****

**本科毕业设计（论文）外文翻译**

|  |  |
| --- | --- |
| **学 院：** | 电力学院 |
| **专 业：** | 电气工程及其自动化(卓越班) |
| **学生姓名：** | 阮艺源 |
| **学生学号：** | 201730210466 |
| **指导教师：** | 张俊勃 |

|  |  |  |
| --- | --- | --- |
| **中文译名** | 量化软件性能、可靠性和安全性：基于架构的方法 | |
| **外文原文名** | Quantifying software performance, reliability and security:An architecture-based approach | |
| **外文原文版出处** | | Journal of Systems and Software 80.4 (2007): 493-509. |
| **译 文：**  摘要：  随着基于组件的系统越来越流行，处理多样化的关键应用，对其进行全面评估的需求变得非常重要。在本文中，我们提出了一种基于架构的软件性能统一层次模型。可靠性、安全性和缓存行为预测。我们采用离散时间马尔可夫链(DTMCs)对软件系统进行建模并提供表达式，用于根据系统的架构以及单个系统的特性来预测系统的整体行为。组成部分。这种方法也有利于识别各种瓶颈。我们通过一些案例研究来说明它的使用情况，并且还提供了执行敏感性分析的表达式。  关键词：软件架构；性能；可靠性；安全；马尔科夫模型 软件架构；性能；可靠性；安全性；马尔科夫模型   1. 介绍   复杂的软件系统几乎在当今生活的每一个方面都在幕后工作，而且比以往任何时候都更依赖于这些系统的可靠性、性能和安全性。不可靠的软件系统可能会造成任何事情，从不便（如网上银行系统故障）到甚至致命的事故（如在太空发射过程中关键软件系统的故障）.软件应用程序的性能不佳可能会导致日常操作的效率和重大延误，造成时间和金钱的浪费。另一方面，不安全的软件系统可能会让入侵者对其进行控制，这取决于应用程序的种类，可能从银行欺诈到恐怖行为。  对软件系统的可靠性、性能和安全性进行充分的分析，可以帮助确定其不足之处。 除了防止大的事故外，确定软件系统相对于它所部署的硬件的行为也可能是有帮助的，并且可以做出与特定方面有关的预测，例如相同的缓存丢失行为。  随着基于组件的设计越来越流行，许多软件系统都是使用现成的组件来构建的，系统不能再被视为上述分析和评估的单一实体。软件系统的体系结构决定了不同组件之间的交互方式，也是影响系统行为和性能的一个重要因素。盛行的黑盒模型显然不适合对这种基于组件的软件系统进行建模。  最近，已经有一些关于基于架构的系统分析的工作。这些方法中，有的分析系统的可靠性(Gokhale和Trivedi,2002;Goseva-Popstojanova和Trivedi,2001;Cheung,1980)，有的分析系统的性能(Smith等人，2005;Sharmaet等人，2005;Petriu等人，2000)。这些方法采用了许多技术来表示软件系统，从马尔科夫链、执行图到UML等。系统的缓存缺失行为也是影响系统性能的一个重要因素，已经提出了一些分析软件系统缓存缺失行为的方法(Clark和Emer,1985；Stone等人，1992；Rao，1978)。然而这些方法并不是专门针对基于架构的系统分析。一些方法也被提出来用于系统的安全评估，但这些方法同样没有考虑到系统的总体架构，而且多次重复性不高。只有少数几种方法如AVA(Voaset al.,1996)产生了可重复的结果，但它们同样倾向于对软件进行整体处理。  一般来说，已知的定量方法通常只集中在某一特定的属性上，对于同一系统的多属性分析，要采用不同的建模和分析方法，需要投入大量的精力。此外，现有的大部分安全和遗漏分析方法都不适合架构级分析。  在本文中，我们提供了一种分层方法，统一分析基于组件的系统的性能、可靠性、缓存丢失行为和安全性。层次化方法的第一步是使用离散时间马尔科夫链或DTMCs对软件架构进行建模（Trivedi，2001）。然后对DTMC模型进行分析，以获得后续属性具体分析的重要信息。该方法的独特能力允许对不同的属性进行定量分析，使其非常适用于比较不同的软件架构以及组件类型，以实现沿每个属性所需的整体行为。然后对DTMC模型进行分析，以获得后续属性分析的重要信息。该方法的独特能力允许对不同的属性进行定量分析，使其非常适用于比较不同的软件架构以及组件类型，以实现沿着每个属性的所需的整体行为。  我们的方法对于系统设计时的分析和现有系统的评估都是有用的。该方法可以很好地适应于软件开发的设计阶段，因为系统软件架构的变化不会导致整个模型的崩溃。此外，单个组件行为的变化也可以在不改变其他模型参数的情况下被容纳，这使得它适合在开发的后期阶段使用，当组件的测试和调整正在进行。我们还提供了可以提取系统的架构来构建模型的方法。  本方法将系统的软件架构作为预测其安全性和缓存性能行为的重要输入，而这一点在大多数相关研究中都被忽略了。我们尽可能地将二阶架构的影响考虑在内，以提高准确度，结果表明，预测结果与观察到的行为非常接近。我们还提供了一种方法，通过进行灵敏度分析来衡量单个组件行为的变化以及工作负载对系统整体行为的影响。  本文其余部分划分如下。第2节简要介绍了相关工作和DTMC的背景，第3节介绍了分层模型，并提供了预测系统可靠性、缓存遗漏行为和花费时间的表达式，此外还定义了漏洞指数，并提供了同样的表达式。第4节深入探讨了提取系统架构的方法。第5节提供了sthe例证和案例研究，其中在方法已被应用，并提出了所获得的结果。第6节讨论了多用户存在下的性能评估问题。第7节讨论了对所考虑的系统进行模型参数敏感性分析的技术。第8节是关于纳入各组件之间的依赖性所需的修改，第9节是本文的结论。   1. 相关工作和背景   我们介绍了在量化各种软件属性方面的相关工作，并简要介绍了DTMCs，我们用它来模拟基于组件的系统中的控制流。   * 1. 相关工作   基于架构的分析旨在考虑到构成应用程序的组件和架构的行为。近来有一些研究(Gokhale和Trivedi，2002；Goseva-Popstojanova和Trivedi，2001；Gokhale等人，1998；Goseva-Popstojanova等人，2001)既关注软件应用的可靠性，又关注软件应用的性能，同时考虑到软件架构。在Goseva-Popstojanova和Trivedi(2001)中，软件可靠性评估的不同方法被分为三类：基于状态的、基于路径的和加法的.基于状态的模型使用软件的控制流图来表示系统的体系结构，它可以被建模为离散时间马尔科夫链或DTMCs(Gokhale和Trivedi2002；Goseva-Popstojanova和Trivedi，2001；Cheung，1980；Gokhale等人，1998)。 Goseva-Popstojanova等人，2001；Reussner等人，2003）、连续时间马尔科夫链（CTMCs）（Laprie，1984；Ledoux，1999）或半马尔科夫过程（SMPs）（Kubat，1989；Littlewood，1975）。在基于路径的模型中(Yacoub et al., 1999; Shooman, 1976)，系统的可靠性是考虑到程序可能的执行路径而计算的。加法模型(Everett, 1999; Xie and Wohlin,1995)没有明确考虑软件的结构，而是在假设各个组件的可靠性可以用非同质泊松过程(NHPP)建模的前提下，以组件故障强度的总和来估计系统故障强度。  基于状态的模型可以进一步分为两类：复合型和层次型。复合模型将软件架构和软件的失效行为合并在同一个模型中，而分层模型则是将软件架构单独建模，然后对其进行求解，并将求解结果与组件的失效行为叠加，以预测可靠性。软件体系结构或单个组件行为的变化在复合模型中比较难以适应。此外复合模型容易出现刚度问题，这是因为组件的失效和转换概率的大小存在巨大的相对差异（与后者相比，前者通常非常小）。  层次模型比复合模型提供了更多的灵活性，因为软件架构是单独建模的。不同的架构选择可以只通过改变架构模型来评估，从而将开销降到最低。僵化问题(Reibman and Trivedi, 1988; Bobbio and Trivedi, 1986)也在很大程度上减少了，因为软件架构模型首先被单独求解，故障行为被叠加在这个解上，以寻找整体可靠性。然而分层模型只提供了复合模型的近似值，因此准确的分层建模是一个重要的课题。Gokhale和Trivedi(2002)已经提出了在建模中加入二阶结构效应的问题，这是实现这一目标的第一步。   1. U.Smith在她的开创性工作（Smith，1990）中首次提出了软件性能工程（SPE）一词，作为软件系统性能评估的方法论。实现这一方法的工具称为SPEÆED。该工具可用于使用执行图(EG)来指定软件执行模型，它就像注解的流程图。系统执行模型，即建模争论，是通过使用模拟来实现的。最近，有人提出了将UML模型导入到SPEÆED的努力(Smith等人，2005)。Petriu和Woodside使用了一个名为UCM2LQN的工具，将基于Use case Map的软件系统表示法转换为LQN模型(Petriu et al., 2000)，然后可以使用他们的LQN求解器进行性能评估(Petriu and Woodside, 2002)。   系统性能的一个重要方面是其缓存行为，它既依赖于系统的软件，也依赖于系统的硬件架构。三种方法已经被普遍用于缓存性能评估：实时执行、跟踪驱动的仿真和建模.实时执行方法(Clark和Emer,1985)的缺点是仅限于存在的硬件。实时执行方法(Clark和Emer,1985)的缺点是只限于现有的硬件，而跟踪驱动的模拟方法(Wang和Baer,1990)往往非常耗费空间和时间(Li,2000)。此外，这两种方法都缺乏对软件架构如何影响缓存性能的见解。经验模型如(Stone et al., 1992; Chow,1976; Thiebaut, 1987)试图从容量、工作集大小、空间和时间位置以及两者之间的相互作用等方面对缓存性能进行参数化。但它们只适用于某些缓存配置和工作负载。分析模型和混合模型，如(Rao,1978；Horowitz等人，1989；Martonosi等人，1997)确实提供了对程序性质和影响缓存性能因素的见解。但它们也有自己的缺点，如大量的参数(Rao，1978；Horowitz等人，1989)和只应用于指定的程序结构(Martonosi等人，1997)。读者可参考(Li,2000)对这些模型的详细研究。  软件安全是另一个正在进行大量研究的领域。已有一些人试图通过 "老虎小组 "的渗透实践来量化软件系统的安全性，即一群专家坐在一起，试图利用软件系统可能存在的任何弱点来入侵。然而这种做法对于组成TigerTeam的人来说是非常主观的，因此具有不可复制性。有一些方法，专注于软件开发过程中所采用的过程，以评估最终产品的安全性。其中一个例子是SSECMM或系统安全工程能力成熟度模型。然而，通过评估软件的开发过程来标榜软件是安全的，并没有得到太多的欢迎。这是因为即使遵循了最佳实践，最终产品中仍有一些弱点存在，而这些弱点不会被发现，直到对其漏洞进行严格测试。  Cigital labs (formerly Software Technologies) 提出了一种产品评价方法，称为 AVA或适应性弱点分析（Voas等人，1996年）。 1996). 其基本重点是观察以下因素的影响： 传入的模拟感染(它模拟的威胁是由一个人或几个人组成的。) AVA）对执行系统的影响。危险的输出条件是通过对程序的断言来指定的。 变量状态。然而这种方法将软件 单片机。软件工程学院（SEI）在 CMU在这个领域也非常活跃，他们有 拟议的V-RATE或供应商风险评估和威胁。 评价（Lipson等人，2001年），可在使用时 决定购买某项产品时选择哪家供应商。 COTS组件。  总的来说，在现有的很多工作中，软件系统的不同属性，如可靠性、性能和安全性等都是分开处理的，很多时候都没有把软件架构纳入到方法中。   * 1. DTMCs-一个基本的背景   在本节中，我们将讨论离散时间马尔可夫链(DTMCs)，我们用它来模拟系统的软件架构。马尔可夫过程是一个随机过程，它的动态行为是这样的：它的未来发展的概率分布只取决于当前的状态，而不取决于该过程如何到达该状态。如果我们假设状态空间I是离散的（有限或可数无限），参数空间T也是离散的，那么我们就有一个DTMC。一个DTMC由它的状态和状态之间的过渡概率来表征。一阶过渡概率矩阵P=[pi,j]是一个随机矩阵，因为P的一行中的所有元素加起来都是1，而且pi,j的每一个都在[0,1]的范围内。  根据我们的目的，我们将DTMCs分成下面两类：  不可减少：如果每一个状态都能从每一个状态在有限步内到达。  吸收：如果至少一个状态没有向外状态转移。  我们可以将吸收DTMC的过渡概率矩阵划分为：  qt_temp  如果DTMC有n个状态，有m个吸收状态，则Q将是一个(n - m)(n - m)的亚稳态矩阵(至少有一行之和<1)，描述瞬态之间的过渡概率，**1**是一个mm的同位矩阵，**0**是一个m(n - m)的零矩阵，**C**是一个(n - m)乘m的矩阵。由Pk给出的k步过渡概率矩阵的形式为：  C:/Users/15337/AppData/Local/Temp/qt_temp.O19732qt_temp  Qk的第(i，j)项表示从状态si开始经过精确的k步后到达状态sj的概率。 可以证明，随着t接近无穷大，C:/Users/15337/AppData/Local/Temp/qt_temp.k19732qt_temp收敛。 因此，存在逆矩阵(I - Q)-1。 这称为基本矩阵**M**  qt_temp  对于包含许多软件组件的应用程序，我们可以使用DTMC表示其软件体系结构。 在执行步骤中DTMC的状态由该步骤中正在执行的组件给出。 而且，状态之间的转换表示控制从一个组件到另一组件的转移。   1. 层次模型及其应用   如第2节所述，可以将终止应用程序的软件体系结构表示为吸收性DTMC。在本节中，我们介绍在分层模型中利用DTMC表示法来预测其可靠性，性能，安全性和缓存行为的方法。 软件。 我们假定应用程序每次访问在每个组件中花费的时间是一个具有均值和方差的随机变量。 我们还假设每个组件的可靠性，漏洞指数和缓存丢失率是确定的，可以确定。  我们假设应用程序有n个组件，其中初始组件索引为1，该组件首先接收控制流，最后一个组件索引为n，之后程序终止。 这两个分量分别由DTMC的初始状态和吸收或完成状态表示。 通常，此DTMC中从状态i到j的转换表示控制从组件或模块i到软件或程序中的j的转移。 通过应用程序组件的控制流由单步转移概率矩阵P给出。因此，我们可以找到对各个组件的预期访问次数以及访问次数的方差，如第2.2节中所述。 可以使用各种配置工具来提取现有软件系统的体系结构，例如gprof（GNU gprof，1998），ATOM（Srivastava和Eustace，1994），ATAC（Horgan和London，1992）等。 第4节中的问题。在系统开发过程中，在体系结构阶段，需要根据类似以前系统的经验来估算组件的各个属性，例如可靠性，性能属性等。 但是，可以通过测试和测量组件属性为现有系统计算参数（Gokhale和Trivedi，2002）。  图1显示了后续小节中遵循的基于体系结构的层次分析方法。 只需将应用程序的软件体系结构建模为DTMC一次，然后通过分配适当的奖励，就可以对应用程序进行性能，可靠性，安全性和缓存性能分析。   * 1. 可靠性预测   可以通过利用DTMC分析获得的访问次数的均值和方差，并将其与各个组件的可靠性结合起来，来计算由多个软件组件组成的软件应用程序的可靠性(Gokhale和Trivedi，2002年)考虑状态i代表执行中的第i个组件。假设组件彼此独立以及在连续的执行中发生故障，如果Ri表示组件i的可靠性，我们可以将应用程序的可靠性表示为    注意，由于访问每个组件的次数是一个随机变量（组件n除外），所以R本身是一个随机变量。 应用程序的预期可靠性由下式给出：    右侧产品中的术语C:/Users/15337/AppData/Local/Temp/qt_temp.t19732qt_temp是单次执行时组件i的预期可靠性。 使用泰勒级数展开式    者可以写作：    请注意，对第n个组件的访问次数（DTMC中的吸收状态）始终为1，因此E [X1，n] = 1，而Var [X1，n] = 0，所以：    因此，考虑到二阶体系结构效应的影响，应用程序的总体预期可靠性(Gokhale和Trivedi，2002)    通过访问组件的次数的变化来捕获二阶体系结构效果，并提供更准确的预测，该模型中唯一近似的来源是泰勒级数截断。 但是，如果忽略了二阶体系结构的影响，则可以得到应用程序的预期可靠性，因为：    我们将具有最低C:/Users/15337/AppData/Local/Temp/qt_temp.r19732qt_temp值的组件标记为软件应用程序的可靠性瓶颈。  近期有许多研究假设组件故障会影响整体可靠性（Goseva-Popstojanova等，2001； Yacoub等，2004）。 但是，这种关于独立组件行为的假设可以放宽。 我们将在第8节中对此进行讨论。   * 1. 性能预测   考虑一下，我们用一个随机变量Ti表示执行一个应用程序的第i个组件所花费的时间。如果DTMC状态i表示驻留在组件i中的控制流，则在状态i中花费的平均时间为C:/Users/15337/AppData/Local/Temp/qt_temp.h19732qt_temp，其方差为C:/Users/15337/AppData/Local/Temp/qt_temp.w19732qt_temp。 因此，我们使用从状态1开始的访问状态i的次数（用X1，i表示）来查找完成应用程序的平均时间（Gokhale和Triv edi，2002）。由随机变量T表示的单次运行完成申请的时间为：    请注意，这里我们为每个状态i分配Ti作为奖励（Trivedi，2001），并且我们有兴趣计算直到达到吸收状态的预期累积奖励，这表示应用程序执行已完成。 该预期奖励表示完成申请的预期时间，可以表示为：    因此，假设在每个模块中花费的时间之间的独立性，可以得出完成应用程序所需时间的方差为：    我们将具有C:/Users/15337/AppData/Local/Temp/qt_temp.h19732qt_temp最大值的组件标记为软件应用程序的性能瓶颈。  组件的预期执行时间的概念在文献中是众所周知的（Trivedi，2001； Smith，1990； Knuth，1997）。 应该注意的是，可以为组件指定许多性能属性，例如CPU时间，磁盘I / O时间等，而不是像执行时间这样的组合度量。 还应注意，上面有关性能预测的讨论假定该应用程序仅处理一个请求，即在托管该系统的任何计算机上都没有争用。 在这种情况下，将需要一个详细的基础性能模型（Trivedi，2001； Smith，1990； Petriu等，2000）来找出完成请求所需的时间。 我们将在第6节中介绍一种基于排队建模的性能评估方法。   1. 提取软件架构   有关系统软件体系结构的知识对于应用建议的层次分析方法至关重要。在构建与系统相对应的DTMC模型时，需要标识各个组件，并考虑它们与其他组件的交互。  在将方法应用于正在构建的软件系统时，软件体系结构通常以某种标准形式（例如UML图）可用，并且可以识别单个组件。有关组件之间交互的信息可以从以前使用类似软件组件的经验中估计出来。 Yacoub等人已经提出了一种基于系统的运行状况，根据各种执行场景的发生概率来估计组件之间的控制流转换概率的方法。（2004）。  关于组件的各个特性的定量信息，即可靠性，性能要求等，也无法整体获得，因此使用了这些值的估计值。但是，随着软件开发的继续，这些估计值会变得更好，因此就准确性而言，分析会随时间而改进。  但是，对于现有系统，主要关注点是确定软件体系结构和构成系统的不同组件。 诸如gprof，ATOM，SWAG Kit等工具可用于提取系统的软件体系结构。 GNU profiler或gprof（GNU gprof，1998）对于导出C和C ++程序的平面轮廓和调用图非常有用。平面配置文件显示程序在每个函数中花费了多少时间，以及该函数被调用了多少次。 调用图提供有关调用特定功能的其他功能以及由该功能调用的功能的信息，以及调用次数。 这些信息有助于构建DTMC模型，并获得相关的转移概率。  ATOM（Srivastava和Eustace，1994年）是一个工具包，提供了一组检测API，可用于提取程序的调用图信息，包括过程的执行时间和被调用的次数。与gprof不同，它不需要应用程序的源代码，并且使用对象模块，因此使其使用独立于编程语言和编译器。但是，ATOM仅可用于Tru64 UNIX操作系统。 最近，已经开发了一种称为SWAGkit的工具包（SWAGkit，2006年），该工具包可用于提取，抽象和介绍软件体系结构。当前，SWAGkit支持提取C / C ++代码，抽象到体系结构级别以及以图形形式表示。它已被用来分析和可视化一些复杂的软件系统，包括Linux操作系统内核（Bowman等，1999）。   1. 结论   在本文中，我们提出了一种基于软件系统的软件体系结构及其组成组件的属性来预测软件系统各种属性的分层方法。 我们的方法的独特之处在于，它在同一模型中将可靠性，性能，安全性和缓存未命中行为预测结合在一起，而该模型本质上是分层的。 本文扩展了在基于体系结构的软件系统可靠性和性能分析领域的先前工作，并通过考虑软件体系结构在安全性预测和缓存缺失分析领域做出了最初的贡献，其他研究 直到现在都缺乏。  该模型相当准确，因为它也考虑了二阶体系结构的影响。 我们还建议使用高阶马尔可夫链来表示系统，其中控制流与通过软件组件的路径无关。  由于模型的分层性质，仅更改体系结构模型就可以非常轻松地实现系统软件体系结构的更改。  因此，我们的方法非常适合在软件开发生命周期的设计阶段中使用。 此外，组件的个别行为的变化不会导致模型崩溃，因为该变化本质上是局部的，同样是由于层次结构的性质。 因此，这种方法在软件开发生命周期的后期阶段，即组件进行测试并同时进行更正和/或更新时，也非常有效。  这种方法的局限性在于，很难使用DTMC对控制流的并发进行建模，并且必须将同时执行的组件建模为DTMC模型中的单个状态  我们在本文中介绍了我们的方法的分析处理，并提供了可靠性，性能，缓存缺失分析和安全性的表达式。 通过展示两个案例研究，我们还展示了这种方法在实际系统中的应用。 我们还说明了该方法在满足多个客户的系统性能分析中的应用，因此具有资源占用。 我们已经介绍了形式敏感性分析（使用导数）以及蛮力方法。 这对于未准确测量或估计各个参数的系统非常重要。  可以扩展此方法以合并软件系统中可能存在的各种级别的容错模型，并可以用来研究它们对各种系统属性的影响。 这是我们当前关注的领域之一（Sharma和Trivedi，正在准备中）。 另一个扩展可能是自动建议替代方案，以优化各种系统属性，例如性能和可靠性。 各种属性之间可能存在折衷，因此选择优化这些属性的系统配置非常重要。  我们也没有在此模型中包含费用。 适当的成本可以与各种组件相关联，并且配置和方法也可以用于输出不同系统配置的成本。 这将有助于软件工程师和设计人员在不超出预算的情况下确定最佳系统。这是未来研究的另一个方向。 | | |
| 审核意见：  指导老师（签字）：  年 月 日 | | |

说明：

1.外文原文出处包括出版社、出版时间、期刊的刊名、刊号、刊期；如翻译内容为外文著作，需写明翻译的著作名称、章节及页码；如翻译多篇文献，需将各篇文献分别列出，翻译外文文献数量最多不超过3篇。

2.外文翻译内容字体为小四号，宋体（英文需用Times New Roman字体），行距为固定值20磅。

3.外文原文用A4纸打印，附在译文后面，与译文一起装订。