ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.2, February 2010, pp.287−304
doi: 10.3724/SP.J.1001.2010.03792

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种从UML模型到可靠性分析模型的转换方法*

柳 毅[1,2], 麻志毅[1,2+], 何 啸[1,2], 邵维忠[1,2]

[1](北京大学 信息科学技术学院 软件研究所,北京 100871)
[2](高可信软件技术教育部重点实验室(北京大学),北京 100871)

Approach to Transforming UML Model to Reliability Analysis Model

LIU Yi[1,2], MA Zhi-Yi[1,2+], HE Xiao[1,2], SHAO Wei-Zhong[1,2]

[1](Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)
[2](Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing 100871, China)
+ Corresponding author: E-mail: mzy@sei.pku.edu.cn

Liu Y, Ma ZY, He X, Shao WZ. Approach to transforming UML model to reliability analysis model. *Journal of Software*, 2010,21(2):287−304. http://www.jos.org.cn/1000-9825/3792.htm

Abstract: In the context of component-based software development, this paper proposes an approach to transforming UML diagrams of software architecture to Markov chain for the quantitative evaluation of reliability. Based on the component-based software architecture, it utilizes four types of UML diagrams: use case, sequence, activity and component diagrams, extending them and annotating them with reliability related attributes. Then, the diagrams are transformed into a Markov chain based analysis model by constructing an intermediate model called Component Transition Graph (CTG). Result of this transformation can be directly used in the existing analysis methods to predict software reliability, which facilitates the analysis task of software designer.

Key words: CBSD (component-based software development); software reliability; UML; model transformation; Markov chain

摘 要: 以构件化的软件开发方法为背景,提出了一种将UML模型自动地转换为可靠性分析模型Markov链的方法.该方法基于构件化的软件体系结构,从UML的用况图、顺序图、活动图和构件图出发,对其进行扩展,在模型中标注了可靠性分析所需的信息.在此基础上,通过构造一个称为构件转移图的中间模型,将标注了可靠性信息的UML模型转换为Markov链.该方法产生的结果能够直接作为现有可靠性相关的数学分析方法的输入,从而使可靠性分析工作变得更加方便、高效.

关键词: 基于构件的软件开发;软件可靠性;UML;模型转换;Markov链
中图法分类号: TP311 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.60773152 (国家自然科学基金); the National Basic Research Program of China under Grant No.2005CB321805 (国家重点基础研究发展计划(973)); the National High-Tech Research and Development Plan of China under Grant Nos.2007AA01Z127, 2007AA010301 (国家高技术研究发展计划(863)); the Fund for Creative Research Groups of China under Grant No.60821003 (国家创新研究群体科学基金)

Received 2009-06-15; Revised 2009-09-11; Accepted 2009-12-07

随着计算机技术的发展,软件系统已经广泛地应用到社会中的各个方面,在人们的工作和生活中发挥着举足轻重的作用.系统的可靠性也越来越多地受到人们的关注.可靠性是软件质量的一个量化指标,它通常被定义为在特定环境、特定时间下系统无故障运行的概率[1].在传统的研究工作中,软件可靠性的分析和度量通常在软件开发实现完成后再进行,这种方法得到的结果一般比较精确.但如果在此阶段才发现系统未满足可靠性需求,修改的代价将十分巨大.因此,一个有效的设计开发过程必须在早期,如体系结构设计阶段,就包含对系统可靠性的分析和预测,以指导软件开发过程,确保最终构造出来的系统满足可靠性需求[2].在软件开发早期对可靠性进行分析,一方面能够帮助软件设计人员对现有的设计方案进行评估,以检验设计是否满足用户需求;另一方面也能帮助设计人员对各种设计决策进行比较,从而选择更优的设计方案.因此,在软件开发早期对系统的可靠性进行预测评估对于构造满足用户可靠性需求的软件具有重要的意义.

模型是软件开发早期最主要的制品.目前,常常用统一建模语言UML来建立系统的模型[3−5],如用用况图表示系统的需求,用类图表示系统的静态结构,用活动图、顺序图、状态机图等表示系统行为等.这些模型从各个角度对系统进行了描述,是早期对可靠性进行分析的一个重要输入.然而,对可靠性进行分析首先需要根据系统的特征建立相应的数学模型,如Markov链、Petri网、故障树等[1,6,7],再根据这些模型的数学特性去分析系统的可靠性.这些数学分析模型与系统的设计模型在语法语义以及表示法等方面都存在着较大的差异,需要软件设计人员具有相关的数学背景才能完成建立可靠性分析模型的工作,这就提高了对软件设计人员的要求,并且加大了他们的工作量.因此,如果能够直接从软件的设计模型出发,将其自动转换为可靠性分析模型,则能大大减轻软件设计人员的负担,使他们专注于软件设计以及其中存在的问题,简化可靠性分析的复杂度.

在基于构件化软件体系结构的可靠性分析工作中,Markov链是最常用、最有效的一种分析模型.它通过构造系统中各构件间的交互和转移发生的情况来对整个系统的可靠性进行分析.基于上述背景,本文提出了一种从UML模型转换为Markov链的方法,使得从设计模型构造分析模型的过程能够自动地进行.该方法主要针对基于构件开发的软件,其产生的结果能够直接作为数学分析方法的输入,从而对系统可靠性进行分析.其主要内容包括两个方面:首先,基于现有的可靠性建模方法[8,9],在用况图、顺序图、活动图和构件图中加入了可靠性分析中需要的信息;然后以这些模型作为输入,利用基于QVT(query/view/transformation)的模型转换方法将其转换为相应的分析模型,即Markov链.

本文第 1 节介绍标注可靠性信息后的 UML 模型并给出一个实例.第 2 节是本文方法概览,对问题进行分析并给出转换方法的框架.第 3 节和第 4 节对方法和转换的步骤进行详细的介绍,并以实际系统为例展示每个步骤的结果.第 5 节介绍对并发情况的转换处理方法.第 6 节介绍相关工作.最后对本文工作进行总结和展望.

1 标注可靠性信息后的 UML 模型

UML 关注的重点是功能方面的建模,为了对系统可靠性进行分析,需要在 UML 模型中加入可靠性相关的信息.根据面向用户的可靠性的思想,可靠性与系统被如何使用密切相关,整个系统的可靠性依赖于组成系统的单个构件的可靠性、构件间的交互情况及执行环境.为了对可靠性进行分析,至少需要以下 4 个方面的信息:一是用户使用系统的情况,即系统中每个功能被使用的概率;二是组成系统的各个构件的可靠性;三是构件间如何交互,控制流在两个构件间发生转移时,转移的概率是多少;四是软件运行环境的可靠性.本文中假设运行环境是可靠的,致力于计算软件本身的可靠性.

在UML提供的模型图中,用况图是用来描述用户和系统间的交互情况的.本文假设一个用况对应系统的一个功能,通过用况图反映系统被使用的情况.顺序图和活动图都是系统的行为视图.本文中,顺序图用来表示为完成某项功能,系统的构件与构件间的交互情况;活动图表示为完成系统中的某项功能,动作和动作间的交互情况,其中每个动作都由相应的构件来完成.因此,可以通过顺序图和活动图中构件或动作间的交互情况,以及构件间发送消息的概率或动作转移的概率来构造系统中构件间的交互情况;对于构件本身的信息,则可以通过构件图进行表示.因此,本文主要是从 UML 的用况图、构件图、活动图或顺序图出发,在这些图中加上可靠性相关的信息,利用这些图提供的信息,自动地构造可靠性分析模型 Markov 链.

目前已有很多工作研究如何将非功能属性加入到原有模型中,如文献[8]提出了一个对基于构件系统的可靠性进行建模的 UML 外廓(profile),文献[9]提出了一个对可信性建模的 UML 外廓.本文基于这些工作并对其进行扩展,在用况图、活动图、顺序图和构件图中标注了可靠性相关的信息,作为转换方法的输入.下面以一个简化的指纹门禁控制系统为例,分别对其进行说明.

(1) 用况图.一个用况通常与系统的一个功能相对应.用况图中的标注主要用来表示每个功能被使用的概率.如图 1 所示,门禁系统简化后的功能主要有两个,一是指纹录入,将数据存入数据库系统,如对公司访客的指纹进行记录并备案,或者公司新进员工的指纹数据录入等;二是识别指纹并开门.其中,每个用况发生的概率通过用户使用系统的比率以及用户在使用系统的前提下使用该用况的概率计算得到.假设第 i 类 Actor 使用系统的比率 p_{A_i},在使用系统的前提下,使用某个功能的概率 p_{ij},于是可得到每一个用况发生的概率为 $p_j = \sum_{i=0}^{m}(p_{A_i} \times p_{ij})$[10].根据公式计算得出指纹录入用况的发生概率为 0.18,识别指纹并开门用况发生的概率为 0.82.当然,如果一个用况(功能)被使用的概率是已知的,那么直接对其进行标注即可.注意到所有类型参与者的使用比率之和为 1,每一类参与者使用各个用况的概率之和为 1,并且所有用况发生的概率之和也为 1.这些概率通常来源于专家知识或功能相似的系统.

(2) 构件图.构件图主要用来表示组成系统的所有构件以及各个构件自身的可靠性.如图 2 所示,系统主要由系统控制、指纹管理、数据库存取和电子锁控制这 4 个构件组成.每个构件的可靠性都在图中进行了标注,如指纹管理构件的可靠性为 0.85,电子锁控制构件的可靠性为 0.9 等.这些可靠性信息可能来源于构件本身的描述文档、专家知识、功能相似的构件等,对于自己开发的构件,也可以通过现有方法[11]进行计算,本文假设这些信息都是已知的.除此之外,构件与构件间的连接子也有可能发生失效.但在实际的系统中,连接子最终可能被实现为一个构件的形式.本文假设图中的连接子只表示简单的依赖关系,可靠性为 1.0.

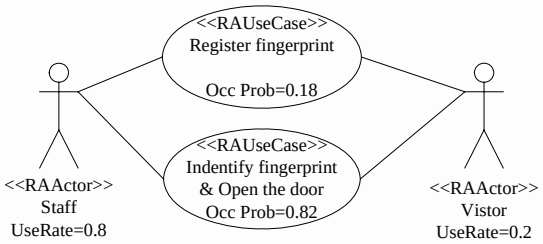


Fig.1 Annotated use case diagram

图 1 标注后的用况图

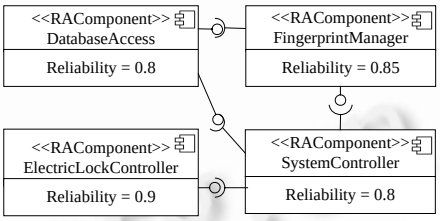


Fig.2 Annotated component diagram

图 2 标注后的构件图

(3) 活动图.如图 3 所示,该活动图描述了在完成识别指纹并开门的用况对应的行为时,系统中各个动作与动作之间的转移关系.在基于构件开发的系统中,每个构件都会向外提供一定的接口和操作,活动图中的动作一般都与这些接口或操作相对应,由相关的构件来执行,因此活动图中标明了每一个动作(除开始结点和终止结点外)对应的构件(对于需要由多个构件共同完成的活动结点,将其分解为粒度更小的活动,直至每个结点由 1 个构件执行),如“识别指纹”对应的构件为“指纹管理构件”,“电子锁打开”对应的构件为“电子锁控制构件”等.活动图中的分支结点表明活动执行过程中不同的路径,它通常与用况执行时发生的不同场景对应,因此,为分支结点的每条出边标上一个概率,表明在当前活动发生的条件下该分支发生的概率.该概率与对应场景发生的概率相同.如上述活动图中“判断指纹是否存在”时可能会发生指纹存在或不存在的情况,它们分别对应了识别指纹和开门用况的两种不同的场景,发生的概率分别为 0.9 和 0.1.该值可以来源于功能相似的系统或专家知识等.

(4) 顺序图.如图 4 所示,在本文中,顺序图用来描述为完成某个用况所对应的功能时,构件与构件间的交互情况.即生命线与某个构件相对应,其中的操作表示构件向外提供的服务或接口,操作之间消息的传递表示构件间的交互情况.与活动图中的分支结点类似,图中的组合片段 Alt 表明顺序图执行的不同路径,它与用况执行时发生的不同场景相对应,根据场景的发生概率标明每个交互片段发生的概率.如图中指纹数据成功录入的概率

为 0.8,由于数据已在数据库中存在而导致录入信息失败的概率为 0.2.同样地,这些信息主要也是来源于专家知识、功能相似的系统或模型模拟等.

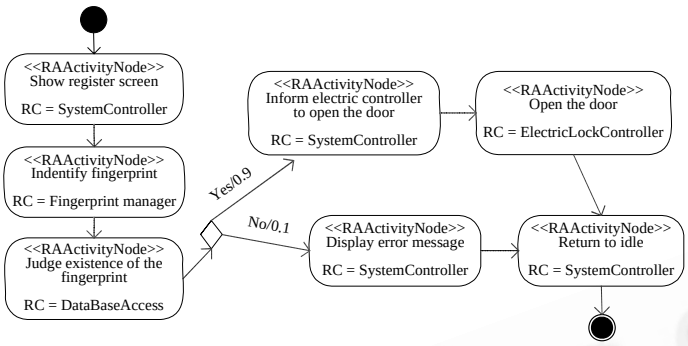


Fig.3 Annotated activity diagram

图 3 标注后的活动图

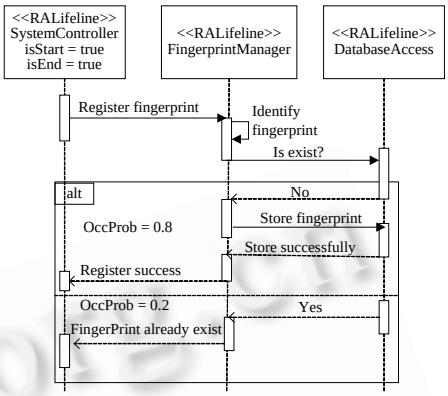


Fig.4 Annotated sequence diagram

图 4 标注后的顺序图

通过上述分析可看出,用况图描述了系统具有的功能,其中每个用况与一个功能对应,且都有自己的发生概率.每个用况对应的功能都可用一个活动图或顺序图来进行描述,有时,由于对用况描述的粒度不同,一个用况可能会有多个对应的活动图或顺序图,对于这种情况,本文需要在转换前将这些图合并,形成一个整体的活动图或顺序图.具体的模型合并的方法目前已有很多研究工作,如文献[12–14]等,本文参考这些工作中提出的方法对多个顺序图或活动图的情况进行合并.对于一个用况既有活动图又有顺序图的情况,本文假设二者间是一致的[15],这样转换得到的结果就不会出现不一致的情况.活动图中的分支结点或顺序图中的Alt片段对应了用况的不同场景,分支概率或交互片段的概率则表明了场景发生的概率.所有需要标注的信息在早期主要来源于专家知识、功能相似的系统、构建描述文档、模型模拟等,随着软件开发过程的进一步进行,将会获取到越来越精确的数据.本文的转换方法就是以这些模型图及标注信息作为输入,通过它们提供的信息来构造基于Markov链的可靠性分析模型.

2 方法概览

2.1 问题分析

面向用户的可靠性分析的思想[16]把软件可靠性看作是一种对软件提供给用户的服务质量的度量.一方面,软件可靠性与软件如何被使用是密切相关的,只有当用户使用的功能执行到导致系统发生失效的代码时,才会对外界显现一个失效.如果在执行某个功能时没有执行这段代码,则从用户的角度来讲,系统是可靠的.另一方面,对于基于构件开发的系统,软件可以看作是由多个构件组成,软件的可靠性依赖于单个构件的可靠性、构件间的交互情况及执行环境.系统被使用的情况主要是指用户使用系统提供的某项功能的情况,如概率.因此,可以通过加入标注信息后的用况图去获取这些信息.对于构件间的交互情况,可以通过顺序图或活动图去构造,其中交互发生的概率可以结合各用况发生的概率、活动图中的分支概率或顺序图中交互片段发生的概率得到.构件自身的可靠性信息可以从构件图中获取.对于软件的执行环境,本文先假设它们是可靠的,以致力于解决软件本身的可靠性问题,转换过程中暂不考虑执行环境的可靠性.

可靠性分析模型Markov链[16,17]主要由状态和转移两类元素组成,其中每个状态与一个构件相对应,当程序运行到某个构件中时,就认为系统处于该构件对应的状态.状态间的转移与构件间的转移相对应.当程序从一个构件运行到另一个构件时,就看作是发生了一次从相应的状态到另一个状态的转移.每条转移都被赋予了一个概率,表明其发生的可能性,这个概率与构件本身的可靠性是有关的,构件的可靠性越低,转移概率越小.除此之

外,还包含一个失效状态F或一个正确状态C,每一个状态都有一条到达F状态的概率为 $1-R_i$的转移(其中R_i为该状态对应的构件的可靠性),表示每个构件的失效都有可能导致系统到达失效状态F.对于正确状态C,只有结束状态能够到达,表示只有系统成功达到结束状态后才正确执行了相应的功能.有了这些信息后,利用Markov链相关的数学分析方法或工具,就能对系统的可靠性进行分析.

由上述分析可以看出,从标注有可靠性相关信息的用况图、顺序图、活动图和构件图出发,将其转换为可靠性分析模型 Markov 链,关键的一步是构造出系统中构件间的交互情况,然后再根据交互情况及构件自身的可靠性构造 Markov 链.为使得转换的过程更加简捷、高效,本文提出了一种称为构件转移图的中间模型,作为 UML 模型和 Markov 链间转换的桥梁.该模型由构件和构件间的转移组成,每条转移都标注有发生概率,与 Markov 链中的转移概率不同,构件转移图中的概率是假设系统中所有构件都正常运行时从起始构件转移到目标构件的概率,而不考虑构件失效带来的影响.加入构件转移图后,整个转换过程变成从 UML 模型转换为构件转移图,再从构件转移图转换为 Markov 链的过程.

对每一个活动图和顺序图都可以构造一个对应的构件转移图,但由于它们只与系统的一个功能相对应,其中的交互可能只有一部分构件参与,所以这些构件转移图只表示了系统中部分构件转移的情况,本文称其为局部构件转移图.其中各个转移概率表示在活动图或顺序图所对应的行为发生的条件下活动边或消息发生的概率,它们表示的是一个局部的交互信息.这些概率通过活动图中的分支概率或顺序图中交互片段发生的概率来确定.在为每个功能都构造出相应的局部构件转移图后,还需将产生的各个局部构件转移图进行合并,以得到整个系统的构件转移图.最后,再基于系统构件转移图以及单个构件的可靠性信息,转换构造用于可靠性分析的 Markov 链.

2.2 基本的转换框架

根据上述分析,可以把 UML 模型转换为 Markov 链的过程分成 3 个步骤:第 1 步是根据活动图或顺序图构造局部构件转移图;第 2 步结合用况图中用况发生的概率,将局部构件转移图合并为整个系统构件转移图;第 3 步是结合构件本身的可靠性信息,将系统构件转移图转换为 Markov 链.具体的转换过程如图 5 所示.

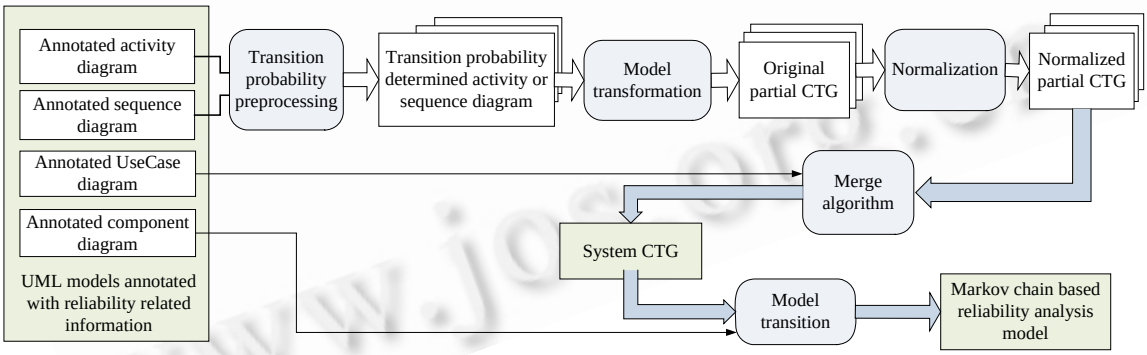


Fig.5 Basic transformation framework

图 5 基本的转换框架

首先,根据标注可靠性信息后的活动图或顺序图构造局部构件转移图.由于输入的活动图或顺序图只是根据场景的概率标注了分支的发生概率,还需要确定除分支外的其他活动边或消息发生的概率,也就是对转移概率进行预处理,这一步主要依据 Markov 性质来进行.然后,根据各活动边或消息的概率都确定之后的活动图或顺序图构造系统中的局部构件转移图.这一步主要是基于模型转换的方法来进行,其中得到的结果可能存在不规范的情况,还需要对转换结果进行规范化处理.每个活动图或顺序图都与一个局部的构件转移图相对应,其中的转移概率是在活动图或顺序图中的行为发生下的条件概率.为了得到系统中全部构件的交互情况,需要把这些局部构件转移图进行合并.由于局部构件转移图是通过活动图或顺序图构造而来的,而每个活动图或顺序图

与用况图中的一个用况相对应,因此,用况发生的概率表明了相应的活动图或顺序图对应的行为发生的概率,同时也影响了对应的局部构件转移图中的转移在整个系统中发生的概率.本文在合并时把用况发生的概率作为合并算法的另一个输入,以体现出用户使用系统的情况对可靠性的影响.经过上述步骤后,得到整个系统的构件转移图.利用系统构件转移图并结合构件本身的可靠性信息(从构件图中获取),通过相应的转换方法构造出最终利用 Markov 链表示的可靠性分析模型.

从UML模型转换为构件转移图,以及将构件转移图转换为可靠性分析模型,都可看作是模型转换的过程.本文利用模型转换语言QVT[18]定义了模型间的转换规则,对于QVT转换不能完成的工作,本文则定义了相应的算法进行处理,使得转换过程能够自动进行.

3 将 UML 模型转换为系统构件转移图

这部分的主要工作是通过标注可靠性信息后的 UML 模型构造整个系统的构件转移图.主要包含如下几个步骤,(1) 确定活动图或顺序图中各转移即活动边或消息发生的概率;(2) 将活动图或顺序图转换为局部构件转移图;(3) 合并局部构件转移图,构造整个系统的构件转移图.

本文假设系统只有 1 个入口点和 1 个出口点,对于存在多个入口点或出口点的情况,可以分别添加一个结点作为总的入口或出口.为了在系统构件转移图表示入口和出口对应的构件,在局部构件转移图中需要标明起始结点和终止结点.相应地,需要在活动图和顺序中标明起点和终点.对于活动图本身的初始结点(intialNode)和终止结点(finalNode),可直接将它们对应为构件转移图中的起始结点和终止结点;顺序图本身没有提供表明起点和终点的建模元素,需要人为地进行标注,本文在生命线中添加了 isStart 和 isEnd 属性,表示其对应的生命线是否是行为开始的起点或终点.最早发出消息的生命线为起点,最晚接收消息的生命线为终点.

3.1 转移概率预处理

活动图或顺序图中各条活动边或消息发生的概率是指在源端动作或操作发生的条件下目标端发生的条件概率.作为输入的活动图或顺序图只是根据场景发生的概率标注了相应分支发生的概率.在进行转换之前,需要先确定除分支外的其他活动边或消息的概率.根据Markov性质[17]可知,系统在时刻t_0所处的状态为已知的条件下,其在时刻$t>t_0$所处状态的条件分布与过程在t_0时刻之前所处的状态无关.也就是说,系统未来的状态只与当前状态有关,而与过去无关.故可以确定图 3 对应的活动图中其他转移的概率如图 6 所示.

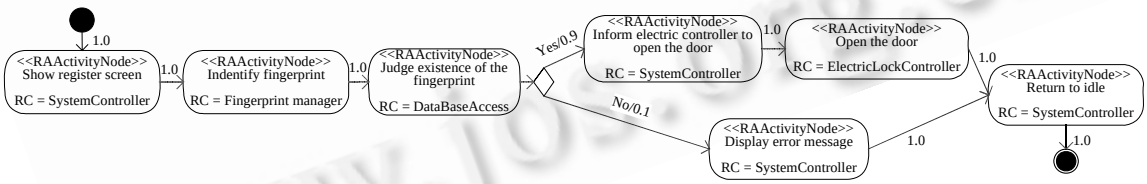


Fig.6 Transition probability preprocessing of activity diagram

图 6 转移概率预处理后的活动图

即除分支概率外,其余边的概率都为 1.0.这是因为在该活动发生的条件下,对于除分支以外的其他边,只要边的起始结点发生,且能正确完成,就一定会转移到边的目标结点.对于活动图中循环的情况,由于它可以被看作是一种特殊的分支,即从循环片段的最后一个结点出发,至少有两条转移路径,一是回到循环的初始结点,一是退出循环进入下一个结点,因此,只需将这两条转移路径看作两个分支,标注分支概率时对其进行相应的标注即可.

对顺序图的处理也是类似的,对于 alt 组合片段,只需将各个交互片段(即分支)的概率赋值给片段中发生的第 1 条消息的概率,其余的概率赋值为 1.0.loop 片段的处理与活动图是类似的,只需将它看作特殊的分支,将相应的分支概率标注给 loop 片段中的最后一条消息以及退出 loop 片段后发生的第 1 条消息即可.

对于系统中存在的并发情况,由于 Markov 链不支持系统同时处于两个或两个以上的状态,含有并发情况的

系统,其 Markov 链的构造相对而言比较复杂,本文将在第 5 节中进行详细的论述.

3.2 构造局部的构件转移图

在转移概率预处理完毕后,即可构造系统中各个局部的构件转移图.根据活动图或顺序图构造局部构件转移图主要分为两步,第 1 步是根据活动图中动作与动作间的转移或者顺序图中操作间的转移构造相应的构件和构件间的转移,各转移边的概率根据活动图或顺序图中动作的转移概率或消息发生的概率来确定.这一步的工作主要通过定义 QVT 规则并利用现有转换引擎实现;但是这样的映射可能会使得生成的局部构件转移图中存在相同的结点(多个动作或操作由同一个构件完成)或相同的边(两个构件间产生了多次转移),因此需要对第 1 步中产生的结果进行规范化,以得到规范化后的构件转移图.规范化过程需要对概率进行计算和处理,本文通过算法来实现.

由于 QVT 要求源端和目标端模型符合 MOF(meta object facility)元模型的约束,因此本文首先定义了加入可靠性信息后的 UML 元模型以及构件转移图的元模型,然后再介绍如何通过 QVT 规则和算法来实现各个转换.

3.2.1 扩展后的活动图、顺序图及构件转移图的元模型

UML元模型本身是符合MOF规范的[4],但本文在其顺序图和活动图中加入了可靠性相关的信息,因此需要对其元模型进行扩展.本文利用基于外廓(profile)的扩展机制来定义加入可靠性信息后的活动图和顺序图的元模型,如图 7 和图 8 所示.

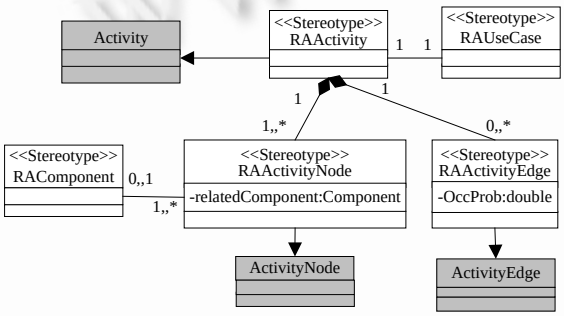


Fig.7 Metamodel of extended activity diagram

图 7 活动图扩展后的元模型

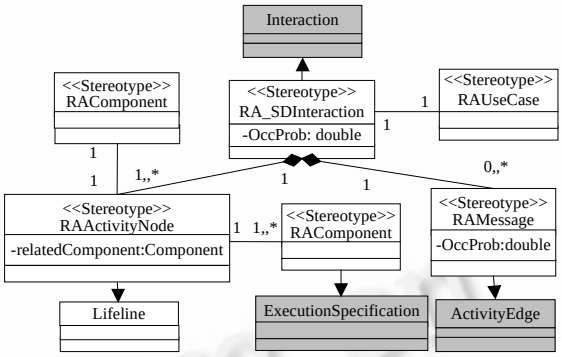


Fig.8 Metamodel of extended sequence diagram

图 8 顺序图扩展后的元模型

图中灰色的建模元素是标准UML中提供的,白色是添加可靠性相关信息后的建模元素,也就是扩展后的建模元素.为了把这两类元素区分开,本文在每一个扩展元素的名称前都加上了“RA”(reliability analysis)前缀,表明这些元素用于可靠性分析中.

每个活动用一个活动图来表示,它与某个用况相对应,用况发生的概率就是该活动发生的概率.ActivityNode 中添加的属性“对应的构件”用来表示执行该结点动作的构件;ActivityEdge 中添加的属性“发生的概率”表示这条边对应的转移发生的概率.顺序图也与某个用况相对应,其发生的概率是对应用况的发生概率.每条生命线都对应着一个构件,且分别用属性 isStart 和 isEnd 来表示该生命线是否为行为的起点和终点,执行规约即生命线中的操作,操作和操作间通过消息进行交互,每条消息都有自己发生的概率.

构件转移图主要由构件和构件间的转移组成,其元模型如图 9 所示.其中 CTGComponent 是最主要的结点,它与 UML 模型中的构件有着直接的对应关系,名字和可靠性都与对应构件相同;CTGStart 和 CTGEndNode 分别表示构件转移图的开始结点和终止结点,它们用来标明构件转移图的起始位置和终止位置,与实际的构件没有直接对应关系.此外,为了能够对并发进行处理,本文还在构件转移图中加入了组合结点(CompositeNode),以表示整个并发片段.它由多个分支结点组成,每个分支结点对应一个并发的分支,并且都可看作是一个构件转移图.转移用 CTGTransition 来表示,每个转移都有自己的发生概率,它与构件与构件间发生的转移概率是对应的.

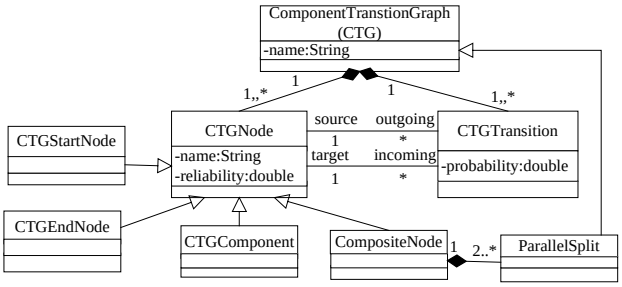


Fig.9 Metamodel of component transition graph (CTG)

图 9 构件转移图(CTG)元模型

3.2.2 活动图转换为初始的局部构件转移图

活动图主要用来对系统的一项功能进行描述,其中每个动作都由相应的构件来执行完成.因此,动作和动作间的一次转移就对应了相应的构件和构件间的一次转移.找出动作对应的构件,根据动作和动作间的转移关系,就可以得到一个构件和构件间的转移关系.动作间转移的概率就是当前活动发生的条件下,构件和构件间转移的概率.

上述对应关系利用 QVT 语言定义出的部分转换规则如图 10 所示.即根据每个活动结点中标注的信息,将活动图中的各个结点转换为对应的构件,控制结点如 Decision Point 等一般不会直接与某个构件对应或者可以把它看们成是其前驱结点对应的构件的行为,所以转换时不再单独考虑控制结点.活动图中的初始结点 IntialNode 和终止结点 FinalNode 分别转换为构件转移图中的起始结点 CTGStart 和终止结点 CTGEnd.原有的活动结点间的转移就映射为相应的构件和构件间的转移,概率保持不变,这样就能得到一个与活动图对应的局部构件转移图.

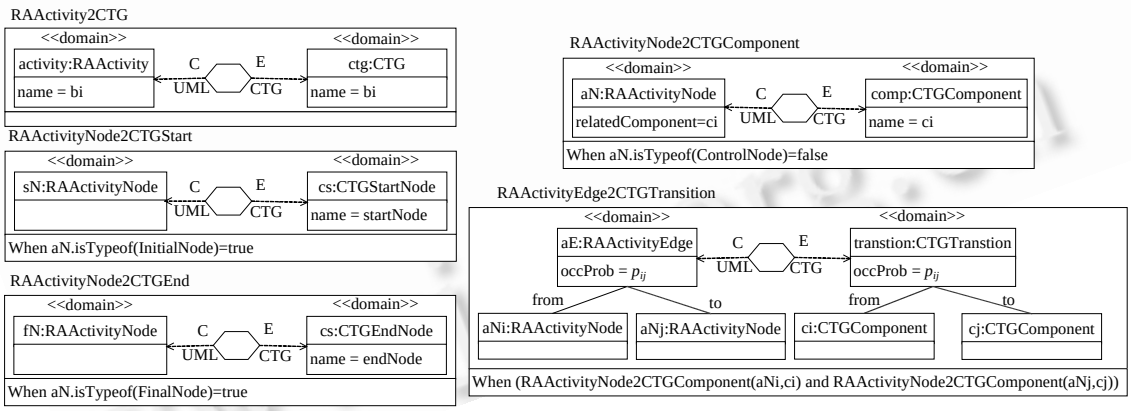


Fig.10 Examples of transformation rules from activity diagram to local component transition graph

图 10 活动图到局部构件转移图的转换规则示例

图 11 是对图 3 中的活动图利用转换规则后得到的局部构件转移图.其中存在相同的结点,在后面的算法中会对其进行规范化处理.

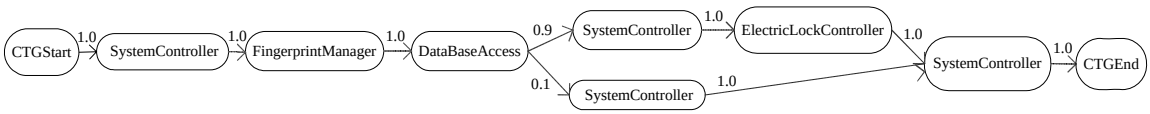


Fig.11 Non-Normalized local component transition graph corresponding to Fig.3

图 11 图 3 对应的未规范化的局部构件转移图

3.2.3 顺序图转换为初始的局部构件转移图

顺序图中主要的建模元素包括生命线、操作和消息.在本文中,顺序图用来表示为完成某个用况所对应的功能时,构件与构件间的交互情况.其中生命线与某个构件相对应,生命线上各个操作间通过消息发生的转移表明了对应构件间发生的转移.顺序图转换为构件转移图的过程与活动图类似.每个操作都与一个构件相对应,各个操作间的转移转换为生命线对应的构件间的转移;对于 isStart 属性为 true 的生命线,需要添加一个 CTGStart 结点和一条从 CTGStart 结点到该生命线对应的结点的转移,并把转移概率设置为 1.0;对于 isEnd 属性为 true 的生命线,需要添加一个 CTGEnd 结点和一条从该生命线对应的结点到 CTGEnd 结点的转移,概率设置为 1.0.用 QVT 描述的部分转换规则如图 12 所示.

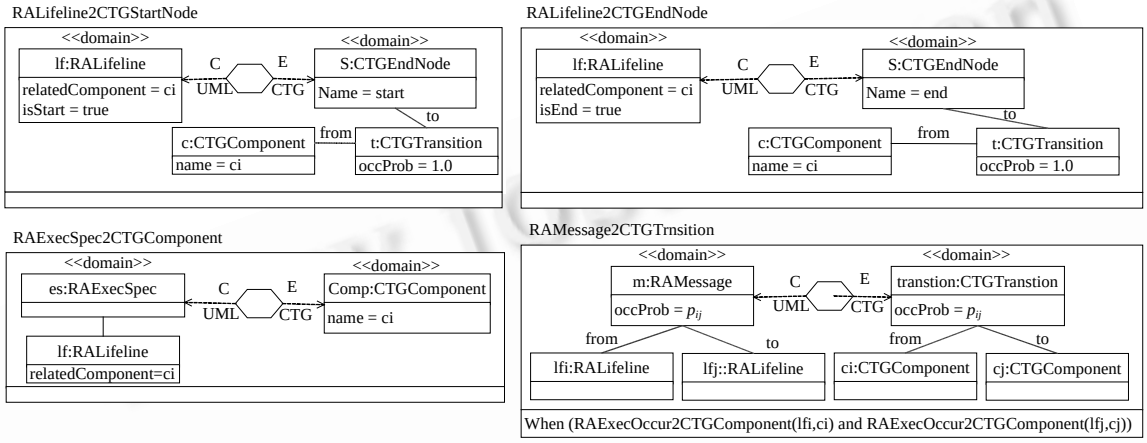


Fig.12 Examples of transformation rules from sequence diagram to local component transition graph

图 12 顺序图到局部构件转移图的转换规则示例

对图 4 中的顺序图执行相应转换规则后得到的局部构件图为图 3.图中存在相同的结点(一条生命线参与了多次交互),所以需要再进一步处理.

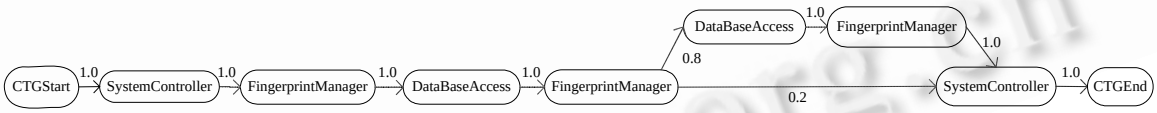


Fig.13 Non-Normalized local component transition graph corresponding to Fig.4

图 13 图 4 对应的未规范化的局部构件转移图

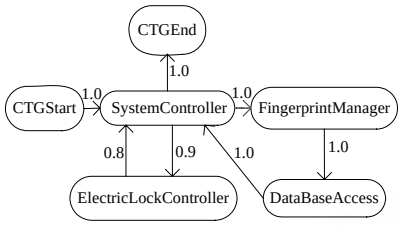
3.2.4 局部构件转移图的规范化

在通过活动图和顺序图构造局部构件转移图时,利用基于 QVT 的模型转换方法转换得到的结果中可能会有相同(同名)结点的情况(如活动图中多个动作由同一个构件执行)或者相同边的情况(如两条生命线间发生了多次交互),因此,需要对上一步中得到的局部构件转移图进行规范化处理,即合并相同结点及相同边.

在构件转移图中,构件间转移概率的统计学意义是指对于系统的 N 个输入集,该转移相对于所有的转移发生的比率,而由活动图或顺序图构造出的未规范化的构件转移图中的转移概率是根据 Markov 性质得出的,在源端的活动或操作发生的条件下转移到目标端活动或操作的概率,即条件概率,因此在合并之前,需要先将分支上所有的边乘以分支发生的概率,以表示在当前活动图或顺序图对应的行为发生的条件下,构件与构件间转移发生的概率.

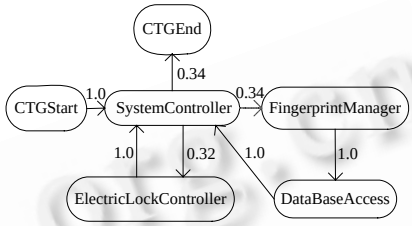
在处理完分支上的概率后,便可对相同(同名)结点和边进行合并.合并的规则是:对于相同且相邻的结点,将其合并为同一个结点,而不考虑其内部间的转移.即将相邻的两个动作或操作在同一个构件中运行的情况看成是构件内部的运行状态,从外部看到的结果只是该构件整体在运行;从该结点出发到达其他结点的转移和从其

他结点到达该结点的转移都保持不变;对于不相邻的相同结点,则在合并的同时,所有转移情况都保持原状.但是,如果存在相同的转移,即有相同的多条源端和目标端相同的转移边,则将这些边的概率相加,只保留其中的一条边,其余边都从图中删掉.由于从某结点出发的各转移边上的概率是指在相应的活动或交互发生的条件下,系统正处于该结点,且再从该结点出发转移到其他结点的概率,而不考虑以前的转移路径(Markov 性质),所以相加仍然能够保证合并后的转移概率反映实际的转移情况.具体算法如算法 1 中的 Step 1 所示.图 11 和图 13 经过 Step 1 处理后得到的结果分别如图 14(a)和图 14(c)所示.



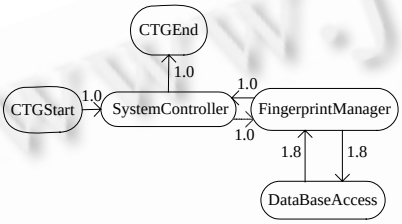
(a) Processing result of CTG in Fig.11 after Step 1

(a) 图 11 中的构件转移图经过 Step 1 后的结果



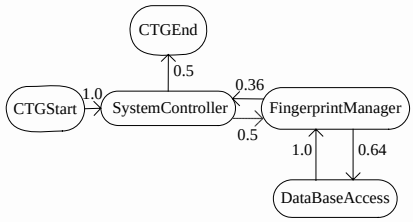
(b) Processing result of CTG in Fig.11 after Step 2

(b) 图 11 中的构件转移图经过 Step 2 后的结果



(c) Processing result of CTG in Fig.13 after Step 1

(c) 图 13 中的构件转移图经过 Step 1 后的结果



(d) Processing result of CTG in Fig.13 after Step 2

(d) 图 13 中的构件转移图经过 Step 2 后的结果

Fig.14 Normalized partial component transition graph

图 14 规范化后的局部构件转移图

由于在基于 QVT 规则直接生成的构件转移图(即初始局部构件转移图)中,从每个结点出发的所有边的概率之和都为 1.在规范化时,将概率直接相加可能会导致从同一结点出发的边的概率之和大于 1,因此,需要对这些结点进行归一化处理.本文的解决方案是考察每一个结点,如果从该结点出发的边的概率之和大于 1,则分别用这些边的概率除以它们的概率之和,以得到的新数值作为边的新概率.这样就保证了从某一结点出发的边的概率之和为 1,同时根据 Markov 性质,系统未来会转移到哪个结点是由当前结点决定的,这样的处理保留了转移的趋势,因此是合理的.具体算法如算法 1 中的 Step 2 所示.对图 14(a)和图 14(c)用 Step 2 处理后的结果分别如图 14(b)和图 14(d)所示.

算法 1. 规范化局部构件转移图.

把转换后得到的图用二元组$\langle V,E\rangle$来表示,其中 V 是结点的集合,E 是有向边的集合,其中对于 $\forall e\in E$, $e=(v_i,v_j,p_{ij})$,其中 $v_i,v_j\in V, p_{ij}\in R\wedge p_{ij}>0$ 为边的权值,表示从结点v_i到v_j的概率.

设$\langle V,E\rangle$为处理之前的图,$\langle V',E'\rangle$为处理之后的图.初始状态时,$V'=\varnothing,E'=\varnothing$.

Step 0. 将各个图中分支路径上的各条转移的概率乘以分支发生的概率.

Step 1. 对相同结点的处理:

For V 中的每一个结点v_i

1. 检查V'中是否有同名的结点.若没有,则将v_i添加到V'中
2. For E中的每条边$e_{jm}(v_j,v_m,p_{jm})$,
   a) If v_j与v_m具有相同名称, then 不进行后续处理(对应相邻的相同结点的情况);

b) If v_j与v_i相同,then

i. If E'中包含起始结点为v_j,终止结点为v_m的边 (v_j, v_m, p'_{jm}), then $p'_{jm} += p_{jm}$;

ii. Else将$e_{jm}(v_j,v_m,p_{jm})$添加到E'中.

Step 2. 对从每个结点出发的边的概率进行处理:

For V'中的每一个结点v_i

1. 从E'中找到所有从v_i出发的边的集合E''
2. 定义临时变量p_{sum}=0;
3. For E''中每个元素$e_{ij}(v_i,v_j,p_{ij})$,

a) p_{sum}+=p_{ij}

将E''中每个元素$e_{ij}(v_i,v_j,p_{ij})$的p_{ij}替换为p_{ij}/p_{sum},使得结点v_i的出边概率之和为 1.

经过上述步骤之后,就构造出了某个活动图或顺序图对应的规范化后的局部构件转移图.

3.3 合并局部构件转移图

一个活动图或顺序图只对应了系统中的一项功能,由它们构造出来的构件转移图只反映了系统中部分构件交互的情况.为了构造整个系统的构件转移图,需要将各个局部构件转移图进行合并.系统通常由多个用况组成,每个用况以一个活动图或顺序图来描述,每个活动图或顺序图都能构造出对应的局部构件转移图.用户使用不同功能的概率是不相同的,在合并时需要考虑局部构件转移图对应的用况发生的概率.

本文中合并的基本方法如下:首先,以用况的发生概率乘以相应的局部构件转移图中的各条转移边的概率;然后类似于图论中的子图合并方法,合并各个局部构件转移图.但各个局部构件转移图中可能并没有包含系统中所有的构件,所以在合并之前需要先做一个预处理,使得每个局部构件转移图中包含所有的构件,以保证构造出的系统构件转移图能够符合概率性质的要求.具体做法是:将局部构件转移图中不包含的结点加入到该图中,为每个这样的结点添加一条转移概率为 1.0 的到达终止状态的边.由于入度为 0,从局部构件转移图的初始状态出发不可能到达这些结点,从统计学意义来讲,它们对局部转移图中结点和结点间的转移是没有影响的.在对各个局部构件转移图预处理之后,合并各个图中相同的结点,如果存在相同边则只保留 1 条,将对应的转移概率相加.由于原本的局部构件转移图满足从各个结点出发的边的概率之和为 1 的条件,而所有用况发生的概率之和也为 1,所有经过上述步骤得到的系统构件转移图也是符合概率性质的.具体的合并算法如算法 2 所示.

算法 2. 合并局部构件转移图.

设系统中包含的构件集合为V,$|V|=m$,$\langle V_1,E_1\rangle,\langle V_2,E_2\rangle,\ldots,\langle V_n,E_n\rangle$为合并之前的图,即各个局部构件转移图;$p_1$,$p_2,\ldots,p_n$是这些图对应的用况发生的概率,$\langle V,E\rangle$为合并之后的图.

Step 1. 预处理.

For每一个构件转移子图$\langle V_i,E_i\rangle$,令$V_{\text{temp}}=V-V_i$

对V_{temp}中的每个结点v_j,都将其添加到V_i集合中,并添加一条到达终止结点CTGEndNode的边,转移概率为 1.0

End for

Step 2. 将预处理后的转移子图合并为系统的构件转移图.

初始状态时,$V=\varnothing$,$E=\varnothing$.

1. 首先对所有的$\langle V_1,E_1\rangle,\langle V_2,E_2\rangle,\ldots,\langle V_n,E_n\rangle$,将其中每条转移的概率乘以对应的用况发生的概率.
2. 构造一个$m\times m$的矩阵$\boldsymbol{A}$,对应i行j列的元素p_{ij}是从结点v_i出发到达结点v_j的概率,即v_i到v_j的转移概率.初始状态下,所有的p_{ij}=0.
3. For每一个图$\langle V_k,E_k\rangle$

a) For E_k中的每一个元素 $e_{ij}(v_i, v_j, p'_{ij})$

i. 在矩阵$\boldsymbol{A}$中找到相应的元素p_{ij},使 $p_{ij} += p'_{ij}$

4. End for

通过这样的方式,基于各个用况发生的概率和这些用况对应的局部构件转移图,就合并得到了整个系统中构件及构件间的转移情况及转移发生的概率.根据用况“指纹录入”和“识别指纹并开门”发生的概率,将图 14(b)和图 14(d)进行合并后得到的结果如图 15 所示.

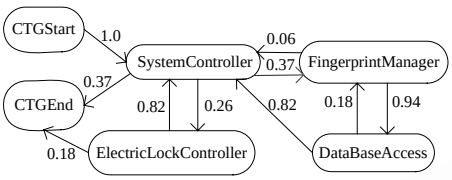


Fig.15 Merged component transition diagram (i.e. system level component transition diagram)

图 15 合并后的构件转移图(即系统的构件转移图)

上述过程首先通过活动图或顺序图构造系统局部的构件转移图,然后根据用况图中的概率信息,将局部的构件转移图进行合并得到整个系统的构件转移图.再从系统的构件转移图出发,结合构件本身的可靠性信息,就能构造出基于 Markov 链的可靠性分析模型.

4 将系统构件转移图转换为 Markov 链

系统构件转移图描述了整个系统中构件与构件间的转移情况及各转移发生的概率.其中的转移概率是假设系统中所有构件都正常运行时从起始构件转移到目标构件的概率,而不考虑构件失效带来的影响.为了分析系统的可靠性,在对应的 Markov 链中需要考虑构件失效为转移带来的影响,也就是各个构件的可靠性信息.

在可靠性分析使用的Markov链中,每个状态都与一个构件相对应.此外,还定义了两个吸收状态:终止状态C和失效状态F,其中C状态表示系统功能成功完成的状态,F状态表示系统发生失效后进入的状态.从构件转移图转换为Markov链的过程如下:构件转移图中的每一个构件结点CTGComponent都转换为Markov链中的一个非吸收状态;起始结点CTGStart或终止结点CTGEnd则转换为Markov链中的起始状态和终止状态;构件转移图中的每条转移都转换为相应的状态间的转移.由于每个构件都有可能发生失效,还需要修改各条转移边上发生的概率.具体做法是将每条转移边上的概率都乘以其源端构件的可靠性,只有当源端构件正常执行完毕时,才能到达其目标结点.即假设对于某条转移(C_i,C_j),其原有的转移概率为p_{ij},源端构件C_i的可靠性为R_i,则在Markov中要把p_{ij}变为$p_{ij}\times R_i$.另外,还要为每个状态C_i添加一条到达F状态的转移,转移概率设置为$1-R_i$.对于终止状态,则添加一条到达正确状态C的转移,转移概率为 1.0.

为了利用 QVT 来定义转换规则,本文定义了 Markov 链的元模型(如图 16 所示)并根据此元模型定义相应的转换规则,如图 17 所示.

根据构件图中每个构件的可靠性信息,对图 15 中的系统构件转移图应用转换规则并修改转移概率后得到的结果如图 18 所示.该模型可以直接作为现有数学分析方法的输入,对系统的可靠性进行分析[17].

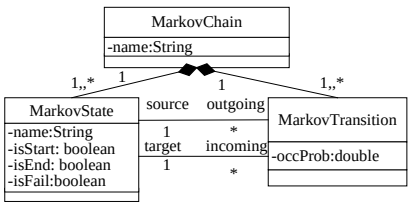


Fig.16 Metamodel of Markov Chain

图 16 Markov 链元模型

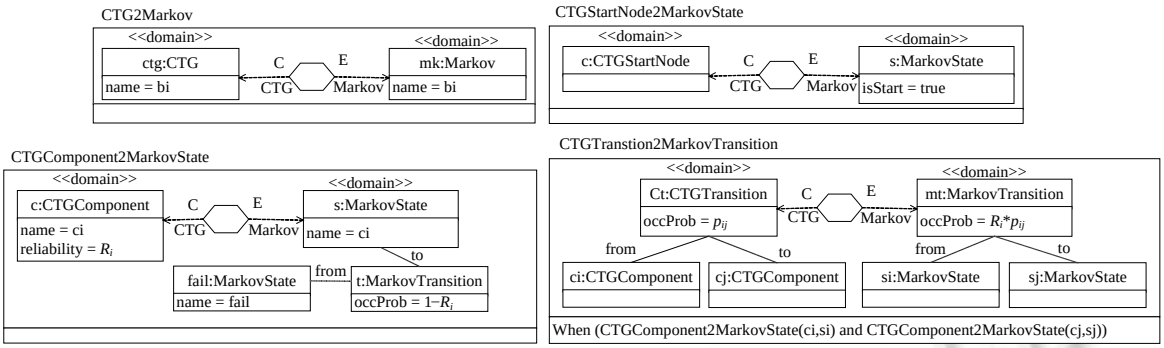


Fig.17 Examples of transformation rules from component transition graph to Markov chain

图 17 构件转移图到 Markov 链的转换规则示例

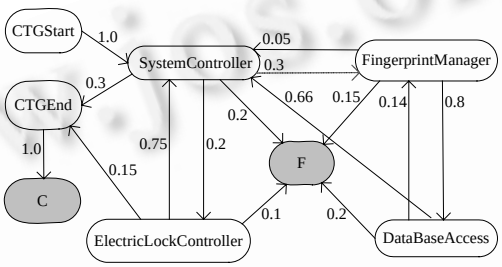


Fig.18 Markov chain corresponding to a simplified fingerprint control system

图 18 简化后的指纹控制系统对应的 Markov 链

5 对系统中并发情况的处理

在可靠性分析模型 Markov 链中,系统在同一时刻只能处于 1 个状态,也就是同一时刻系统中只能有 1 个构件在运行.然而,不管是顺序图中的异步消息、并发片段(parallel),还是活动图中的分岔(fork)和汇合(join)结点,都表示系统中存在并发的情况,也即系统中在同一时刻可能有多个构件在运行.这在 Markov 链中是不能直接表示的,转换过程中还需要对并发的情况进行特别的处理.

本文的解决方案是在构件转移图中引入组合结点的概念,将并发片段整体看作一个组合结点去参与局部构件转移图的构造,如图 19 所示.这样就把带有并发片段的构件转移图的构造过程变成了一个基本的不带有并发片段的转换过程,复用前面所述的转换方法就可构造出含有组合结点的 Markov 链.之后再对组合结点进行单独处理.由于构件转移图中只需要知道每个结点的可靠性和各条转移边的概率,因此,最终只需再计算出组合结点的可靠性即可构造出最终的 Markov 链.下面以一个活动图为例进行说明.

图 20 是一个带有并发片段的活动图,其中粗框部分是一个并发片段.引入组合结点后的转换过程主要包含如下几个步骤:(1) 将活动图转换为初始的局部构件转移图,具体做法与第 3.3.2 节相同,即并不考虑并发片段的特殊性,转换结果如图 21 所示;(2) 将并发片段整体看作一个组合结点(如图 19 所示),执行局部构件转移图的规范化处理,其中不需要考虑组合结点的内部结构;(3) 将带有组合结点的部分构件转移图当作一般的部分构件转移图,与其他的部分构件转移图合并得到系统的部分构件转移图;(4) 将带有组合结点的系统构件转移图转换为 Markov 链.此时构造出来的 Markov 链将会含有一个组合结点,其可靠性是未知的,还需要通过组合结点的内部信息进行计算.

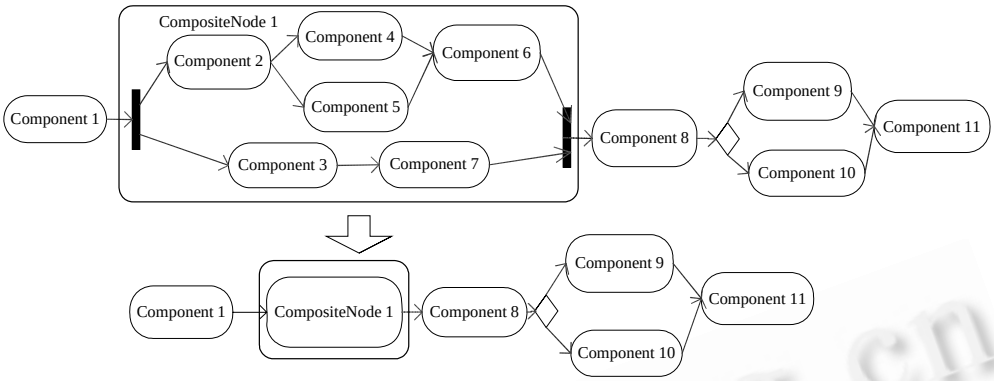


Fig.19 Original local component transition graph after introducing CompositeNode

图 19 引入组合结点后的初始局部构件转移图

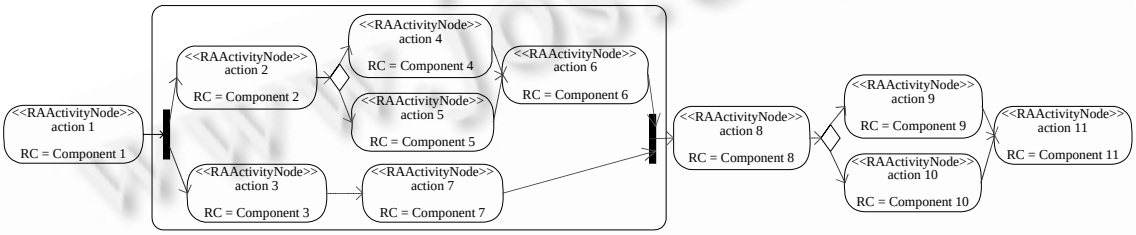


Fig.20 Activity diagram with parallel segment

图 20 带有并发片段的活动图

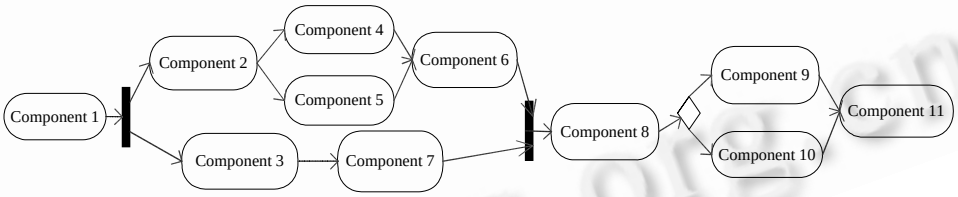


Fig.21 Original (non-normalized) local component transition graph

图 21 初始的局部构件转移图

本文中,每个组合结点对应于一个并发片段,而这些并发片段通常由多个并发分支(分区)组成(如图 22 所示).在顺序图里用par操作符表示的片段中,每个分区表示一个并行(并发)计算,其内部的消息是顺序执行的.如果不同的计算之间有交互存在,就不能用这种操作符[3].活动图中的分岔结点把一个单独的控制流分成两个或更多的并发控制流.一个分岔可以有 1 个进入转移和 2 个或更多的离去转移,每一个离去转移表示一个独立的控制流.在这个分岔之下,与每一个路径相关的活动将并行地继续[3].因此,一般来讲,顺序图或活动图的表示的并发片段中的各个分支之间都是相互独立、无交互存在的,只有当所有的并发分支都成功执行后,并发片段才算成功执行.所以,这种情况下并发片段的可靠性就可以看作是所有并发分支可靠性的乘积.从而求解组合结点的可靠性就变成了求解每个并发分支的可靠性.由于每个并发分支都可以看作是一个小的构件转移图,都分别对应着一个Markov链.因此,只需递归调用文中所述的转换算法,构造各个并发分支的Markov链,通过该Markov链计算出各分支的可靠性,就能计算出组合结点的可靠性,从而最终构造出含有并发控制流的系统的Markov链模型.含有并发控制流的活动图或顺序图转换为局部构件转移图的算法以及计算并发片段对应的组合结点的可靠性的具体算法如算法 3 所示.

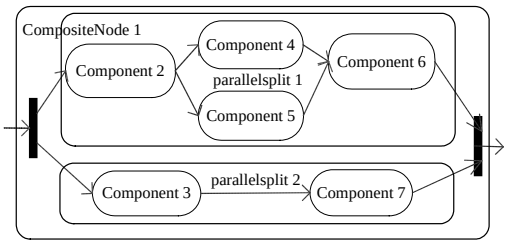


Fig.22 Internal structure of CompositeNode

图 22 组合结点内部结构

算法 3. 对活动图或顺序图中并发控制流的处理.

算法 3.1. 转换为含有组合结点的构件转移图.

Step 1. 根据第 3.3.2 节中的转换方法,将活动图或顺序图转换为初始的局部构件转移图.

Step 2. 将初始局部构件转移图中的并发片段替换为一个组合结点CN_i;并发片段中的并发分支都变成该组合结点的分支结点.

Step 3. 将组合结点看作普通结点,对局部构件转移图进行规范化,从而得到规范化后的含有组合结点的局部构件转移图.

算法 3.2. 计算并发片段对应的组合结点的可靠性.

假设组合结点为CN_j,其包含的并发分支的集合为$(branch_1,branch_2,\ldots,branch_n)$,每个并发分支$branch_i$都对应一个子的构件转移图$CTG_i$.

Step 1. For 每一个并发分支$branch_i$,

递归调用本文的转换方法,将其对应的CTG_i转换为一个Markov链M_i

根据M_i计算该分支的可靠性R_i

End for

Step 2. 根据分支的可靠性R_i,计算组合结点CN_j的可靠性 R_{CN_j}, $R_{CN_j}=\prod R_j$.

6 相关工作

在软件开发早期,基于系统模型对可靠性进行分析的过程主要包括 3 个步骤:第 1 步是在模型中加入可靠性相关的信息,也就是在原有模型中标注上这些信息,将原有模型和可靠性信息一起作为可靠性分析的输入;第 2 步根据标注有可靠性相关信息的模型构造对可靠性进行分析的数学模型,如 Markov 链、Petri 网、故障树等;第 3 步基于第 2 步中构造的分析模型,利用相关分析方法和工具对可靠性进行分析.

在软件设计模型中,表示可靠性信息的工作大多是基于UML来开展的,它们主要研究可靠性分析中的概念及相应的表示法,即建立可靠性分析的元模型,以便能够利用这些元模型在原有模型标注可靠性相关的信息.如文献[9]利用衍型、标记值、约束等对UML进行轻量级的扩展,使其能够对基于构件的系统的可靠性进行建模.文献[8]提出一个支持可信性建模的UML外廓,使得用户能够在UML模型中标注可信性相关的信息,为最终的分析提供支持.此外,还有一些用于非功能属性进行建模的标准的UML外廓,例如SPT[19],QoS&FT[20],MARTE[21].其中,SPT主要用来对性能、可调度性、时间等进行建模;QoS&FT是一个用于在UML中描述QoS特性的通用的框架;MARTE中提出一个非功能属性的建模框架,介绍了如何自定义非功能属性并将其在UML模型中进行表示.这些工作主要提供可靠性或其他非功能属性建模的方式,而没有进一步考虑如何将其转换为可靠性分析模型.本文则是以这些工作为基础,继续作更进一步的探讨,研究从这些模型到可靠性分析模型的 转换.

在基于数学模型对可靠性进行分析方面也有很多研究工作,如利用故障树、Markov链以及贝叶斯模型等[22].这些工作的侧重点主要是可靠性分析模型,研究如何利用这些模型分析软件的可靠性.在软件开发早期,尤

其是基于构件开发的软件系统中,Markov链是应用较广泛的一种数学模型[1,23−25].它通过系统中构件与构件间交互的情况来对可靠性进行分析.现有方法大多假设系统中构件与构件间的转移情况是已知的,而没有考虑如何通过早期的软件设计模型,如构件图、活动图、顺序图等来构造构件间的交互情况.因此,本文的工作与现有的可靠性分析工作是互补的,本文产生的结果能够作为这些方法的输入,为可靠性分析提供基本的支持.

文献[26]提出一个模型驱动的转换框架,通过自定义的语言 KLAPER 实现从设计模型到分析模型间的转换.其中间模型提出的目的是解决设计模型表示法的异构问题带来的转换复杂度的增长.该模型主要用来在不同的设计模型中捕获性能分析和可靠性分析中需要用到的信息,并把它们用统一的方式表示出来.该模型的抽象粒度是步骤(step),而本文则是构件.文献[27]以函数抽象为基础,提出了一个可靠性通用模型——构件概率迁移图.这个概念与本文中的构件转移图类似,但是其定义和构造与本文有着较大的差距.例如,本文中为了处理并发的情况还引入了组合结点的概念,并且它是通过函数抽象来计算迁移图中的概率,而本文则是以 UML 模型作为出发点.文献[28]给出了一种从软件 UML 模型构造软件 Markov 链使用模型的算法,但其主要工作在 UML 中加入统计测试约束来构造软件使用模型,以产生软件测试用例,而本文则是构造基于 Markov 链的可靠性分析模型.二者的目标有着较大的不同,在构造方法上也有着较大的差异.

另外,直接从 UML 模型出发对系统可靠性进行分析也有相关的一些工作.文献[10]从 UML 的用况图、顺序图和部署图出发,定义了一个基于贝叶斯的框架来对可靠性进行评估.文献[2]介绍了一个对可信性进行建模和分析的框架,其输入是 UML 中的结构视图对象图和类图,最终构造出一个时间 Petri 网来对可靠性和可用性进行分析.

在基于 Markov 模型对可靠性进行分析的工作中,文献[29]通过对用户剖面和环境构件的可靠性进行组合分析预测系统的可靠性,其系统模型使用体系结构描述语言 RADL 进行描述,Markov 链则根据用户剖面中的信息手工构造;文献[30]通过场景以及其中构件的可靠性来预测系统可靠性,其中的 Markov 链根据系统的消息序列图(message sequence chart)来构造;文献[25]过操作剖面、构件可靠性和体系结构来预测异构系统的可靠性,其中 Markov 链是通过系统的状态机图构造出来的;本文则主要是通过 UML 用况图、活动图、顺序图、构件图,自动地构造分析模型,为基于 Markov 链的可靠性分析提供支持.

7 总结及工作展望

在软件设计开发过程中,尤其是早期对可靠性进行分析对于构造满足用户可靠性需求的软件具有重要意义.根据可靠性分析的结果可以及时对软件设计中不合理的地方进行调整,或者在存在多种设计方案的情况下选择较优的设计,避免发生由于设计不合理导致软件最终实现不符合需求的情况,从而减少修改的代价,提高开发的有效性.软件设计模型是软件开发早期最主要的制品,它们是设计开发过程中对可靠性进行分析的重要输入.但是,软件模型与可靠性分析所用的数学模型间存在着较大的差异.为了简化可靠性分析的复杂度,减少设计人员的工作量,本文提出了一种从 UML 模型自动转换为可靠性分析模型 Markov 链的方法.该方法基于现有的可靠性建模和分析相关的研究工作,在 UML 的用况图、活动图、顺序图、构件图中加入可靠性相关的信息,通过活动图和顺序图构造系统局部的构件转移图,再根据用况图中描述的每个用况发生的概率,对这些局部转移图进行合并,构造出整个系统的构件转移图;根据系统的构件转移图,并结合构件图中描述的单个构件的可靠性,最终构造出用于可靠性分析的 Markov 链.其中,构件转移图是本文定义的一种中间模型,通过该模型能够简化转换的过程和复杂度,模型之间的转换工作主要通过定义 QVT 规则来完成,与概率的计算相关的工作则定义了相应的算法来处理.

本文的主要贡献在于,提出了一种自动地从 UML 模型转换为可靠性分析模型 Markov 链的方法.它将现有的可靠性建模工作和可靠性分析方法有机地结合起来,一方面,以现有可靠性建模工作的结果作为输入,可以看作是对现有建模工作的一个延续;另一方面,其产生的输出能直接作为可靠性分析的输入,使得在软件设计开发过程的早期对可靠性进行分析变得更加便捷和高效.甚至在代码实现完成后,本工作也能为基于体系结构的可靠性分析提供支持.未来的工作主要包括在转换过程中加入执行环境的可靠性信息;为转换过程的自动化提供

有效的工具支持等.

References:

[1] Gokhale SS, Trivedi KS. Analytical models for architecture-based software reliability prediction: A unification framework. IEEE Trans. on Reliability, 2006,55(4):578−590.

[2] Majzik I, Pataricza A, Bondavalli A. Stochastic dependability analysis of system architecture based on UML Models. In: de Lemos R, Gacek C, Romanovsky A, eds. Architecting Dependable Systems. LNCS 2677, Berlin, Heidelberg: Springer-Verlag, 2003, 219−244.

[3] Booch G, Rumbaugh J, Jacobson I. Unified Modeling Language User Guide. 2nd ed., New York: Addison Wesley Publishing Co., 2005.

[4] Object Management Group. Unified modeling language: Superstructure version2.0. OMG Document, formal/05-07-04, 2005. http://www.omg.org/spec/UML/2.0/Superstructure/PDF/

[5] Shao WZ, Yang FQ. Object-Oriented System Analysis. 2nd ed., Beijing: Tsinghua University Press, 2006 (in Chinese).

[6] Gokhale SS. Architecture-Based software reliability analysis: Overview and limitations. IEEE Trans. on Dependable and Secure Computing, 2007,4(1):32−40.

[7] Immonen A, Niemela E. Survey of reliability and availability prediction methods from the viewpoint of software architecture. Software and Systems Modeling, 2008,7(1):49−65.

[8] Bernardi S, Merseguer J, Petriu DC. Adding dependability analysis capabilities to the MARTE profile. In: Czarnecki K, Ober I, Bruel JM, Uhl A, Völter M, eds. Proc. of the 11th Int'l Conf. on Model Driven Engineering Languages and Systems. Berlin: Springer-Verlag, 2008. 736−750.

[9] Corellessa V, Pompei A. Towards a UML profile for QoS: A contribution in the reliability domain. ACM SIGSOFT Software Engineering Notes, 2004,29(1):197−206.

[10] Cortellessa V, Singh H, Cukic B. Early reliability assessment of UML based software models. In: Balsamo S, ed. Proc. of the 3rd Int'l Workshop on Software and Performance. New York: ACM Press, 2002. 302−309.

[11] Cheung L, Roshandel R, Medvidovic N, Golubchik L. Early prediction of software component reliability. In: Schäfer W, ed. Proc. of the 30th Int'l Conf. on Software Engineering. New York: ACM Press, 2008. 111−120.

[12] Mäkinen E, Systä T. MAS—An interactive synthesizer to support behavioral modeling in UML. In: Müller HA, ed. Proc. of the 23rd Int'l Conf. on Software Engineering. Washington: IEEE Computer Society Press, 2001. 15−24.

[13] Uchitel S, Chatley R, Kramer F, Magee J. System architecture: The context for scenario-based model synthesis. In: Roy BK, Meier W, eds. Fast Software Encryption, the 11th Int'l Workshop, FSE 2004. Berlin: Springer-Verlag, 2004. 33−42.

[14] Sun J, Dong JS. Design synthesis from interaction and state-based specifications. IEEE Trans. on Software Engineering, 2006, 32(6):349−364.

[15] Huzar Z, Kuzniarz L, Reggio G, Sourrouille JL. Consistency problems in UML-based software development. In: UML Modeling Languages and Applications. LNCS 3297, Berlin, Heidelberg: Springer-Verlag, 2005. 1−12.

[16] Cheung RC. A user-oriented software reliability model. IEEE Trans. on Software Engineering, 1980,6(1):118−125.

[17] Trivedi KS. Probability and Statistics with Reliability, Queuing and Computer Science Applications. New York: John Wiley&Sons, 2001.

[18] Object Management Group. Meta object facility (MOF) 2.0 query/view/transformations specification, version 1.0. OMG Document, formal/08-04-03, 2008. http://www.omg.org/spec/QVT/1.0/PDF/

[19] Object Management Group. UML profile for schedulability, performance and time. OMG Document, formal/05-01-02, 2005. http://www.omg.org/cgi-bin/doc?formal/05-01-02.pdf

[20] Object Management Group .UML profile for modeling QoS and fault tolerance characteristics and mechanisms, version1.1. OMG Document, formal/08-04-05, 2008. http://www.omg.org/spec/QFTP/1.1/PDF/

[21] Object Management Group. UML profile for modeling and analysis of real-time and embedded systems version1.0. OMG Document, formal/2009-11-02, 2009. http://www.omg.org/spec/MARTE/1.0/PDF/

[22] Singh H, Cortellessa V, Cukic B, Gunel E, Bharadwaj V. A Bayesian approach to reliability prediction and assessment of component based systems. In: Proc. of the 12th Int'l Symp. on Software Reliability Engineering (ISSRE 2001). Washington: IEEE Computer Society Press, 2001. 12−21.

[23] Gokhale SS, Trivedi KS. Analytical models for architecture-based software reliability prediction: A unification framework. IEEE Trans. on Reliability, 2006,55(4):578−590.

[24] Trung PT, Thang HQ. Building the reliability prediction model of component-based software architectures. Int'l Journal of Information Technology, 2009,5(1):18−25.

[25] Wang WL, Pan D, Chen MH. Architecture-Based software reliability modeling. The Journal of Systems and Software, 2006,79(1): 132−146.

[26] Grassi V, Mirandola R, Sabetta A. Filling the gap between design and performance/reliability models of component-based systems: A model-driven Approach. The Journal of Systems and Software, 2007,80(4):528−558.

[27] Mao XG, Deng YJ. A general model for component-based software reliability. Journal of Software, 2004,15(1):27−32 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/27.htm

[28] Yan J, Wang J, Chen HW. Deriving software Markov chain usage model from UML models. Journal of Software, 2005,16(8): 1386−1394 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1386.htm

[29] Reussner RH, Schmidt HW, Poernomo IH. Reliability prediction for component-based software architectures. Journal of Systems and Software, 2003,66(3):241−252.

[30] Rodrigues G, Rosenblum D, Uchitel S. Using scenarios to predict the reliability of concurrent component-based software systems. In: Cerioli M, ed. Fundamental Approaches to Software Engineering, FASE 2005. Berlin: Springer-Verlag, 2005. 111−126.

附中文参考文献:

[5] 邵维忠,杨芙清.面向对象的系统分析.第 2 版,北京:清华大学出版社,2006.

[27] 毛晓光,邓勇进.基于构件软件的可靠性通用模型.软件学报,2004,15(1):27−32. http://www.jos.org.cn/1000-9825/15/27.htm

[28] 颜炯,王戟,陈火旺.基于 UML 的软件 Markov 链使用模型构造研究.软件学报,2005,16(8):1386−1394. http://www.jos.org.cn/1000-9825/16/1386.htm

柳毅(1987－),女,云南昭通人,博士生,CCF 学生会员,主要研究领域为面向对象建模,元建模,模型驱动的体系结构,非功能属性的建模和分析.

何啸(1983－),男,博士生,主要研究领域为面向对象建模,元建模,模型驱动的体系结构,模型转换技术.

麻志毅(1963－),男,博士,副教授,CCF 高级会员,主要研究领域为软件工程与支撑环境,软件建模技术,面向对象技术.

邵维忠(1946－),男,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件工程环境,面向对象方法,软件复用与构件技术.