一种基于故障扩展SysML活动图的安全性验证框架研究

仵志鹏 黄志球 王珊珊 曹德建
(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 随着嵌入式系统在能源、交通等安全关键领域的广泛应用,针对嵌入式软件的安全性分析与验证方法一直是学术界和工业界的研究热点之一。使用扩展了故障树语义信息的SysML活动图来统一系统的功能模型与安全需求分析模型,并在保留故障树和SysML活动图两种模型语义描述的基础上,提出了一种基于故障扩展SysML活动图的安全性验证框架,包括:首先利用故障树最小割集提取故障信息并给出故障树逻辑门的转换规则;然后给出故障扩展SysML活动图的构建步骤;最后使用Promela对故障扩展SysML活动图进行建模,并使用模型检测工具SPIN对其进行分析验证。通过一个燃气灶控制系统验证了此方法的有效性。

关键词 安全性验证,故障树语义,SysML活动图,Promela
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.7.048

Research on Framework of Safety Verification Based on Fault-extended SysML Activity Diagram

WU Zhi-peng HUANG Zhi-qiu WANG Shan-shan CAO De-jian
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract Embedded system has been widely used in safety-critical areas such as energy, transportation, etc. The safety analysis and verification for embedded software have always been one of the hot topics in both academia and industry. In order to unify function model and safety requirements analysis model of the system, we introduced extended SysML activity diagrams with semantic information of fault tree. On the basis of retaining the semantic descriptions of both the fault tree and the SysML activity diagram, we presented a kind of safety verification framework based on fault-extended SysML activity diagrams. Firstly, we used minimal cut sets to extract fault information and presented transformation rules of fault tree logic gate. Then, we presented build steps of fault-extended SysML activity diagrams. Finally, we used Promela to model fault-extended SysML activity diagrams and used model checking tool SPIN to analyze and verify it. We verified the effectiveness of the method by using it in a gas stove control system.

Keywords Safety verification, Semantic information of fault tree, SysML activity diagram, Promela

1 引言

嵌入式软件在汽车、核能、航空等安全关键领域的普遍应用使得软件的失效可能造成财产的损失、环境的破坏甚至人员的伤亡,保障嵌入式软件的安全性已成为近年来软件工程领域的研究热点[1]。为了提高嵌入式软件系统的安全性和可靠性,针对软件模型的安全性分析与验证已经成为软件开发周期中十分重要的一环。

故障树分析法(Fault Tree Analysis, FTA)[2]作为安全性分析的传统方法,在嵌入式软件系统安全工程中扮演着重要的角色。故障树分析法通过顶事件描述系统的某一特定故障结果,通过自上而下的演绎方法分析该故障产生的原因,描述了软件故障结果和产生原因之间的逻辑关系。而在软件系统中,故障发生往往是由于正常行为的组合错误或执行顺序错误,而不是软件行为本身的问题。故障树分析法虽然能够分析出哪些性质可能导致系统失效,但并不能证明系统本身是否存在这些性质。

统一建模语言(Unified Modeling Language, UML)[3]作为描述能力强大且涵义直观的可视化建模语言,已被工业界认可且得到了广泛的使用。但是UML在嵌入式实时系统建模上存在缺少一致性、模型互操作性差、对系统工程建模能力不足等问题。SysML[4](System Modeling Language)是系统工程应用开发的标准建模语言,支持对各种复杂系统进行详细说明、分析、设计和验证。SysML活动图作为一种半形式化模型,被广泛应用于软件的功能建模。但SysML活动图主要描述的是系统的行为,而忽略了对故障信息的描述。故障树分析可以很好地弥补这种不足。

功能模型和安全需求分析模型是分析嵌入式安全关键系

到稿日期:2014-07-31 返修日期:2014-09-28 本文受国家自然科学基金(61100034,61170043),中国博士后科学基金(20110491411),江苏省普通高校研究生科研创新计划资助项目,中央高校基本科研业务费专项资金(CXZZ11_0218)资助。
仵志鹏(1989—),男,硕士生,主要研究方向为软件工程、形式化验证,E-mail:zhipengwu2013@163.com;黄志球(1965—),男,教授,博士生导师,CCF杰出会员,主要研究方向为软件工程;王珊珊(1963—),女,副教授,主要研究方向为软件工程、软件度量;曹德建(1988—),男,硕士生,主要研究方向为软件工程、形式化验证。

统的两个重要方面，但两种模型一般都被分开使用。在软件开发过程中，故障分析是保证系统功能模型安全性和正确性的关键。本文提出了一个针对故障扩展 SysML 活动图的安全性验证框架：首先给出了一种借助故障树信息，针对故障问题对软件系统 SysML 活动图进行扩展并保留系统行为描述和故障逻辑的方法；然后对故障扩展 SysML 活动图利用 Promela 进行建模；最后根据安全需求规约和 Promela 模型通过 SPIN 进行安全性验证。该方法能够帮助安全分析人员找到导致系统故障发生的行为组合，更好地理解系统各组件及其行为之间的逻辑关系；为安全关键系统的建模验证工作提供证据，提高安全关键系统的安全性。

本文第 2 节简要介绍故障树分析法和 SysML 活动图，给出故障扩展 SysML 活动图的相关定义，并提出了基于故障扩展 SysML 活动图的软件安全性验证框架；第 3 节介绍了基于故障树最小割集的故障规约的提取和故障扩展 SysML 活动图的构建步骤，给出了使用 Promela[5] 对故障扩展 SysML 活动图建模的过程；第 4 节使用本文提出的方法对一个小型燃气灶控制系统进行了验证；第 5 节介绍相关工作；最后总结全文。

2 故障树分析法与 SysML 活动图

2.1 故障树分析法

故障树分析法[2]关注的是导致系统失效的故障行为。它以系统的某一具体故障作为分析对象，通过对可能造成故障的各种因素进行分析，确定导致故障发生的各种可能的原因，并通过逻辑门等符号直观地描述故障与故障成因之间的逻辑关系，从而提前发现系统的设计错误和安全隐患，用于指导软件设计和后期的维护工作。

事件和逻辑门是故障树最基本的构造单元。根据故障的类型，事件可分为顶层事件、中间事件和基本事件 3 种。逻辑门用来表示各种故障因素的关系，常见的逻辑门有与门、或门、优先与门、非门、表决门、禁门等。这些逻辑门使分析人员能够清晰地构建出系统各故障及组件之间的关系，定位故障的位置。

故障树的分析主要包括定性分析和定量分析两类。定性分析是为了找出导致顶层事件发生的所有可能的故障模式，分析各类型故障的危险性，定位安全薄弱环节，安排预防保障措施以避免故障发生[6]。定量分析通过确定每个基本事件发生的概率来计算顶层事件发生的概率，并以此作为系统安全评估的评级标准之一，为系统安全优化提供科学依据[7]。

确定故障树的最小割集，是故障树定性分析中最重要的方法之一。割集是能使顶层事件发生的基本事件的集合。最小割集是能引起顶层事件发生的基本事件的最小集合。最小割集中任意一个基本事件不发生，则顶层事件就不会发生。计算故障树最小割集的方法有许多，其中最著名的是布尔代数法、下行法和上行法。

然而，故障树分析法由于是以对系统的非形式化描述为基础，缺乏精确语义[8]，无法对软件故障进行形式化规约和建模验证。

2.2 SysML 活动图

SysML 活动图[9]可以用来对嵌入式实时系统的活动行为建模。在 SysML 活动图中，控制既能使动作开始，又能使正在执行的动作终止，SysML 扩展了对象节点并为活动扩展了类图符号，说明了活动之间的组合关联语义，定义了活动图和类图之间的一致性规则。

SysML 活动图能更好地描述嵌入式实时系统中存在的大量活动行为。一方面，使用控制作为数据来控制活动的开始和终止，能够更好地控制活动按照安全可靠的方向流动；控制操作符的使用能够精确地定义活动的输入和输出，从而提高系统活动的可控性以及安全性，也为活动图转换到形式化的表达方式提供了精确的语义。另一方面，针对实际系统的活动，存在着各种因素的影响，并非每个活动都是按照活动次序一定发生。尽管 SysML 活动图能够有效地描述系统的活动行为，但是它依然缺少类似故障树分析中定性分析的能力。

2.3 故障扩展 SysML 活动图

根据上述对故障树分析法和 SysML 活动图的介绍，可以发现，传统的安全分析方法尽管在各自的应用场景发挥了巨大的作用，但是依然存在一些不足。

首先，SysML 活动图能够有效描述系统的行为和执行过程，是系统功能常见的描述方式，但缺乏对系统故障信息的直观描述，隐含的可能导致系统失效的行为或行为组合很难被发现；其次，故障树分析侧重的是系统故障问题、引起故障的原因和它们之间的关系，却没有把导致故障的原因与系统的具体行为联系起来，难以确认系统中是否存在这样的故障；然而，在系统软件安全性分析[10]工作中，分析验证人员不仅要对表达系统状态行为的功能模型具有全面认识，同时也应对系统的故障模型有深入的了解，需要将系统具体的行为与可能引起的故障关联起来。因此，包含故障信息的系统行为模型在系统的安全性分析工作中显得尤为必要。

针对以上需求，本文通过对 SysML 活动图进行故障扩展，将系统功能模型与安全需求分析模型合并，给出故障扩展 SysML 活动图的相关概念。

概念 1（故障扩展 SysML 活动图，Fault-extended SysML Activity Diagram） 以系统原始 SysML 活动图为基础，通过在其上扩充故障逻辑活动，使其能够同时表达系统安全需求和功能行为的扩展 SysML 活动图。

在故障扩展 SysML 活动图中，其故障逻辑活动来自故障树的故障信息，而系统原始 SysML 活动图本身不包含这些信息，因此故障逻辑与原始 SysML 活动图无关，这种无关性在扩展 SysML 活动图中表现为“正交”，即两个区域在层次上是等价的，且两个区域之间没有任何的控制流和对象流。本文将这两个相互正交的区域分别称为故障域和功能域，并给出如下概念。

概念 2（故障域，Fault Region） 故障扩展 SysML 活动图中描述故障逻辑活动的集合，来自于故障信息的转换，与功能域相互正交。

概念 3（最小割集故障域，Minimal Cutset Fault Region） 故障扩展SysML 活动图中故障逻辑活动的集合，属于故障域的一部分。

概念 4（功能域，Function Region） 故障扩展 SysML 活动图中描述系统行为的活动的集合，来自于系统原始 SysML 活动图，与故障域相互正交。

由于故障域的活动不会对系统功能造成影响，因此这种对系统 SysML 活动图进行故障扩展的方式，支持故障逻辑的动态扩充。当新的故障信息出现时，可以直接对已扩展的

SysML 活动图进行增量扩充，而无需重新构造。

2.4 基于故障扩展 SysML 活动图的安全性验证框架

该安全性验证框架主要包含两方面工作：一方面，通过确定故障树的最小割集和提取故障信息，利用逻辑门的转换规则对系统原始的 SysML 活动图进行故障扩展；另一方面，对包含故障信息的故障扩展活动图利用 Promela 语言进行形式化建模，结合模型检测工具 SPIN 对系统的安全属性进行验证分析。具体的安全性分析框架如图 1 所示。

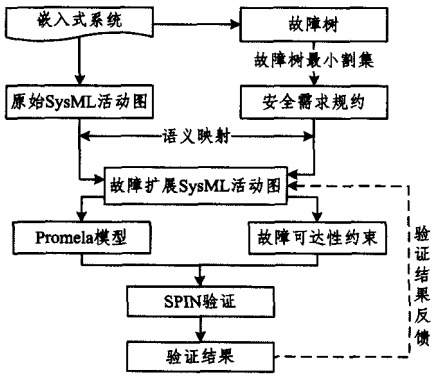

图 1 基于故障扩展 SysML 活动图的安全性验证框架

3 故障扩展 SysML 活动图的构建与建模

本节主要介绍故障扩展 SysML 活动图的构建，以及对得到的故障扩展 SysML 活动图使用 Promela 进行建模。

3.1 故障扩展 SysML 活动图的构建

故障扩展 SysML 活动图能够同时描述系统的安全性需求和组件的行为，它由两部分信息组成：功能信息和故障信息。其中的功能信息由系统原始 SysML 活动图表示，因此只要在原始 SysML 活动图中增加有关故障信息的描述，就可以构建系统的故障扩展 SysML 活动图，具体的构造过程如图 2 所示。因此在构建故障扩展 SysML 活动图时只需要给出故障域的构建过程。

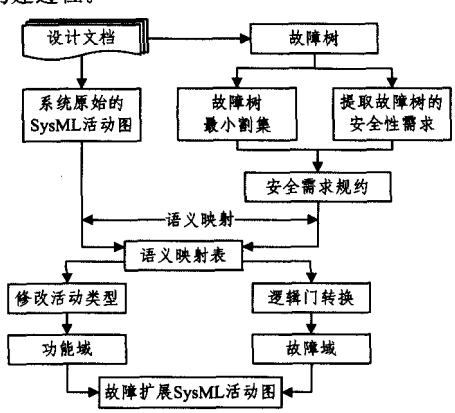

图 2 故障扩展 SysML 活动图的构建过程

故障信息的提取：传统的故障树分析直接从故障树本身入手，没有考虑故障树模型的转换，当叶子节点描述的不仅是单组件的活动时，就需要根据其语义将其再次分解，然后才能与 SysML 活动图中的活动相关联。因此，故障树中故障信息的提取包括 3 个步骤：1）确定故障树的最小割集；2）从故障树中提取安全性需求；3）建立语义关联，完成故障树故障信息与 SysML 活动图中组件活动的映射，统一模型间的语义。

步骤 1 确定故障树的最小割集。本文利用下行法来确定故障树的最小割集，对故障树进行自顶向下的分析，逐层进行事件的替换，用下层事件将上层事件替换。同时依据与门增加割集容量、或门增加割集数量的原则，遇到“与”门，则横向列出此门的输入事件，遇到“或”门则纵向列出此门的输入，直到全部逻辑门都被基本故障事件置换。然后运用集合运算法将割集进行化简，得到故障树的最小割集。

步骤 2 提取故障树的安全性形式化规约。故障树的安全性需求和其表达式符号的语法可以用 Backus-Naur 范式表示，如图 3 所示。其中〈Gate〉代表故障树的逻辑门，〈Gate-Input〉代表逻辑门的输入，〈leaf-node〉代表基本事件，〈operand〉表示连接各类型事件的逻辑操作符。由于在步骤 1 中确定了最小割集，假定我们求的最小割集的逻辑操作符仅包含“或”和“与”两种，则〈operand〉表示的逻辑操作符仅包含逻辑与 ∧、逻辑或 ∨ 两种，分别与故障树中的与门、或门对应。

〈Safety-Requirement〉::=〈Gate〉

〈Gate〉::=(〈operand〉,〈Gate-Input〉)

〈Gate-Input〉::=〈leaf-node〉,〈leaf-node〉|〈leaf-node〉,〈Gate〉
|〈leaf-node〉,〈Gate-Input〉|〈Gate〉,〈Gate-Input〉

〈operand〉::=∧|∨

图 3 安全性需求规约表达式的 BNF 语法

根据这一语法，故障树中的安全性需求就可以表达为由〈leaf-node〉和〈operand〉组成的布尔表达式，当且仅当表达式为真时，顶层故障事件才会发生。

步骤 3 建立故障树与 SysML 活动图元素的语义映射表。建立语义映射，确定基本事件与系统 SysML 活动图等价的部分。对于软件控制系统，导致故障发生的简单事件都应在相关的活动和守卫条件中体现出来。建立语义映射表的具体过程如下：1）根据系统功能需求描述，结合活动图，找出与简单事件功能性相关的组件在活动图中的活动和守卫条件的集合；2）分析简单事件的故障语义，在组件的活动和守卫条件的集合中找到与其描述一致的活动或者守卫条件；3）依照这一对应关系，建立简单事件与活动图元素的语义映射表。

故障树逻辑门的转换规则：由于本方法采用最小割集对故障树进行分析，而且所有的逻辑门都可以转换成逻辑与、或、非，因此给出或门和与门的逻辑转换规则。与门是故障树中最常见的逻辑门之一，在安全需求表达式中用符号 ∧ 表示，表示当且仅当输入事件全部发生时，输出事件才会发生。如图 4 所示，故障树中 A、B 事件同时发生后事件 T 发生，则在活动图中将 A、B 转换成并发的两个活动，表示只有 A、B 活动都完成后，T 活动开始。

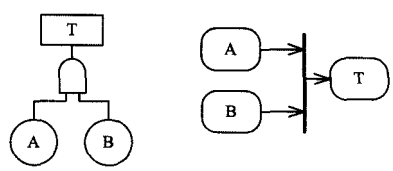

图 4 与门的转换

或门也是故障树中重要的逻辑门之一，表示当输入事件中至少有一个发生时，输出事件发生，用符号 ∨ 表示。如图 5 所示，故障树中 A 或 B 事件发生，则 T 事件发生，活动图中使用合并结构表示 A、B 活动，A、B 活动中任何一个完成，则 T 活动开始。

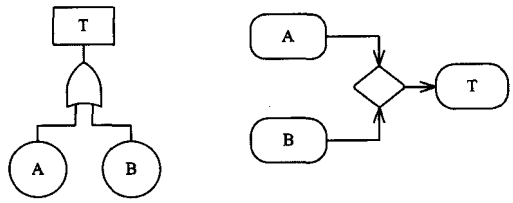

图 5　或门的转换

故障域的构建：在故障扩展 SysML 活动图中，故障域以活动图的形式描述了系统故障信息中的故障逻辑。根据前面的描述，故障树中包含的故障信息可以表示为一个简单事件和逻辑操作符构成的布尔表达式。下面将利用该布尔表达式构造故障扩展 SysML 活动图的故障域。

在利用安全需求规约构造故障域时，根据本节中的方法，首先利用故障树的最小割集得到安全需求规约表达式，接着建立简单事件与 SysML 活动图的语义映射表，最后参照故障树分析中“上行法”的思想，自下而上逐层转换复合规约，直到所有规约表达式转换完成。算法描述如下：

1)从左至右读取安全需求规约，直到遇到第一个操作符，分析其操作对象，判断规约类型；

2)如果规约为简单规约，则以操作对象为输入，该规约为输出，根据操作符的类型，依照给出的故障树逻辑门的转换规则并结合语义映射表，在活动图中绘出其对应的逻辑域，并以该规约命名；

3)如果规约为复合规约，且其操作对象中的嵌套表达式(简单规约或复合规约)未被绘出，则该表达式为新的安全需求规约，返回步骤 1，递归执行整个流程；

4)如果规约为复合规约，且其操作对象中的嵌套表达式均已被绘出，则以这些表达式的转换结果和规约中的简单事件(如果存在)为输入，以该规约为输出，在活动图中绘出其对应的逻辑域。

系统原始 SysML 活动图的改造：通过修改活动的类型，使之具备驱动故障域的能力。若系统原始 SysML 活动图与简单事件对应的活动不是信号发送活动，则将其修改成信号发送活动，同时在故障域中用与该活动对应的一个信号接收活动表示与之对应的简单事件。

经过上述修改，系统的原始 SysML 活动图具备了驱动故障域的能力。与这些修改过的活动对应的信号接收活动均只存在于故障域中，因此不会对系统的功能域造成任何影响。

3.2　故障扩展 SysML 活动图的 Promela 建模

在 3.1 节中，通过将故障信息添加到 SysML 活动图中，形成了故障扩展 SysML 活动图。在此基础上，给出与 SysML 活动图中一些基本元素等价的 Promela 模型的转化规则，然后进行重组完成故障扩展 SysML 活动图对应的 Promela 模型。

转化规则：首先将故障扩展 SysML 活动图拆分成若干个包含基本元素的独立子图。为了保留活动图中并发特性转换前后的一致性，参考文献[11,12]将基本元素转换为 Promela 相应的等价语句，提出如下的转换规则：把每个元素单独映射为 Promela 语言的单独进程，而把活动图的转移映射为 Promela 信道。为了防止命名冲突，把进程的名字命名为 action 加节点的 id 值，如 action_id。

规则 1(迁移)　SysML 活动图的节点之间通过转移进行连接，从而在 Promela 的进程之间通过信道进行消息交互。我们定义 chan edges=[10]of{int,msg}；定义两个数据域，第一个表示信息的标志，即活动图中转移的 id 值；第二个表示传输的数据包。进程在信道等待接收其需要的数据包，如果信道中有该数据包的 id 值，则接收数据包，否则继续等待。

规则 2(初始节点)　活动图中仅有一个初始节点完成整个活动图的启动作用。初始节点的转换如图 6 所示。

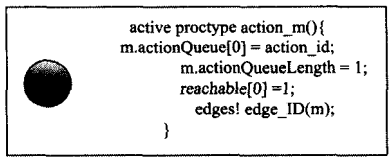

图 6　初始节点的转换

规则 3(活动节点)　当相应的进程完成了数据包的接收，并在数据包中完成了本进程 id 值的添加后，将其发送给下一个进程。活动节点的转换如图 7 所示。

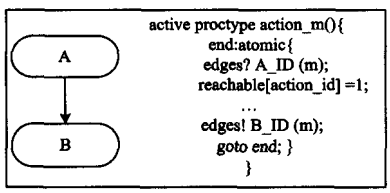

图 7　活动节点的转换

规则 4(分岔节点)　分岔节点表示把一个单独的控制流分成二个或多个并发的控制流。其转换如图 8 所示。

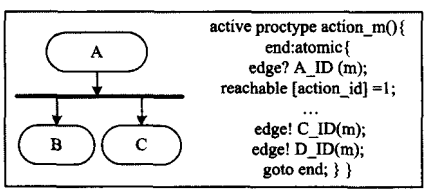

图 8　分岔节点的转换

规则 5(汇合节点)　刚好与分岔节点相对应，汇合节点多个并发控制流同步发生，此时需要记录本进程的 id 值，还需要合并数据包。汇合节点的转换如图 9 所示。

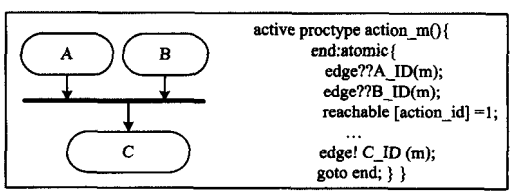

图 9　汇合节点的转换

规则 6(分支节点)　分支节点的每条输出转移都为可执行的，但某一时刻只能选择其中一条。其转换如图 10 所示。

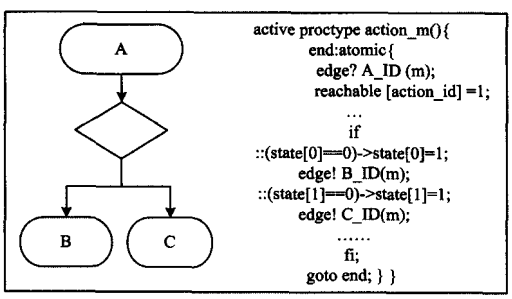

图 10　分支节点的转换

规则 7(合并节点)　合并节点是二个或多个进入转移和

一个输出，多个进入转移只有其中的某一个有效。其转换如图 11 所示。

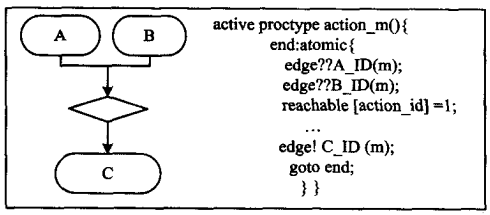

图 11　合并节点的转换

规则 8(终止节点)　终止节点是活动图的完结活动。终止节点的转换进程处理从起始节点开始的数据包，如图 12 所示。

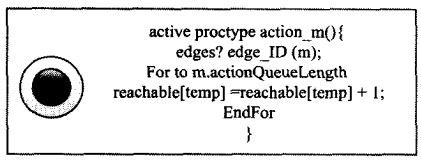

图 12　终止节点的转换

重组 Promela 模型：前面定义转换规则时将活动图按基本元素拆分为若干个独立子图，并将其作为转换单元转换为相应的 Promela 模型，但是各个模型间是相互独立的，所以要想被 SPIN 所识别，需要重组所有的 Promela 模型。

重组的基本原理是：基于活动子图之间的嵌套关系，用子图替代复合活动节点，即修改子图的初始点和终止点为普通活动节点，同时这两节点将负责复合节点的进入转移和输出转移工作。

1)子图起始节点：活动图中的复合节点都是单输入转移，而子图的起始节点没有输入转移。此时将子图的起始节点修改为普通活动节点，同时把复合节点的输入转移改为起始节点的输入转移，起始节点的输出转移不变。相应的起始节点进程接收发送给复合节点进程的数据包。

2)子图终止节点：活动图中的复合节点都是单输出转移，而子图的终止节点没有输出，此时将终止节点修改为普通节点，并将复合节点的输出转移改为终止节点的输出转移，其输入转移不变。相应终止节点发送本该由复合节点发送的数据包。

3)复合节点：子图的起始节点和终止节点已替代复合节点的输入转移和输出转移，为了防止信息的冲突，把该复合节点删除，并删除相应的复合节点进程。

利用上面的重组方法将活动图中的所有复合节点用相应的活动子图替换，从而使原来的多层次活动图变成一张完整的活动图，同时各个独立的活动子图 Promela 进程连接成为一个可通信的整体。整个 Promela 程序只保留着顶层的起始节点和终止节点，而子图的起始节点和终止节点只是作为普通的单输入单输出的活动节点，直到得到完整的 Promela 模型。

4　实例分析

本节通过对工程实践中一个微型燃气灶控制系统进行分析，说明本文提出的方法的有效性。

4.1　燃气灶控制系统

燃气灶的主要功能就是通过燃烧燃气产生热量。燃气灶控制系统的需求说明主要有以下几点[13]：1)在任何时间，燃气灶周围的燃气浓度都应该小于临界值；2)当有热请求的时候，燃气灶必须能正常工作并产生热量；3)当没有热请求的时候，燃气灶必须停止工作并关闭燃气和空气的输入。燃气灶控制系统主要包括 4 个组件[14]：空气开关控制系统、燃气开关控制系统、火焰监测器、点火控制系统。根据文献[15,16]，得到燃气灶控制系统的原始 SysML 活动图，如图 13 所示。其中的⊗表示禁门，由于其可以用与门和或门表示，到达此活动表明其不影响功能。

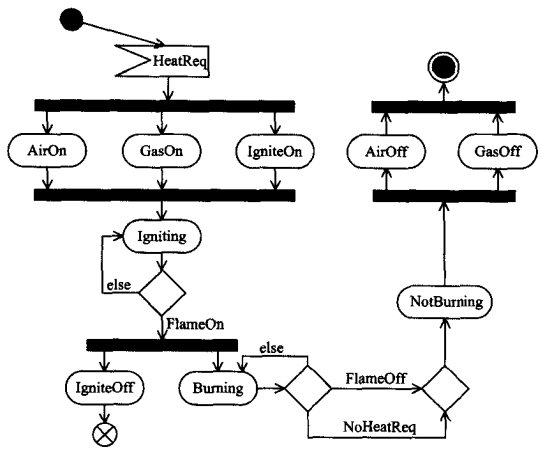

图 13　燃气灶控制系统的原始 SysML 活动图

根据文献[14,17,18]，通过提取故障树故障信息建立一棵故障树，然后通过最小割集的方法得到最小割集故障树，如图 14 所示，即当燃气灶周围燃气浓度超过临界值且进行点火或内部短路时，将导致爆炸事故。其中该故障树的最小割集为(Air Present，Gas Open $t>4$s，Ignition Attempted)和(Air Present，Gas Open $t>4$，Short Circuit)。

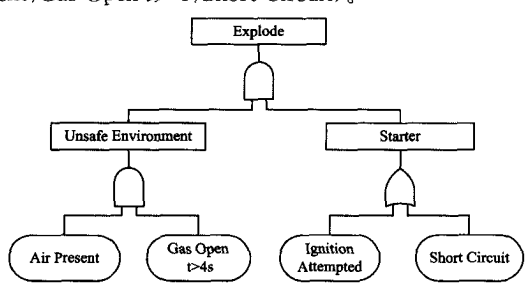

图 14　燃气灶爆炸事故故障树

4.2　故障扩展 SysML 活动图的建立

故障扩展 SysML 活动图的构建分为两部分，首先依据故障树的安全需求规约和语义映射表，按照故障树逻辑门的转换规则，构造故障扩展 SysML 活动图的故障域；然后根据语义映射表对系统原始 SysML 活动图进行改造。结合故障树和原始系统故障图通过第 3 节的方法得到故障扩展 SysML 活动图，如图 15 所示。图 15 虚线左侧功能域所示的活动图与燃气灶控制系统原始 SysML 活动图唯一不同的地方就是 AirOn、GasOn、IgniteOn 这 3 个活动从普通活动变成了信号发送活动，即在正常打开开关的同时发送出开关已打开的信号。当故障域接收到燃气阀门打开的信号 4s 以后，进行判断，如果火焰监测器监测到火焰，说明点火成功，则故障域中的进程结束，不会发生爆炸事故；否则说明燃气泄漏，进入“UnsafeEnvironment”活动状态，此时若点火器仍在点火或者发生短路故障，则将发生爆炸事故。在功能域中，火焰监测器监测不到火焰，点火器将一直点火，显然若这个过程持续 4s 以上，故障域中将到达“Explode”活动状态，即发生爆炸事故。

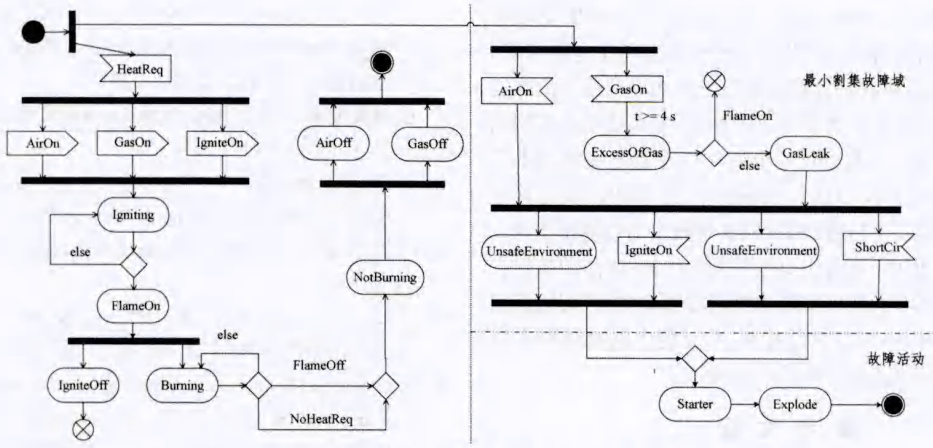

图 15　燃气灶控制系统的故障扩展 SysML 活动图

4.3　燃气灶故障扩展 SysML 活动图的建模验证

通过 3.2 节给出的 SysML 活动图转换规则，可以得到故障扩展 SysML 活动图对应的完整的 Promela 模型源码。这里仅给出故障域对应的 Promela 源码片段，如图 16 所示。

```
active proctype q() {
if
:: edge??[2,Gon]->atomic{
   if :: (time>=4)->msg!1,Excess;
      if ::edge??[5,fon]->skip
        ::else->msg!2,GasLeak;
      fi;
     ::else->skip;
   fi;}
:: else->skip;
fi;
...
if
::(msg??[0,unsafeE]&&edge??[3,on])||
(msg??[3,unsafeE]&&msg??[4,shortCir])->msg!5,explode;
   reach=1;
::else->skip;
fi;
}
ltl m{ []<>(reach==0)}
```

图 16　故障扩展 SysML 活动图故障域的 Promela 转换源码片段

系统的安全性验证转化为故障活动的可达性验证。在实例中，描述故障活动不可达的时序逻辑公式为[]〈〉(reach==0)。本文将针对故障扩展 SysML 活动图所建立的完整的 Promela 模型作为验证模型，以该公式作为验证命令的公式输入，在 SPIN 的环境下，进行模型检测的运行，结果如图 17 所示。从图 17 中可以看到存在一个错误，说明故障活动可达。经过对故障扩展 SysML 活动图中功能域的分析，发现缺乏 Igniting 离开到 FlameOn 的约束，只有尝试打火 3 次仍不能打着，才需要停止继续尝试。通过计数器进行控制，可以修复这个故障，使系统更加稳定。

```
Full statespace search for:
        never claim             + (m)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness enabled)
        invalid end states      - (disabled by never claim)

State-vector 352 byte, depth reached 9999, errors: 1
    75600 states, stored (75602 visited)
 14746771 states, matched
 14816373 transitions (= visited+matched)
 53945412 atomic steps
hash conflicts:     19306 (resolved)
```

图 17　SPIN 环境下的验证结果

5　相关工作

为了提高嵌入式软件系统的安全性，针对软件模型的安全性分析与验证已经成为软件开发周期中的重要环节。文献[15]提出了一种使用详尽的转换步骤使故障树和状态图的语义信息得以保留的方法；给出了将故障树逻辑门转换到状态图标记的一系列转换规则，并利用这种带标记的状态图在需求验证的过程中故障条件发生来表现系统行为。文献[16]通过一种基于故障生成测试用例的方法，使用故障树分析确定系统不正确的状态，得到通过失效、守卫条件或交互影响系统行为的路径的测试模型。文献[17]首先构建了一张映射表，之后分析故障树基本事件和中间事件，按照各自的属性分别填写映射表相关内容，接着根据映射表将相应的故障信息扩展到系统功能模型中，最后利用模型检测的方法生成测试用例。文献[18]将故障树集成到基于状态的行为模型，系统地给出了与故障树相关的元素、单一关键基本事件、系统状态或事件序列到基于状态的行为模型的元素，由此产生的模型可以自动生成考虑了基于风险测试的测试用例。文献[19]指出由于故障树缺乏形式化语义，在应用的过程中可能导致不正确性、不一致性和二义性，因此提出应该将故障树分析应用到形式化模型当中。他们的方法是用形式化模型和故障树同时建模，但是构建两种模型的过程是相互影响和关联的。文献[20]通过将故障树信息扩展到状态图模型中，提出一种基于故障信息并利用故障树最小割集生成测试用例的方法。与已有工作相比，本文提出了一个基于故障扩展 SysML 活动图的模型检验框架，同时给出了故障扩展 SysML 活动图、最小割集故障域、功能域等概念，定义了故障树模型到活动图模型的转换步骤和逻辑门的转换规则、故障扩展 SysML 活动图到 Promela 模型的转换规则。利用文中方法得到故障扩展 SysML 活动图，并将其转化为 Promela 模型，利用模型检测工具 SPIN 对其进行安全性验证。

结束语　传统的 SysML 活动图通过对系统行为和执行过程的描述，能够判断系统设计是否满足功能需求，但是却不能对软件系统存在的故障问题进行有效分析。而故障树分析法虽然能够有效描述造成软件故障的因素以及因素之间的逻辑关系，却无法确定系统是否存在这样的故障。

针对以上问题，本文给出了利用故障树最小割集构造故障扩展 SysML 活动图的方法，统一了系统的安全需求分析模型和行为模型。故障扩展 SysML 活动图模型在保留 SysML 活动图对系统行为描述的基础上，增加了对故障逻辑的描述，使其能够同时描述系统的安全需求和功能行为，更加适用于安全验证工作。在故障扩展 SysML 活动图的基础上给出了一个安全性验证框架。首先将故障扩展 SysML 活动图各个元素转化为与其等价的 Promela 模型，然后将这些模型进行

重组，得到完整的 Promela 模型。在故障扩展 SysML 活动图模型中，若故障活动可达，说明行为模型不满足安全性需求，利用该模型可以迅速找到故障发生的过程和导致这个故障发生的软件行为，从而对行为模型进行修正，使系统功能模型满足安全性需求。最后，本文对一个燃气灶控制系统利用本文方法得到了带有故障信息的扩展 SysML 活动图以及与其等价的 Promela 程序，并利用 SPIN 对其进行安全性验证工作。

在下一步工作中，将考虑离散时间的转换；当前的映射表仍是人工完成，将考虑如何实现映射表的自动构建；利用模型驱动的方法，扩展 SysML 的元模型，用以包含从故障树中提取的故障信息，然后进行安全性分析验证。

参 考 文 献

[1] 黄志球，徐丙凤，阚双龙，等. 嵌入式机载软件安全性分析标准、方法及工具研究综述[J]. 软件学报，2014，25(2)：200-218
Huang Zhi-qiu, Xu Bing-feng, Kan Shuang-long, et al. Survey on Embedded Software Safety Analysis Standards, Methods and Tools for Airborne System [J]. Journal of Software, 2014, 25(2): 200-218

[2] Vesely W E, et al. Fault tree handbook[R]. Washington DC: U. S. Nuclear regulatory commission, 1981

[3] OMG. Unified Modeling Language: super structure v2.0 [EB/OL]. http://www.omg.org/docs/formal/05-07-04.pdf, 2005

[4] OMG. Systems Modeling Language. v1.2[EB/OL]. http://www.omg.org/spec/SysML/1.2, 2008

[5] 肖美华，薛锦云. 基于 SPIN/Promela 的并发系统验证[J]. 计算机科学，2004，31(8)：201-203
Xiao Mei-hua, Xue Jin-yun. Verification of Concurrent System Using SPIN/Promela[J]. Computer Science, 2004, 31(8): 201-203

[6] Walker M, Papadopoulos Y, et al. Qualitative temporal analysis: Towards a full implementation of the Fault Tree Handbook[J]. Control Engineering Practice, 2009, 17(10): 1115-1125

[7] Chu T L, Apostolakis G. Methods for probabilistic analysis of noncoherent fault trees[J]. IEEE Transactions on Reliability, 1980, 29(5): 354-360

[8] Schellhorn G, Thums A, Reif W. Formal fault tree semantics [C]// Proceedings of The Sixth World Conference on Integrated Design & Process Technology, Pasadena, CA. 2002

[9] Jarraya Y, Debbabi M, Bentahar J. On the meaning of SysML activity diagrams[C]// 16th Annual IEEE International Conference and Workshop on Engineering of Computer Based Systems (ECBS 2009). IEEE, 2009: 95-105

[10] 樊晓光，褚文奎，张凤鸣. 软件安全性研究综述[J]. 计算机科学，2011，38(5)：8-13
Fan Xiao-guang, Chu Wen-kui, Zhang Feng-ming. Surveys on Softwate safety[J]. Computer Science, 2011, 38(5): 8-13

[11] 薛克. 基于 SPIN 的 UML 活动图验证[D]. 上海：华东师范大学，2008
Xue Ke. Verification of UML ctivity chart using Spin [D]. Shanghai: East China normal university, 2008

[12] Guelfi N, Mammar A. A formal semantics of timed activity diagrams and its PROMELA translation[C]// 12th Asia-Pacific Software Engineering Conference (APSEC'05). IEEE, 2005: 8

[13] Ravn A P, Rischel H, et al. Specifying and verifying requirements of real-time systems[J]. IEEE Transactions on Software Engineering, 1993, 19(1): 41-55

[14] Lano K, Kan P, et al. Linking hazard analysis to formal specification and design in B[M]// Computer Safety, Reliability and Security. Springer Berlin Heidelberg, 1998: 60-74

[15] Ariss E O, Xu Dian-xiang, et al. Integrating safety analysis with functional modeling[J]. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 2011, 41(4): 610-624

[16] Sánchez M A, Felder M. A Systematic Approach to Generate Test Cases based on Faults[C]// ASSE 2003, Buenos Aires. 2003

[17] Hansen K M, Ravn A P, et al. From safety analysis to software requirements[J]. IEEE Transactions on Software Engineering, 1998, 24(7): 573-584

[18] Nazier R, Bauer T. Automated Risk-Based Testing by Integrating Safety Analysis Information into System Behavior Models [C]// 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2012: 213-218

[19] Reif W, Schellhorn G, et al. Safety analysis of a radio-based crossing control system using formal methods[C]// 9th IFAC Symposium Control in Transportation Systems. 2000

[20] Dasso A, Funes A, et al. Verification, validation and testing in software engineering[M]. IGI Global, 2007

(上接第 193 页)

[4] 陈杰，胡予濮，张跃宇. 用不可能差分法分析 17 轮 SMS4 算法[J]. 西安电子科技大学学报(自然科学版)，2008，35(3)：455-458
Chen Jie, Hu Yu-pu, Zhang Yue-yu. Impossible differential attack on the 17-round block cipher SMS4[J]. Journal of Xidian University(Natural Science), 2008, 35(3): 455-458

[5] Zhang L, Zhang W, Wu W. Cryptanalysis of Reduced-Round SMS4 Block cipher[C]// Proceedings of ACISP 2008. Springer-verlag, 2008, 5107: 216-229

[6] Kim T, Kim J, Hong S, et al. Linear and Differential Cryptanalysis of Reduced SMS4 Block Cipher[OL]. http://eprint.iacr.org/2008/281

[7] Km T, Kng J, Hong S, et al. Linear and differential cryptanalysis of reduced SMS4 block c-ipher[R]. Cryptology ePrint Archive: Report 2008/281, 2008

[8] 张美玲，刘景美，王新梅. 22-轮 SMS4 的差分分析[J]. 中山大学学报(自然科学版)，2010，49(2)：43-47
Zhang Mei-ling, Liu Jing-mei, Wang Xin-mei. Differential Attack on 22-Round SMS4 Block Cipher[J]. Acta Scientiarum Naturalium Universitatis Sunyatseni, 2010, 49(2): 43-47

[9] Biham E, Biryukov A, Shamir A. Cryptana lysis of Skipjack reduced to 31 rounds using impossible differentials[C]// Advances in Cryptology-Eurocrypt, 1999. Springer Berlin Heidelberg, 1999: 12-23