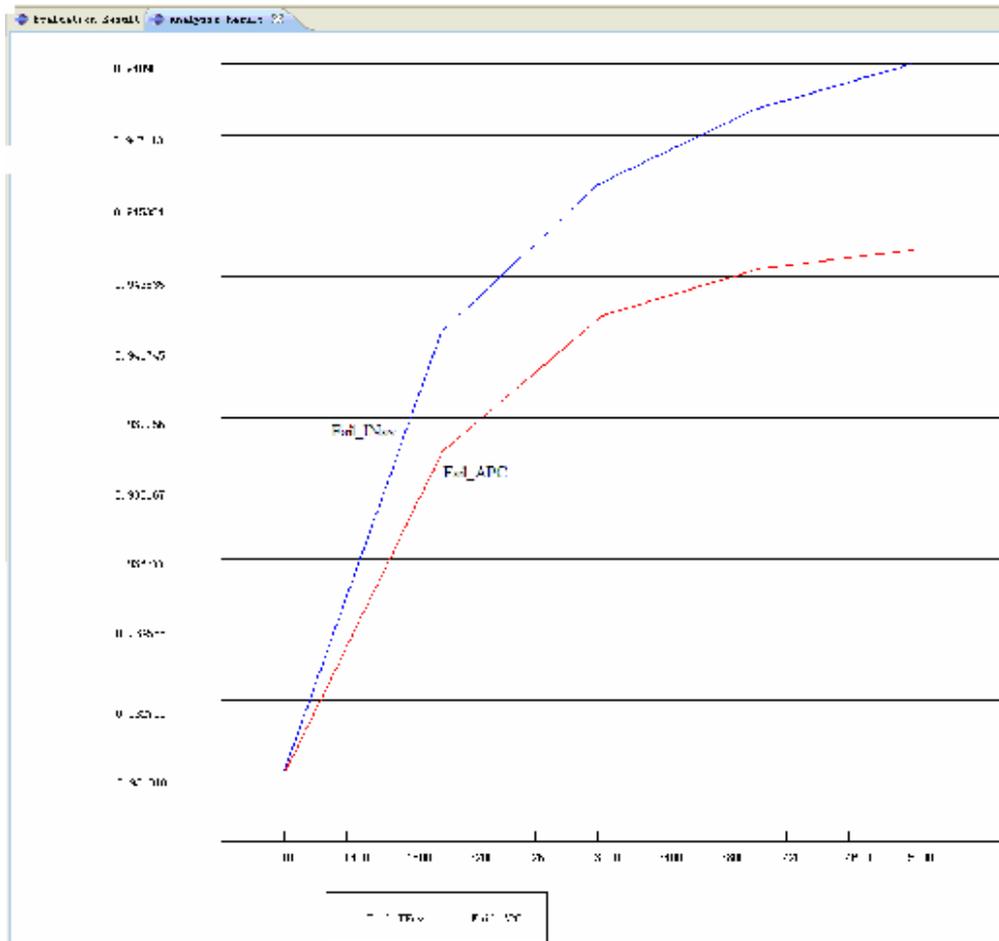


图6.9 评估器分析结果

第七章 总结与展望

本文以软件可靠性的发展现状和嵌入式软件工程的最新进展为背景，在对相关理论技术进行研究和分析基础上，提出了基于 AADL 的嵌入式软件可靠性建模和评估方法。该方法建立的软件可靠性模型能够更清晰地反映因嵌入式软件交互频繁而引起的故障传播因素，刻画构件之间的依赖关系，符合在早期对软件体系结构设计进行可靠性评估和分析的思想原则。本文的主要工作有：

(1)首先，研究了基于 AADL 的嵌入式软件系统的可靠性建模方法。该方法以 AADL 结构模型和错误模型为基础，刻画构件之间的故障传播关系，降低可靠性建模的复杂度，并使用 AADL 形式化语义进行描述。

(2)之后，研究基于 AADL 的嵌入式软件系统的可靠性评估和分析方法。要达到定量评估的目的，需要有软件性能定量评价工具。本文使用广义随机 Petri 网作为定量评估工具，它不仅反映了 AADL 错误模型描述的嵌入式软件的动态失效行为，还具有定量评估能力。通过模型转换，将 AADL 可靠性模型转换为 GSPN 模型，最终实现可靠性评估和分析，为下一步的软件开发工作提供依据。

(3)最后，根据本文的方法，开发了基于 AADL 的可靠性评估工具。该工具通过对 AADL 模型文件进行解析，获取结构模型和错误模型中的相应信息并根据转换规则生成对应 GSPN 模型，继而使用 GSPN 分析工具得出可靠性定量评估与分析结果。该工具实现了对 AADL 可靠性模型的自动化评估和分析功能，可以减轻开发人员的负担。

本文所作的工作只是软件可靠性研究中很小的一点，软件可靠性工作需要与软件工程同步展开，与开发中的各个阶段紧密结合，才能保证满足软件可靠性需求。本文工作中存在的不足与对未来工作的展望如下。

(1)本文对软件可靠性分析所做工作比较薄弱。由于随机 Petri 网不仅具有强大的动态建模能力，还有良好的数学分析和仿真能力，本文工作未能发挥随机 Petri 网理论的分析优势，在未来加强这方面工作，相信对软件可靠性保证工作大有裨益。

(2)GSPN 模型的一个主要问题是状态空间爆炸，这也是基于状态的软件可靠性评估方法的一个亟待解决的难点。但是可以利用 Petri 网理论中的压缩、分解技术以及随机高级 Petri 网等理论技术做一些工作，来解决这个问题。

(3)进一步完善 AADL 软件可靠性评估器，以使其更加实用。如加入 AADL 可靠性模型的图形编辑、查看功能，加强软件可靠性分析功能等。

致谢

基于导师申请的课题，在攻读硕士期间我对 AADL、嵌入式软件、软件可靠性领域进行了学习和研究，本论文是我攻读硕士期间的一个总结。所以我特别感谢我的导师李青山教授。李老师不光给了我锻炼的机会，还以身作则，通过他自身严谨的治学态度、丰富的科研经验和精益求精的工作作风为我树立了良好的榜样。他对我的严格要求，对工作的执着追求，对科研的一丝不苟将使我终身受益。在此谨向他表示最衷心的感谢和深深的敬意！

在本论文课题的进行期间，得到了课题组成员郑少华、苏春宇的极大帮助，大家一起克服了课题中的诸多困难。他们以敏捷的思维和良好的动手能力，对我的学习和工作产生了很大的影响，在此对他们表示感谢。

感谢上一届的刘鹤、负海顺师兄，在刚进入实验室时，他们给了我很多指导与建议，使我很快适应了实验室的科研环境。感谢好友魏柯宁、杜林、翟向前、段庆领和赵龙等，他们在学习中给了我很多鼓励，在人生的旅途上给了我很多帮助。

感谢西安电子科技大学的老师在研究生期间对我的谆谆教诲，他们丰富的学识和精深的学术造诣给了我许多有益的启发，他们的学者风范和敬业精神都令人钦佩。

感谢所有关心和支持过我的亲朋好友！

最后，我要真诚地感谢给予我无私关怀、鼓励的家人，正是他们的默默付出和支持才使我学业得以顺利完成。

参考文献

- [1] Mellor S., Clark A., Futagami T. Model-Driven Development IEEE Software, vol. 20, no. 5, pp. 14-18, Oct. 2003.
- [2] Kashi R.N., Amarnathan M. Perspectives on the use of model based development approach for safety critical avionics software development. International Conference on Aerospace Science and Technology, 2008.
- [3] Embedded Computing Systems Committee, Aerospace Avionics Systems Division, Architecture Analysis and Design Language (AADL), AS5506, Nov. 2004.
- [4] SAE International. Architecture Analysis and Design References Language (AADL), AS5506. 2004.
- [5] 孙昌爱, 金茂忠, 刘超. 软件体系结构研究综述. 软件学报, 2002, 13(07).
- [6] Delange J., Pautet L., Plantec A., etc. Validate, simulate, and implement ARINC653 systems using the AADL. ACM SIGAda Ada Letters Vol. 29, Issue 3, pp. 31-34, Dec. 2009.
- [7] Feiler P. H., Gluch D. P., Hudak J. J., The Architecture Analysis & Design Language (AADL): An Introduction, Technical report, CMU, SEI-2006-TN-011, 2006.
- [8] Hudon G. R. Program Errors as a Birth and Death Process. Report SP-3011, Santa Monica, CA: System Development Corporation, 1967.
- [9] Littlewood B., Verrall J. L. A Bayesian reliability growth model for computer software, 1973 IEEE Symp. Computer Software Reliability. New York, pp. 70-77, 1973.
- [10] Musa, J.D. A Theory of Software Reliability and System and Its Applications. IEEE Transactions on software Engineering, Vol. 12(3), 1975.
- [11] Trivedi A.K., Shooman M.L. A Many-State Markov Model for the Estimation and Prediction of Computer Software Performance Parameters. International Conference on Reliable Software, pp. 208-220, 1975.
- [12] Schneidewind N. F. Analysis of Error Processes in Computer Software, IEEE Transactions, Naval Postgraduate School, March 1975.
- [13] Shooman M. L., Natarajan S. Effect of manpower deployment and bug generation on software error models, Proc. Symp. Computer Software Engineering, pp. 155 1976.

- [14] Cheung R.C. A user-oriented software reliability model, *IEEE Transactions on Software Engineering*, 6(2), 118-125, 1980.
- [15] Shooman M. Structural models for software reliability prediction. *Proceedings of the second International Conference on Software Engineering*. 268-280, 1976.
- [16] Xie M. Wohlin C. All additive reliability model for the analysis of modular software failure data//*Proceedings of the Sixth International Symposium on Software Reliability Engineering*, 188-194, 1995.
- [17] 徐仁佐 软件可靠性专家系统. 北京: 清华大学出版社, 1996.
- [18] Cai K.Y. *Introduction to Fuzzy Reliability*, Kluwer Academic Publishers, Boston/Dordrecht/London, 1996.
- [19] Cai K.Y. *Software Defect and Operational Profile Modeling*, Kluwer Academic Publishers, Boston/Dordrecht/London, 1998.
- [20] Lu M.Y., Brocklehurst, Littlewood Combination of predictions obtained from different software reliability growth models, *Journal of Computer and Software Engineering*, v1, n4, 303-23, 1993.
- [21] Minyan Lu, Yunfeng Bai, Min Cong Practical software reliability measurement framework based on failure data, *Proceedings of the Annual Reliability and Maintainability Symposium*, pp.131-137, 2000
- [22] Karunanithi N., Whitley D. and Malaiya Y.K. Using NeuralNetworks in Reliability Prediction, *IEEE Software*, vol. 9, no. 4, pp. 53-59, 1992.
- [23] Malhotra M., Trivedi K. Dependability Modeling Using Petri Nets *IEEE Trans. Reliability*, vol. 44, no. 3, pp. 428-440, Sept. 1995.
- [24] Delanote D., van Baelen S., Joosen W., etc. Using AADL in model driven development. In: Canals A, Gerard S, Perseil I, eds. *Proc. of the Int'l Conf. on Engineering Complex Computer Systems*. Washington: IEEE Computer Society Press, pp. 1-10, 2007.
- [25] Dessaux P., Singhoff F., Stood and Cheddar: AADL as a pivot language for analysing performances of real time architectures. *Proc. of the 4th European Congress on Embedded Real-Time Software*. Toulouse, France, Jan. 2008.
- [26] Jouault F., Kurtev I. Transforming models with ATL. In *Proc. of the Model Transformations in Practice Workshop at Models 2005*. Berlin: Springer-Verlag, 128-138, 2006.
- [27] Singhoff F., Legrand J., Nana L., etc. Scheduling and memory requirements analysis with AADL. *Ada Lett.*, XXV(4):1-10, 2005.
- [28] Singhoff F., Legrand J., Nana L., etc. Cheddar: A flexible real time scheduling

- framework. *ACM Ada Letters*, 24(4):1–8, 2004.
- [29] Rugina A.E., Kanoun K., Kaâniche M. An architecture-based dependability modeling framework using AADL. *Proc. of the 10th Int’l Conf. on Software Engineering and Applications*. Washington: IEEE Computer Society Press, 222–227, 2006.
- [30] Rugina A.E., Kanoun K., Kaâniche M. A system dependability modeling framework using AADL and GSPNs. de Lemos R, et al., eds. *Architecture Dependable Systems IV*. LNCS 4615, 14–38, 2007.
- [31] Boudali H., Crouzen P., Haverkort B.R., etc. Arcade: A formal, extensible, model-based dependability evaluation framework. *Proc. of the 13th IEEE Int’l Conf. on Engineering of Complex Computer Systems*.
- [32] Sun H., Hauptman M. and Lutz R.R. Integrating Product Line Fault Tree Analysis into AADL Models. In *Tenth IEEE Int. Symp. on High Assurance Systems Engineering (HASE 2007)*, pp. 15–22. IEEE Computer Society, 2007.
- [33] 何国伟, 王纬 软件可靠性. 北京: 国防大学出版社, 1998.
- [34] 阮镰, 陆民燕 软件可靠性工程的研究现状和发展趋势 中国航空学会 2005 年学术年会论文.
- [35] Musa J. D. 软件可靠性工程. 韩柯, 译. 北京: 机械工业出版社, 2003.
- [36] Abdelmoez W., Nassar D.M., Shreshevsku M., etc. Error propagation in software architectures. In: *10th IEEE international software metrics symposium (METRICS 2004)*, 11–17 September 2004, Chicago, USA: IEEE Computer Society; 2004..
- [37] 林闯. 随机 Petri 网和系统性能评价. 北京: 清华大学出版社, 2000
- [38] German R., Kelling Ch., Zimmermann A., etc. TimeNET - a Toolkit for Evaluating Non-Markovian Stochastic Petri Nets. *Perf. Eval.*, 24:69-87, 1995.
- [39] Feiler P.H., Gluch D.P., Hudak, J.J., etc. *Embedded Systems Architecture Analysis Using SAE AADL*. Technical Report CMU/SEI-2004-TN-005, Carnegie Mellon University, 2004.

在研期间研究成果

参加科研项目

项目名称：“XXX 领域的软件可靠性评估方法研究”，课题来源：航空科学基金，起止年月：2008 年-2010 年，职务：组长。