一种基于UML类图和活动图的故障树生成方法

徐 慧 燕雪峰 周 勇
(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 针对UML活动图在生成故障树的过程中只能反映活动事件流故障导致的系统失效,不能反映系统静态状态故障的问题,提出了一种活动图结合类图生成故障树的方法。在原有活动图的基础上,使用类图增加系统静态状态信息,设计活动图和类图到故障树模型的转换规则,将活动图中动态行为信息和类图静态状态信息转化为故障树中的节点要素。基于转换规则设计算法逆向遍历活动图和类图,自顶向下生成故障树。经过实例建模生成故障树,表明该方法能反映系统的动态行为和静态状态两方面的故障信息,为故障树生成提供了一种新的有效途径。

关键词 UML模型,活动图模型,类图模型,故障树
中图法分类号 TP391.9 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.7.033

Fault Tree Generation Method Based on UML Class Diagram and Activity Diagram

XU Hui YAN Xue-feng ZHOU Yong
(School of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China)

Abstract Aiming at the fault tree generated from UML activity diagram which can only reflect the behavior stream fault and can't reflect the static fault, a method was proposed by using activity diagram combined with class diagram. On the basis of the original activity diagram, class diagram is used to describe the system static state information, designing transformation rules from activity diagram and class diagram to the fault tree model to transform the activity diagram dynamic behavior information and class diagram static state information into fault node elements. Based on the transformation rules, the algorithm is designed to reversely traverse activity diagram and class diagram to top-down generate fault tree. Modeling to generate fault tree indicates that the fault tree generated by UML activity diagram combined with class diagram model can reflect the system behavior fault information and static state, providing a new effective way in the generating the fault tree.

Keywords UML model, UML activity diagram model, UML class diagram model, Fault tree

1 引言

现代科学技术的发展,促使了大量多功能的复杂系统的出现。复杂系统有软硬件结合和能执行多种功能任务活动的特性。大规模的复杂系统给人们带来了巨大的经济效益,但一旦出现故障则需要将故障及时排除,否则就会造成重大损失。FTA作为经典的安全分析方法,广泛应用于不同的领域[1-4],在系统实现过程中加强设备的可靠性建设以及在后期系统运用中实现故障的全面诊断、快速排除和维护具有重要作用。然而传统故障树是基于分析师的经验建立的,很难反映系统真正的设计情况,同时随着系统复杂度的提升,人工方式已无法构造和修改故障树[5]。UML模型作为软件工程领域中的标准建模语言,能够真实地描述软件系统的静态属性和动态属性[6,7]。基于UML模型的故障树生成技术为解决复杂故障树生成问题提供了一个有效的途径。

在现有的利用UML模型生成故障树的方法中,利用动态行为图生成故障树能够描述系统动态行为故障导致的系统失效[6],静态组成结构图生成故障树能够反映系统的静态属性故障导致的系统失效,大规模的复杂系统能执行多种功能任务活动,不同的功能任务活动由不同的软硬件组成,表现出的故障模式也不同。目前故障树的生成方法通常采用单一的动态行为图或静态组成结构图来生成故障树。文献[7]利用状态图和类图,根据设备状态的转换和端口连接关联到故障传播来生成故障树,因此只考虑静态属性故障。文献[8]提出利用活动图和用例图生成故障树,通过用例图对系统功能进行描述,设计用例图到故障树的转换规则,从功能层次上描述系统失效,但只考虑了系统的动态行为且未实现活动图到故障树的转换。文献[10]利用活动图描述系统功能任务的执行流程,设计活动图组成元素到故障结构的转换规则来实现故障树的自动生成,因此也考虑了系统的动态行为失效引起的故障。

活动图能显示从活动到活动的控制流[8],利用活动图生成故障树能够反映活动状态迁移过程中由动态行为导致的系统失效,但是不能描述系统内部静态状态故障。类图是UML

到稿日期:2015-07-16 返修日期:2015-11-03 本文受国防科工局十二五重大基础科研项目(c0420110005,NS2013091)资助。
徐 慧(1989-),女,硕士生,主要研究方向为系统建模与仿真,E-mail:xuhui0890@126.com;燕雪峰(1975-),男,副教授,主要研究方向为计算机网络、分布交互仿真等;周 勇(1975-),男,副教授,主要研究方向为人工智能、专家系统等。

建模中最常见的图，用于对系统静态设计视图建模，它可以描述系统中类的组成以及类之间的关系。针对活动图在生成故障树中的不足，本文提出了一种活动图结合类图生成故障树的方法。通过对系统建模得到系统的类图，在类图增加静态状态信息并设计类图和活动图中节点到故障树节点的转换规则。在故障树自动生成算法中，考虑每个活动节点所属的对象类可能出现的失效状态，从动态行为和静态状态两方面自顶向下生成故障树，以解决单一活动图生成故障树中缺少系统静态状态故障信息的不足。

2 类图模型和活动图模型

不同的应用活动包括不同的故障原因，将系统故障转化为故障树的结构，必须要考虑具体的应用信息，而活动图模型包含具体应用的活动状态信息。

一个活动图由许多元素组成，主要有〈泳道，动作，初始状态，转换，终止状态，结合，分支，判定，约束〉等。各元素常用的符号如表 1 和表 2 所列。

表 1 UML 活动图基本组成元素符号表

符号	泳道	动作	初始状态	转换	终止状态
名称					

表 2 UML 活动图基本组成元素符号表

符号	结合	分支	判定	约束
名称				[约束]

类图模型包含系统关键静态类的状态信息。复杂的大系统中包含许多静态类，每一个类都可能会因为一个或多个内部原因而导致状态失常。类的状态信息增加到属性〈state〉中，并且与具体的活动状态是独立的，类处于任一状态，都会导致该类失效。图 1 表示一个系统的类图，类图中包含 3 个类，在各个类的〈state〉属性中增加了该类的静态状态信息。

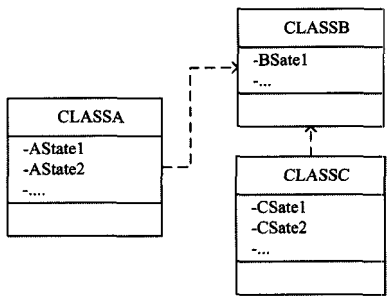

图 1 类图模型

图 2 表示一个系统应用活动图和类图模型，活动图显示活动状态的迁移过程和由泳道关联的类图静态状态信息。泳道 swimlane1 负责活动 activity1，泳道 swimlane2 负责活动 activity2 和activity3，泳道 swimlane3 负责活动 activity4。活动行为流为：活动 activity1 经分叉到达活动 activity2 和activity3，最后汇合到 ctivity4。活动图中的泳道关联出类图中的各个类状态信息，由 swimlane1 关联类图中 CLASS1，swimlane2 和 swimlane3 同理。

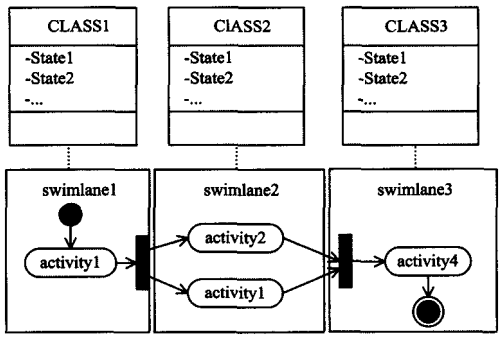

图 2 活动图模型和类图模型

2.1 活动图模型和类图模型到故障树模型的转换

故障树中故障可以分为两大类：1)动态行为故障，依赖于具体的应用服务活动；2)系统的静态类状态导致的故障，属于静态状态故障，独立于具体的应用活动。动态行为故障从活动图中转换，前序活动故障会导致活动故障，可以作为故障树结构中的顶事件和中间事件，初始活动点可以作为基本事件。静态状态故障从类图中提取，类的状态故障作为一个中间事件或门连接各失效状态，失效状态作为基本事件，该类型的故障不需要再进一步细化。

一个活动图由〈泳道，初始节点，转换，终止节点，结合，分支，判定，约束〉组合而成。在这里根据活动图的组成元素分类来制定规则(事件加上划线表示故障事件，基本事件用椭圆符号表示)。

规则 1(活动图泳道关联的类节点) 根据类名新增“类故障”节点作为中间事件或门连接各个失效状态，失效状态作为基本事件。泳道关联的类节点的转换如图 3 所示。

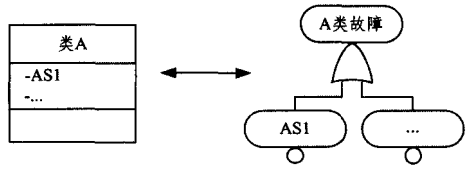

图 3 泳道关联的类节点的转换

规则 2(活动图的初始节点) 活动初始节点转化为故障树节点中的基本事件，后序节点转为中间事件或门连接初始节点的转换节点和类状态信息转换的故障树组合结构，类状态信息转换运用规则 1，转换结构如图 4 所示。

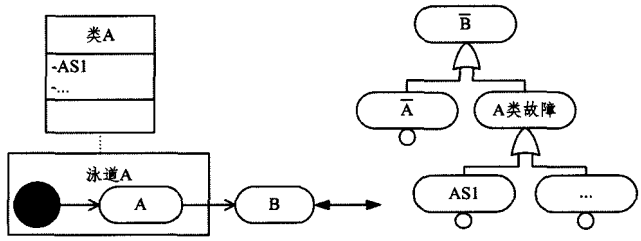

图 4 初始节点的转换

规则 3(活动图的终止节点) 终止节点转换为故障树节点中的中间事件或顶事件。A 作为活动图中的一个终止状态，转换后作为故障树节点的中间事件或顶事件，转换结构如图 5 所示。

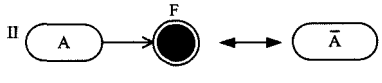

图 5 终止节点的转换

规则 4(活动图的转换结构) 转换结构的后序转换节点或门连接前序转换节点和类状态信息转换的故障树组合结构,类状态信息转换运用规则 1,转换结构如图 6 所示。

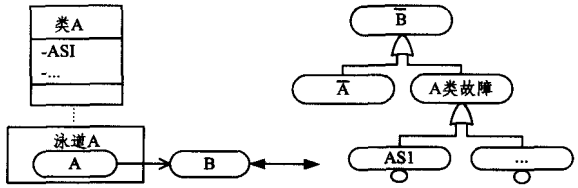

图 6 多前序输入的转换

规则 5(活动图的分叉结构) 分叉结构的后序转换节点或门连接前序转换节点和类状态信息转换的故障树组合结构,类状态信息转换运用规则 1,转换结构如图 7 所示。

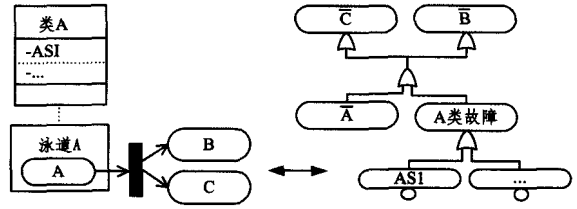

图 7 分叉的转换

规则 6(活动图的汇合结构) 汇合结构的每个前序转换节点和所属的类状态信息转换的故障树组合结构作为输入连接逻辑或门,后序转换节点或门连接所有的中间或门,转换结构如图 8 所示。

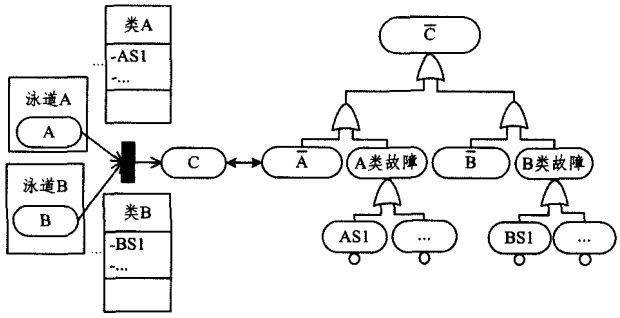

图 8 汇合的转换

规则 7(活动图的分支结构) 分支结构每个分支都有对应的约束条件,条件约束转换为故障树结构中的基本事件。每个后序转换节点或门连接前序转换节点、所属的类状态信息转换的故障树组合、对应约束条件转换节点,转换结构如图 9 所示。

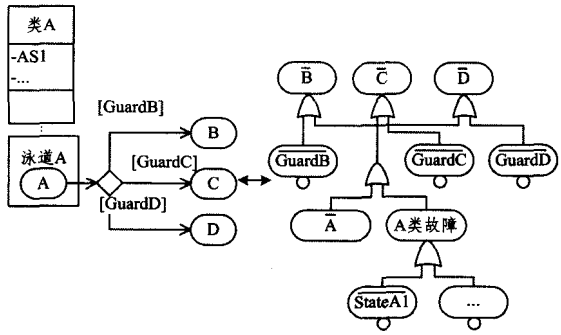

图 9 分支的转换

规则 8(活动图的合并结构) 合并结构的每个前序转换节点和所属的类状态信息转换的故障树组合结构作为输入连接逻辑或门,后序转换节点或门连接所有的中间或门,转换结构如图 10 所示。

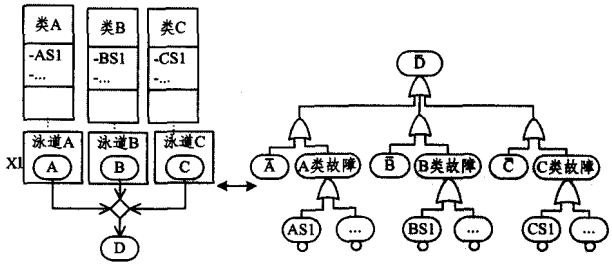

图 10 合并节点的转换

根据上述制定的活动图的基本组成元素,联合类图状态信息到故障树的转换规则,可以将活动图每个组成活动节点和节点所属类静态状态信息转换为故障树中的节点,从而能将一个活动图和类图转换为一棵完整的故障树。

2.2 UML 模型到故障树自动生成算法

在活动图和类图转换为故障树算法中,从活动图中选择期望得到活动对应的故障事件作为故障树的顶事件。判定活动事件的输入状态类型,从而得到输入状态连接的活动事件和活动事件关联的类信息,并结合转换规则进行活动图到故障树的转换,添加节点。自底向上遍历活动图到初始状态,依次连接每个节点生成故障树。

算法 1 活动图和类图故障树自动生成 CreateTree

输入:活动图 ArrayList〈ActivityNode〉AcList;
　　类图 ArrayList〈Classnode〉ClList;
　　顶事件对应的活动节点名 nodename
输出:故障树 FaultTree fautree
1. FaultTree fautree=new FaultTree;
2. Treeitem rootitem=fautree. addrootitem(nodename);
3. ActivityNode node=AcList. find(nodename);
4. Treeitem subtree=CreateSubTree(node,rootitem);
5. fautree. additem(rootitem,subtree);

算法 2 子树的生成 CreateSubTree

输入:活动图 ArrayList〈ActivityNode〉AcList;
　　类图 ArrayList〈Classnode〉ClList;
　　活动节点 ActivityNode node;
　　故障树父节点 Treeitem pareitem
输出:故障子树 Treeitem subtree
1. Treeitem subtree=new Treeitem;
2. If(node 类型为初始节点)
3.　　return subtree;
4. end if;
5. if(node 类型为…)
6. subtree=pareitem. addchilditem("OR");
7. end if;
8. For(node 的所有前序活动子节点名 name)
9.　ActivityNode childnode=AcList. find(name);
10.　Treeite childitem=subtree addchilditem("name");
11.　If(childnode 约束条件 guard 不为空)
12.　　subtree. addchilditem(guard);
13. End if;
14.　Classnode clasnode=ClList. find(childnod 所属类名 clname)
15. Treeitem classitem=subtree. addchilditem(clname,);
16. Treeitem classlink=classitem. addchilditem("OR");
17. For(clasnode 潜在的故障 classfault)
18. classlink. addchilditem (Classfault);
19. End For;
20. return CreateSubTree(AcList,ClList,childnode,childitem);

算法 1 中,根据输入活动图、类图顶事件生成故障树,调

用 CreateTree 创建整棵树的根节点，根据活动节点名在链表中找到该节点，然后调用 CreateSubTree 创建该根节点的子节点。

算法 2 中，根据传入的活动图、类图、活动节点、故障树父节点生成子故障树。步骤 1－7 为判断传入节点的前序节点的输入转换结构符合何种类型并创建逻辑门；步骤 8－13 为加入前序活动信息；步骤 11－13 加入约束条件信息；步骤 14－19 为加入类静态状态信息，对每一个活动所属的类状态信息遍历并添加到对应的故障树结构节点中；步骤 20 为递归调用 CreateSubTree 创建子故障树。该算法为递归函数，依据顶事件到初始节点的个数 n，最大分叉数为 K，该递归的最大时间复杂度为 $O(n\log_K n)$。

3 实验方案

区域防空网络化反导作战系统的作战使命主要是担任要地周边区域内的防空反导任务，也称区域反导作战。反导作战系统由雷达探测网、指挥控制网、拦截网组成。雷达探测网是获取目标信息，包括目标探测、目标跟踪。指挥控制网提供全面的空情态势图和火力协调，对软硬抗击武器进行综合运用以发挥整体防空能力。拦截网的任务是拦截行动执行，拦截作战单元对目标的打击。

根据对系统组成的总体描述，可以将整个作战过程分成 3 个功能模块，模块之间相互联系，相互制约，通过控制网络的连接，形成信息一体化的作战体系。整个系统的结构如图 11 所示。

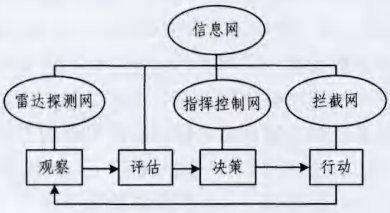

图 11　反导作战系统结构

3.1 反导作战系统活动图和类图

根据作战系统结构之间的关系图（见图 11）以及反导作战的基本过程，可以得到反导作战过程涉及的类图和作战活动图，具体的类图和活动图如图 12 所示。

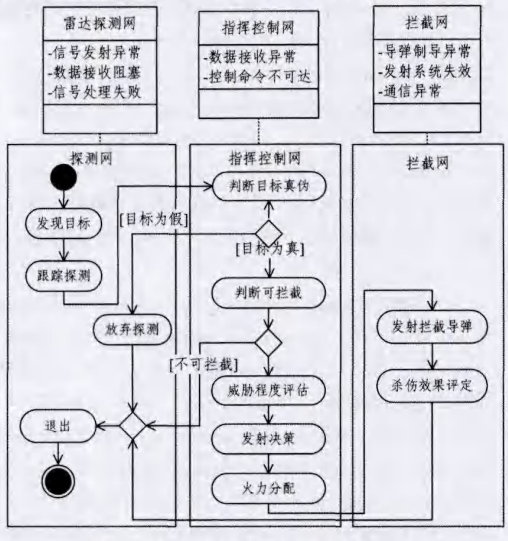

图 12　反导作战的 UML 类图和活动图

3.2 基于反导作战活动图和类图生成故障树

此案例选择的期望活动为“判断可拦截”，则生成的故障树对应的顶事件是“判断可拦截故障”。“判断可拦截”的前序活动为“判断目标真伪”且控制结构为分支节点，根据转换规则 7，选择故障类型由“目标为真判断失误”作为基本事件、“判断目标真伪故障”作为中间事件，“指挥控制网故障”或门连接而成，将指挥控制网的静态状态信息或门连接在“指挥控制网故障”。再根据前序节点自顶向下递归生成故障树。生成的故障树如图 13 所示。

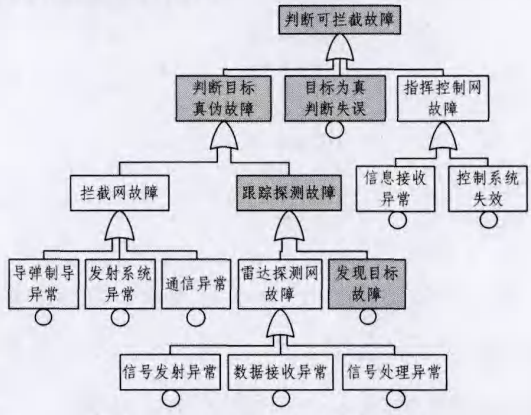

图 13　基于活动图和类图生成的故障树

图 13 中完整的故障树根据活动图和类图结合生成，阴影部分故障树仅根据活动图生成。

3.3 结果分析

对生成的故障树进行定性分析，求解故障树的割集。图 13 根据活动图和类图生成的故障树割集 $T1$ 为：

$T1$＝{信号处理异常，数据接收异常，信号发射异常，导弹制导异常，发射系统异常，通信异常，发现目标故障，信息接收异常，控制系统失效，目标为真判断失误}

同理，仅根据活动图生成的故障树（图 13 中阴影部分）对应的割集求 $T2$ 为：

$T2$＝{发现目标故障，目标为真判断失误}

比对 $T1$，$T2$ 结果可知，$T2 \in T1$。$T1$ 包含更全面的故障事件集合。

对生成的故障树进行定性分析求得割集后，可以对故障树进行定量分析，定量分析的目的是计算或近似估计顶事件的发生概率以及底事件重要度。将计算的顶事件发生概率与预定的目标值进行比较，如果超出目标值就应该采取必要的改进措施。重要度有助于有效地改进系统可靠度，对系统的运行提供有效的设计方法或寻找系统的失效原因等。

生成的故障树中各底事件发生的概率如表 3 所列。

表 3　故障树中各底事件发生概率

代号	底事件	发生概率（$\times 10^{-6}$）
X1	导弹制导异常	3.18
X2	发射系统异常	1000
X3	通信异常	12.5
X4	发现目标故障	20.7978
X5	信号发射异常	0.480
X6	数据接收异常	45.359
X7	信号处理异常	38.103
X8	目标为真判断失误	0.48
X9	信息接收异常	27.189
X10	控制系统失效	0.481

根据底事件的发生概率可以计算出顶事件的发生概率为

$P(T1)=\sum_{i=1}^{10}P(K_i)=1.148\times10^{-3}$，$P(T2)=P(k_4)+P(k_8)=2.128\times10^{-5}$。

概率重要度分析是故障树分析中的重要部分，反映了底事件概率变化对顶事件概率变化的难易程度。设 $t=1000$h，λ 为各底事件的发生概率，则可得各底事件的可靠度 $R_i(t)=e^{-\lambda t}$。各底事件发生的概率如表 4 所列。

表 4　故障树中各底事件的可靠度

代号	底事件	底事件可靠度
X1	导弹制导异常	0.997
X2	发射系统异常	0.368
X3	通信异常	0.987
X4	发现目标故障	0.979
X5	信号发射异常	0.999
X6	数据接收异常	0.955
X7	信号处理异常	0.962
X8	目标为真判断失误	0.999
X9	信息接收异常	0.973
X10	控制系统失效	0.999

当 $t=1000$h，$F_i(t)=1-R_i(t)$ 时，概率重要度的计算公式为 $\Delta g_i(t)=\frac{\partial P_s(t)}{\partial F_i(t)}$，其中 $P_s(t)=1-\sum_{i=1}^{10}(1-F_i(t))$。各底事件的概率重要度为 $\Delta g_i(t)=\frac{\partial P_s(t)}{\partial F_i(t)}$，如表 5 所列。

表 5　故障树中各底事件的概率重要度

代号	底事件	概率重要度
X1	导弹制导异常	0.221
X2	发射系统异常	0.850
X3	通信异常	0.231
X4	发现目标故障	0.239
X5	信号发射异常	0.219
X6	数据接收异常	0.263
X7	信号处理异常	0.256
X8	目标为真判断失误	0.219
X9	信息接收异常	0.245
X10	控制系统失效	0.219

通过分析表 5 可知“发射系统异常”概率重要度最大，因而在系统使用过程中，该事件出现异常对顶事件的影响程度最大。设计人员在设计过程中应该采取必要的检测手段和保护措施来提高其可靠性和安全性。

活动图和类图能反映系统动态属性和静态属性，使得生成的故障树相对较完整。对生成的故障树进行定性和定量分析，得到顶事件的失效概率和各底事件的概率重要度，便于在系统的设计过程中采取措施提高其可靠性，在使用过程中实现快速故障诊断。

4　对比分析

将本文故障树构建方法与其他故障树构建方法进行对比。文献[6]利用 SysML 组件图生成故障树，但只考虑了静态故障，生成的故障树不够完整。文献[7]利用状态图和类图，根据设备端口连接关联到故障传播来生成故障。文献[8]利用用例图和活动图，并设计用例图到故障树的转化规则生成故障树。文献[9]利用顺序图分析故障模式，人工建立故障树。文献[10]利用活动图描述系统功能任务的执行流程，设计活动图组成元素到故障结构的转换规则来实现故障树的自动生成。文献[11]提出利用 UML 顺序图、用例图和结构图生成故障树，输入 UML 结构和行为的信息，利用混合的转换语言 ATL 实现故障树的生成。文献[12]利用 Altarica 模型生成故障树，Altarica 模型在设计中自带故障信息，没有转换规则，还需要人工确立故障树。对比结果如表 6 所列。

表 6　生成故障树方法的对比

文献	原模型	缺陷
[6]	SysML 组件图	没有定义转换规则
[7]	UML 类图和状态图	没有定义转化规则
[8]	UML 用例图和活动图	活动图没有转化规则，得到的故障树不完整
[9]	UML 顺序图	只考虑动态行为，半自动转化，且有冗余
[10]	UML 活动图	没有完整的转换规则，只考虑动态行为故障树
[11]	UML 顺序图，用例图，结构图	转换语言的实现比较复杂
[12]	Altarica 模型	没有转换规则，需要手动配置实现故障树的生成

通过比较可以看出，本文结合 UML 类图和活动图，从系统的静态和动态两方面分析潜在的系统故障，依据活动图和类图提取动态和静态状态故障逻辑信息，提出了完整的转换规则和故障树生成算法，实现了故障树的自动生成，同其他方法相比在自动生成故障树和描述系统的动态行为故障和静态状态故障上具有一定优越性。

结束语　本文考虑到复杂系统同时能执行多种功能的任务活动，在此基础上提出针对具体的应用活动，并结合具体应用活动关联的系统静态类状态信息生成故障树的方法。在系统设计的初始阶段，对系统建模分析构建具体应用活动的活动图和类图，从活动图的活动流中提取故障逻辑信息，并结合类图提取系统静态类的内部状态信息自动生成故障树。所提方法能够有效地解决仅依据动态结构或静态结构生成故障树的方法中只能反映系统动态属性和静态属性的不足。实验结果验证了所提方法的可行性，其为基于 UML 模型故障树自动生成和安全分析提供了一种新的有效途径。

参 考 文 献

[1] Xu Bing-feng, Huang Zhi-qiu, Hu Jun, et al. Time Property Analysis Method for State/Event Fault Tree[J]. Journal of Software, 2015, 26(2): 427-446(in Chinese)
徐丙凤，黄志球，胡军，等. 一种状态事件故障树的时间特性分析方法[J]. 软件学报，2015，26(2)：427-446

[2] Zhang Hong-lin, Zhang Chun-yuan, Liu Dong. An Identification Method of Independent Module Applying to Dynamic Fault Tree with Interdependent Basic Events and Repeated Events[J]. Chinese Journal of Computers, 2012, 35(2): 229-243(in Chinese)
张红林，张春元，刘东. 一种适用于具有相互依赖基本事件和重复事件的动态故障树独立模块识别方法[J]. 计算机学报，2012，35(2)：229-243

[3] Nguyen T P K, Beugin J, Marais J. Method for evaluating an extended Fault Tree to analyse the dependability of complex systems: Application to a satellite-based railway system[J]. Reliability Engineering & System Safety, 2015, 133: 300-313

[4] Bechta Dugan J, Sullivan K J, Coppit D. Developing a low-cost high-quality software tool for dynamic fault-tree analysis[J]. IEEE Transactions on Reliability, 2000, 49(1): 49-59

[5] Domis D, Trapp M. Integrating safety analyses and component-based design[M]// Computer Safety, Reliability, and Security.

Springer Berlin Heidelberg,2008:58-71

[6] Xiang J,Yanoo K,Maeno Y,et al. Automatic synthesis of static fault trees from system models[C]// 2011 Fifth International Conference on Secure Software Integration and Reliability Improvement (SSIRI). IEEE,2011:127-136

[7] Lauer C,German R,Pollmer J. Fault tree synthesis from UML models for reliability analysis at early design stages[J]. Acm Sigsoft Software Engineering Notes,2011,36(1):1-8

[8] Hu W,Deng Z,Hong Y. A method of FTA base on UML use case diagram[C]// 2011 9th International Conference on IEEE Reliability,Maintainability and Safety (ICRMS). 2011:757-759

[9] Harper D C. Fault Tree Analysis of UML Designs[J]. Technometrics,2012,19(3):346-347

[10] Tiwari S,Gupta A. An Approach to Generate Safety Validation Test Cases from UML Activity Diagram[C]// 2013 20th Asia-Pacific Software Engineering Conference. IEEE,2013:189-198

[11] Zhao Z. UML Model to Fault Tree Model Transformation for Dependability Analysis[D]. Carleton University Ottawa,2014

[12] Li S, Li X. Study on generation of fault trees from Altarica models[J]. Procedia Engineering,2014,80:140-152

(上接第 156 页)

除存储空间外,SEKM 机制还具有其他显著的性能优势,如计算和通信负载少、支持 SN 移动、抵御 CH 被俘、基于签密确保 Sink-CH 之间的安全通信等。将 SEKM 机制与 SACK、RDEC、PKAS 和 PIBK 机制进行了比较,如表 3 所列。PKAS 与 SEKM 机制具有最少的缺点,RDEC、PKAS 和 PIBK 机制中,SN-CH 之间的链路使用 PKC 机制,这种方式增加了 SN 的计算量,且 SEKM 机制和 RDEC 机制不需要事先分簇。

表 3 5 种机制性能的对比

特点	机制				
	SACK	RDEC	PKAS	PIBK	SEKM
节点移动性	无	无	无	无	有
SN 中密钥数	3	2	3	4	3
单个 SN 被俘	网络失效	多个 SN 失效	仅单个 SN 失效	整个网络失效	仅单个 SN 失效
SN 与 CL 差别	无	有	有	无	有
PKC 使用形式	加密	加密和签名	加密	密钥协商	签密
PKC 使用范围	CH-Sink	SN-CH CH-Sink	SN-CH CH-Sink	SN-CH CH-Sink	CH-Sink
可扩展性	支持	支持	支持	支持	支持
密钥管理基于分簇	是	否	是	是	否
簇密钥更新效率	低	无	无	低	高

结束语 密钥管理是 WSN 安全的基础。近年来,针对非对等无线传感器网络已设计了一些密钥管理机制。本文提出一种新颖、高效和安全的混合密钥管理机制 SEKM。SEKM 机制在安全性和性能方面具有显著优势和特征,包括在 CH 和 Sink 之间使用基于椭圆曲线密码的签密机制确保数据传输的安全性,提出的签密机制具有前向安全性和公开验证特点,并且协议开销小。针对 SN 的加入、退出和移动,提出轻量级的簇密钥生成和更新算法,不仅资源和能量消耗少,而且基于簇基密钥方式可以有效避免因簇首被俘而导致簇中所有 SN 节点甚至整个网络失效。实际上,SEKM 不仅具备较好的安全属性,而且计算、通信和存储开销较小。

参 考 文 献

[1] Zeng Wei-ni, Lin Ya-ping, Yu Jian-ping, et al. Group Key Management Based on Random Perturbation in Wireless Sensor Networks[J]. Journal of Software,2013,24(4):873-886(in Chinese)
曾玮妮,林亚平,余建平,等. 传感器网络中基于随机混淆的组密钥管理机制[J]. 软件学报,2013,24(4):873-886

[2] Guo Song-hui, Niu Xiao-peng, Wang Yu-long. Elliptic Curve Based Light-weight Authentication and Key Agreement Scheme [J]. Computer Science,2015,42(1):137-141(in Chinese)
郭松辉,牛小鹏,王玉龙. 一种基于椭圆曲线的轻量级身份认证及密钥协商方案[J]. 计算机科学,2015,42(1):137-141

[3] Wang Gang,Wen Tao,Guo Quan,et al. An Efficient and Secure Group Key Management Scheme in Mobile Ad Hoc Networks [J]. Journal of Computer Research and Development,2010,47(5):911-920(in Chinese)
王刚,温涛,郭权,等. 移动自组网中安全高效的组密钥管理方案[J]. 计算机研究与发展,2010,47(5):911-920

[4] Lee J,Kapitanova K,Son S H. The price of security in wireless sensor networks [J]. Computing Network Journal, Elsevier, 2010,54(17):2967-2978

[5] Oliveira L B, Aranha D F, Gouvea C P L, et al. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks [J]. Computing Communication Journal,Elsevier,2011,34(3):485-493

[6] Hagras E A,El-Saied D,Aly H H. Energy efficient key management scheme based on elliptic curve signcryption for wireless sensor networks [C]// 28th National Radio Science Conf(NRSC). 2011

[7] Riaz R, Naureen A, Akram A, et al. A unified security framework with three key management schemes for wireless sensor networks[J]. Computing Communication Journal,Elsevier,2008 (31):4269-4280

[8] Du X,Guizani M,Xiao Y,et al. A routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks [J]. IEEE Transactions Wireless Communication,2009,8(3):1223-1229

[9] Mizanur R,Sk M,El-Khatib K. Private key agreement and secure communication for heterogeneous sensor networks [J]. Parallel Distribution Computing,Elsevier,2010(70):858-870

[10] Khaliq-ur-Rahman Raazi S M,Lee H,Lee S,et al. MUQAMI+: A scalable and locally distributed key management scheme for clustered sensor networks [J]. Annalso of Telecommunications, Springer,2010,65(1/2):101-116

[11] Stavrou E,Pitsillides A. A survey on secure multipath routing protocols in WSNs [J]. Computing Network Journal,2010(54): 2215-2238

[12] Alagheband M R,Aref M R. Dynamic and secure key management model for hierarchical heterogeneous sensor networks [J]. IET Information Security,2012,6(4):271-280

[13] Zhan Guo-xing,Shi Wei-song,Deng Ju-lia. Design and implementation of TARFA trust-aware routing framework for WSNs [J]. IEEE Transactions on Dependable and Secure Computing, 2012,9(2):184-197