计算机科学2000Vol. 27№. 6

软件体系结构分析及场景技术在其中的应用

Software Architecture Analysis Method and Application of Scenario in SAAM

孙昌爱　刘　超　金茂忠

（北京航空航天大学软件工程研究所　北京100083）

Abstract　A scenario-based Software Architecture Analysis Method (SAAM) is introduced. In this paper, scenario is applied to analyze quality of OO software architecture for the first time, such as coupling, cohesion and complexity. Thereby software architecture can be measured quantiatively while it was too abstract to be measured in the past. SAAM introduced by this paper is demonstrated by applying it to analyze the attribute of software architecture of SafeproC, a test tool for C software. It is proved that SAAM in this paper can effectively measure not only static quality of software architecture, but also dynamic quality of software architecture

Keywords　Software architecture analysis, Software architecture measure, Scenario technology

计算机应用系统的日益复杂和庞大，使得软件体系结构的研究成为当前的研究热点。软件体系结构设计已经成为软件生命周期中的一个重要环节。但是，如果无法对一个软件体系结构进行客观的、可行的定量和定性分析和评价，那么这种软件体系结构是不可靠的。现代软件的需求不断变化、业务规则和新的软件技术变化不停，要求软件体系结构在高层上必须考虑软件进一步演化，才可能为将来的软件体系结构的变更留有一定的余地。因此，软件体系结构的设计在某种程度上利用了软件产品线的思想。

场景技术在软件生命周期中有着广泛的应用。在需求分析阶段，场景可以用来捕获需求和系统的功能[1]。在软件设计阶段，场景还是软件体系结构建模的主要依据，是软件主要行为的集中体现[2]。在测试阶段，场景应该是测试用例产生的主要依据，或者说场景为测试用例的开发提供了很强的指导。在维护阶段，软件的进一步演化以及对现有软件体系结构的分析，场景都是很好的工具[3~5]。本文介绍一种基于场景的体系结构分析方法，试图利用场景技术来分析面向对象软件体系结构的质量特性，这对于结构化的软件体系结构也有一定的借鉴作用。

1　概述

无论软件处于最初的设计、动态配置，还是维护阶段，软件体系结构对于系统的设计和理解非常重要，而场景则是实现一个满足特定质量要求的体系结构的重要工具。本文将场景这一有用的工具融合到了体系结构分析中。

1.1　软件体系结构

软件体系结构描述了组成软件的构件的高层配置以及协调这些构件活动的连接。软件体系结构在软件生命周期中有如下的作用：

1）软件体系结构往往是软件的雏形，表达了所有的软件需求如何实现。体系结构表达的设计决策一但确定下来以后很难改变，因此必须仔细考虑。

2）软件体系结构是成功软件产品线工程的重要原型。软件产品线是指只要花费较少的精力、花销，就可以有规律地开发相似系统的产品谱系，而且这种开发方式所冒的风险比单独开发每一个系统还要少。

3）软件体系结构往往是那些不熟悉某系统的维护程序员工作的切入点，他们通过软件体系结构能了解不熟悉的软件系统。

软件体系结构分析可以及早地发现软件设计中的理解错误，并能降低在整个软件生命周期中有效地修改软件和预测修改所付出的花销，从总体上降低了软件的开发代价。

1.2　质量属性

软件分析和评估是两个不同的概念[6]。分析是指

孙昌爱　博士生，研究兴趣为软件体系结构、构件技术；刘　超　副教授，主要研究方向为软件工程、面向对象技术；金茂忠　教授、博导，主要研究方向为软件工程、软件工程环境。

分解系统，并分析其组成要素、要素之间的联系及其复杂性；而评估则指对系统进行评价（打分）。我们希望通过评估体系结构，决定以此为基础的系统是否满足特定的属性或质量，如修改性、安全性。然而，由于这些抽象的质量太模糊，并且缺少评估体系结构的过程支持，因此分析起来非常困难。

对于现在以及可预见的未来，软件体系结构的质量度量并不存在一个通用的度量方法，必须在具体的执行或开发环境下进行度量才有意义。虽然，我们希望能够有一个更易理解、更通用的质量属性的表达方法，但是目前我们必须在特定的操作环境下识别系统的作用。因此，我们采用场景表达这种上下文相关性。

1.3 场景技术

场景应用广泛，已被认为是一种用于需求提取，特别是系统操作提取的技术。同时，场景还用作一种比较设计方案的方法。但是，场景还没有用作质量分析的工具，我们恰恰利用了这一点：用场景来表述对于用户（或系统）很重要的每一个质量属性的特定实例，通过分析如何满足每个场景所要求的约束来分析软件的体系结构。这里，场景用来简要描述系统预期的或所希望的使用方式。场景适合描述系统中的任何角色，包括操作员、系统设计人员、修改人员、系统管理员和其他的人员。

用场景进行分析的过程迫使设计者考虑系统的将来使用方式、需求的变更。我们真正想知到的是："软件体系结构怎样容纳将来的变化？"或"这种体系结构怎样适应某一类的变化？"。我们用体系结构分析来指导对体系结构的考察，将注意力集中到潜在的变化上。

2 基于场景的软件体系结构层分析方法

一种特定的基于场景的体系结构分析方法称为软件体系结构分析方法（SAAM）[5]，其最初用于体系结构方案的比较。尽管在实际运用 SAAM 的过程中，并没有严格执行该方法描述的步骤，但是场景都被用来作为说明体系结构属性的基础。

SAAM 的工作原理图如图1所示。图1中介绍了五个主要的活动：

1）描述待分析软件的体系结构：体系结构应该以一种（对于分析人员）易于理解的、合乎语法规则的体系结构表示，而且这种表示应能体现系统的计算构件、数据构件以及构件之间的关系（又称为连接件）。

2）场景开发：主要开发一些任务场景，这些场景能够体现系统所支持的各种活动，或者经过一定日程后系统将要进行的各种变化。开发场景的关键是捕获系统重用方式（用例）。所以，场景能够表现与任务相关的各种角色：终端用户/客户、市场、系统管理员、维护人员和开发人员。

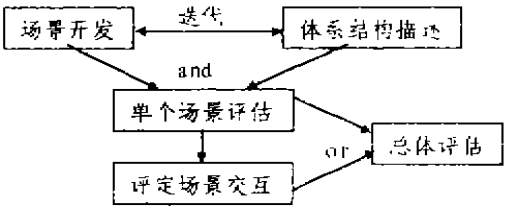

图1 SAAM 的活动过程图

3）场景评估：首先，进行场景分类（直接场景和间接场景）。其次，对于间接场景则要列出体系结构的变化及变化所需的代价。另外，体系结构的变化将要引进一些新的构件及连接件，而构件和连接件的添加又要影响体系结构，所以必须评估这种影响和变化。最后，应生成一个关于特定体系结构的场景描述列表。

4）场景交互：有可能不同的间接场景需要修改同一个构件和连接件，这种情况下，我们说场景与构件和连接件发生了交互。研究场景交互实际上是识别影响一组常见构件的场景的过程。场景交互度量的是体系结构对各个分立主题的支持程度。对于每个构件最好能列出对其可能产生影响的场景。在 SAAM 中，希望体系结构中的场景冲突愈少愈好。

5）总体评估：按照重要性为每个场景及场景交互分派权重，并将这些权重加起来确定总的级别。权重的选择，反映了该场景表现的质量因素的重要程度，对于体系结构影响较大的质量因素，其权重较大，而影响该质量因素的场景和场景交互也应该赋予较高的权重。

描述体系结构和开发场景是互为促进的过程，而且相互依赖。具体说来，场景的开发有助于体系结构的充实和改进，而场景反映的活动又是体系结构能支持的和将要支持的活动。静态场景有助于体系结构的静态模型的开发，动态场景有助于体系结构的动态模型的建立，但对于某个特定的体系结构可能开发出不同的场景集。

3 SAAM 应用实例及场景技术的应用

SAAM 的方法一般用于系统设计的早期，对设计好的软件的体系结构进行基于场景的分析，从而确定所设计的体系结构能否满足所需的质量特性要求。为了验证该方法的有效性及可操作性，本节详细讨论该方法的使用过程和场景技术在其中的应用。

3.1 系统描述

建议尽量以领域术语描述系统的主要功用，应能体现系统中不同角色和相应的使用方式，从而便于系统中主要场景的提取。另外，在系统描述中可以采用用

例分析的技术，提取系统的主要使用场景、任务、角色，从而便于体系结构的描述和场景的开发。

SafeproC 是北航软件所研制多年的"C 软件辅助理解和测试工具"[5]。该工具软件在源程序分析的基础上，提取程序的结构，提供了程序的通用的度量，分析程序的静态数据流，并通过专门的时间分析工具提取 C 语句行的静态时间，并基于流程图进行可视化的时间分析，检测被测程序中的中断函数以及中断安装、中断端口使用。在静态分析的基础上，SafeproC 还支持动态测试(所谓动态测试即是在运行被测程序的基础上进行的测试)，包括断言跟踪、动态数据流跟踪、动态时间检测、语句和分支覆盖率测试，模块动态调用次数。

SafeproC 在软件的开发、测试和维护阶段都提供对软件理解和质量保证的重要支持，而且适用不同的角色。软件开发人员可以利用它进行静态时间分析以期提高对软件的实时性能的信心，通过分析其结构保证程序逻辑的正确性，对于刚加入系统开发的人员，可以通过该工具迅速地理解现有程序的结构；对于测试人员来说，利用该工具可以得到测试的覆盖率，检测程序静态数据流和动态数据流，模块的动态调用次数，断言跟踪；而维护人员则可以利用该工具进行程序结构分析，中断函数及安装检测；系统管理员则需要通过 SafeproC 管理测试用例和打印相应的测试文档。

3.2 开发场景和描述体系结构

在体系结构评估前，我们必须得到软件的体系结构层表示。其中，每个组件或连接件必须都有一个比较具体的语义解释。一般说来，现行的软件不具有直接的体系结构表示。因此，必须在分析之前利用逆向工程的方法提取软件的体系结构层的表示。其方法是充分利用已有的文档、源代码、用户手册、设计手册、实例产品以及设计人员充分的交互以理解设计的思想。

具体说来，体系结构的获取是一个渐进的过程，也是一个迭代的过程。在其每个阶段中，我们研究产品、产品文档和已经提炼到体系结构的描述，并不断地提出问题，形成问题列表。对于问题的回答则有助于澄清现行的软件体系结构描述。这样，随着阶段的不断推进，我们对问题的理解越来越深入，越来越全面，描述的体系结构信息也愈来愈具体。

一般地，通常进行三到四次的迭代，便可以得到一个满足体系结构评估的表示。另外，在场景开发过程中，应该尽可能考虑到系统中的角色，比如用户、开发者、维护人员、系统管理人员。这对于从不同的角度考虑系统的体系结构非常有益，因为通过与这些角色的讨论可以尽可能地开发出系统中必要的场景，而且保证了场景的正确性。进一步地，便于开发出系统的人员组织结构视图，从而从资源管理上提供了参考信息。

根据系统的描述，以及应用上述的过程，我们可以得出 SafeproC 的场景清单，并且是按角色分类的。

开发人员：1)进行静态时间分析以期提高对软件的实时性能的信心；2)分析程序结构保证程序逻辑的正确性；3)可以通过该工具迅速地理解现有程序的结构(局部和宏观的)；4)分析程序的复杂度。

测试人员：1)得到测试的覆盖率(分支和语句)；2)检测程序静态数据流；3)检测动态数据流；4)检测断言跟踪；5)测试模块的动态调用次数。

维护人员：1)进行程序结构分析；2)中断函数及安装检测。

系统管理员：1)管理测试用例；2)打印相应的测试文档。

另外，SafeproC 应该支持中英文界面的需求，这当然是所有色角都要求具备的系统任务。

通过三次的迭代过程，我们得到了关于 SafeproC 在体系结构层上的表示(如图2所示)。由于程序规模较大，结构比较复杂，我们只列出其静态分析的体系结构层描述部分，对于场景来说主要限于开发人员这一角色。

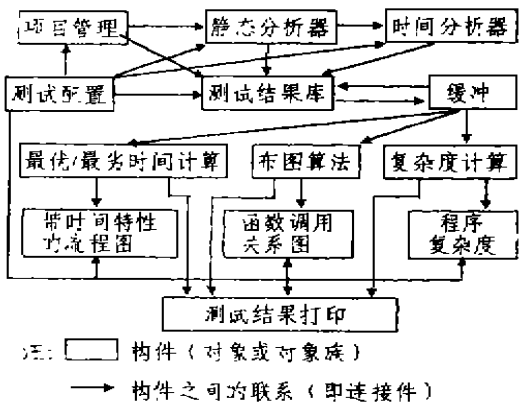

图2 Safepro 部分体系结构描述

3.3 场景评估

首先，对场景进行分类，将其分为直接场景和间接场景。所谓直接场景，是指开发的系统已能满足的场景，而间接场景则是指需要对现有的体系结构中的构件和连接件做适当的变化才能满足的场景。

对于间接场景，则要考虑体系结构为了满足这种任务必须作的变化，以及这种变化难易程度、实现的代价(以人时/代码行计)。在 SafeproC 中已经支持所有的英文界面的场景，而支持中文化界面则是一间接场景。

此时，我们应该列出已开发出的场景对体系结构

中的构件和连接件的影响的列表。

表1 构件及构件变更情况列表

构 件	影响数
测试配置	8
测试结果库	6
缓冲	5
静态分析器	4
……	

实际上，构件的变化影响数表现了构件与其他构件的耦合，对于那些耦合较高的构件，在系统设计时必须重点考虑。

3.4 场景交互

当两个间接的任务必须影响系统中的一些构件时，则称这些场景之间存在一定的交互。场景交互非常重要，因为它在一定程度上表达了产品设计的功能分配，以一种非常清晰的方式显示了模块的不同本质。场景交互愈高则说明特定的构件内部的模块封装特性较差，同时也为设计人员在以后的设计过程中提供了注意点。当然，场景交互的数量也在一定程度上与度量之间存在一定的联系，同时它还与产品最终的缺陷数有着很强的关系。

我们认为：在探究场景交互的过程中，应能分析得出系统中所有的场景对系统中的构件产生的影响的列表，最后应研究如何以可视化的方式表现这种结果，比如以不同的颜色、大小，并结合构件之间的交互，形成系统的场景交互图。场景交互图实际上是上层的场景分析而得出的场景关联。其中，一些特定的构件集完成某一场景任务，而场景任务集又可能对某些构件有不同的修改要求，从而形成场景交互。

3.5 总体评估

首先开发出所有的场景并将其映射到结构层的描述。这里的映射方法对于结构化软件和面向对象软件之间有一定的差别。然后，尽可能地研究场景之间的关系（即探究场景交互，对于体系结构的变迁和演化在前期做好其在体系结构上的考虑）。分别对各个场景及场景交互进行体系结构度量（内聚性、耦合性、体系结构复杂度 MaCabe 和 Halstead），并根据每个场景在体系结构层的重要性分配相应的权重（优先级），最后综合场景权重和场景度量得出体系结构的最终得分，给出定性结论，指出不足之处。当然，每个场景的优先级必须由用户自己把握，而且对于可复用的场景在不同的系统中的重要性也可能不同。

按照以上的分析方法，我们发现 SafeproC 的软件体系结构的内聚性较高，构件之间的耦合度较低，但程序结构相对比较复杂。总体上软件的体系结构比较合理。在此基础上进行系统的演化还是可取的。在最终的实践中，我们的实验结果完全符合我们的理论预估，在界面汉化的实现过程中，我们仅花了五个人天的花销便实现了汉化的需求。

4 方法应用总结

通过对 SaferpoC 的体系结构进行分析，我们渐渐地领会了软件体系结构分析的模式，并且体会到进行软件体系结构分析可以使以后的软件过程取得更好的结果。

4.1 开发团体应充分认识 SAAM

体系结构分析过程有助于我们将注意力集中到体系结构比较重要的细节上，而忽略一些次要的细节，同时，场景的应用有助于小组开发人员之间以及开发人员与管理人员之间的通讯。场景的使用过程是：提炼体系结构描述，提问，提炼分析。当然，不同的开发人员可能提炼到不同的软件场景，因此，究竟什么是合适的场景集无法定论。但场景开发的过程应该集中在系统的主要架构上，而且最终应达成一致。那些能够发现体系结构不足的场景对于以后的软件的进化将起到非常重要的作用。

4.2 SAAM 和传统的体系结构度量

体系结构度量与传统的设计概念（内聚和耦合）之间的关系非常紧密。从功能分解的角度看来，一个好的体系结构应该具备很高的内聚性和较低的耦合性。从体系结构分析看来，低耦合意味着某个场景不应影响太多的结构化的构件，高内聚性则要求一个结构化的构件不要牵涉太多的场景交互。体系结构分析与传统的设计概念的巧合，实际上说明了体系结构可以在高层上直接分析内聚和耦合性。

本文提出的 SAAM 方法通过度量特定的场景或场景集内聚和耦合来改进体系结构的原始度量。例如，原始度量中，耦合解释为：只要两个模块是耦合的，则不管它们之间通讯是一次还是多次。这种复杂度量不考虑体系结构的变化，所以不能很好地体现体系结构的动态演进。

4.3 确定体系结构描述的层次

软件体系结构的最大益处在于从高层上抽象软件实体，这就意味着体系结构图要有用的话，必须选择合适的描述层次。在获得体系结构的初始结构化描述之后，需要将场景映射到结构上。特别对于那些间接场景，则应加亮间接场景将要影响的构件和连接件。我们之所以对间接场景感兴趣是因为间接场景表达了体系结构将要满足的额外功能。直接场景及其交互则暗示了构件潜在的复杂度。

将场景映射到结构层的描述上有两个目的：指导体系结构的评估过程，并且确认场景交互。在此过程中，可能遇到多个间接场景都影响同一个模块（场景交

互)。这意味着:

1)这些场景是同一类的场景,也就是说是同一基本场景的不同变种。同一类场景群束在同一个构件下被认为是一个好的设计,这也预示着系统的功能分配比较合理。换句话说,这一类场景在体系结构方面的内聚性比较高。

2)场景交互可能意味着场景是不同类的,而且构件可以进一步划分,但是在原始的体系结构描述中没有这种划分。正如我们在前面讨论时,体系结构的描述层次是否合适是由场景描述决定的,即以出现尽可能少的场景交互为原则。

3)交互的场景可能属于不同的类,而且不可进一步划分。这样的话,则预示着软件体系结构在这个领域上存在潜在的问题,因为不同类的场景却影响着同一个构件。

上面讨论的场景交互体现的内涵,不仅指出场景交互对于体系结构分析的意义,实际上也说明了体系结构的描述和表示必须遵循的一些原则。这正是基于场景的软件体系结构分析方法的真正的动机,也是场景描述在体系结构分析中的真正意义所在。

4.4 决定场景集

在基于场景的软件体系结构分析方法中,场景是核心。那么,对于一个软件系统来说,到底该有多少场景才可以充分分析软件的体系结构呢?我们认为:新场景的开发已经无益于设计,则应该停止开发新场景。这就和测试一样,无论什么时候,都不能说测试用例已经非常充分了,但是在新的测试用例加入且已经无法改进软件时,我们可以将其定为一个测试充分点。减少场景的数量的一个有效方法是进行场景的等价类划分。

4.5 进一步的工作

基于场景的软件体系结构分析方法充分地利用了场景的上下文相关性,从场景的开发过程中不断地提取软件的体系结构信息,形成了一系列的场景交互。通过场景和体系结构中的构件耦合进行相应的场景评估,根据体系结构中质量因素的重要程度分配相应的权重,最终得出体系结构的评估结果。按照这样的思路,基本上可以比较客观地评价待分析的软件的体系结构的静态质量属性,而且还可以有效地度量其动态质量属性,即进一步演进的可行性及代价。

但该方法的某些活动上的不确定性使得可操作性存在一定的局限,如场景集的确定会因人而异,从而评价结果很可能不一致。另外,总体评估未能给出较通用的数学模型(和质量度量模型吻合),使得执行该方法时缺少必要的尺度。这些正是作者进一步研究的兴趣所在。

参 考 文 献

1 Dardenne A. On the Use of Scenarios in Requirements Acquisition: [CIS-TR-93-18]. Department of Computer and Information Science, University of Oregon, 1993

2 Kruchten P B. The 4+1 view model of architecture. IEEE Software, 1995,12(6):42～50

3 Kazman R, et al. Scenario-Based Anslysis of Software Architecture. IEEE Software, 1996(Nov.):47～55

4 Carriere J, Kazman R. SEI CMU, Assessing Design Quality From a Software Architectural Perspective. Available at: http://www.sei.cmu.edu/publications/articles/

5 Kazman R, et al. SAAM: A Method for Analyzing the Properties of Software Architecture. Available at: http://www.sei.cmu.edu/publications/articles/

6 Brown A, et al. A Case Study in Assessing the Maintainability of a Large, Software-Intensive System. In: Proc. of the Intl. Symposium on Software Engineering of Computer Based Systems, Tucson, Az., IEEE Computer

7 SafeproC 用户手册. 北航软件所

(上接第94页)

但该集成模型也存在一些缺点。虽然该模型详细地考虑了两种模型集成的各种情况,但没有考虑该模型对系统性能的影响,许多地方还有待进行优化,因此,以这种模型为基础的应用系统效率不高,而且由于该模型考虑的主要是一些通用的情况,对于一些特殊的应用背景未做考虑,因此,对于真正的应用系统的开发,还需要许多额外的工作。

参 考 文 献

1 Adam N R, Gangopadhyay A. Database issues in Geographic information systems. Kluwer Academic Publishers. 1997

2 Chrisman N R. Concepts of space as a guide to cartographic data structures. In: Proc. of the first interl. advanced study symposium on topological data structures for geographic information systems. 1978. 1～19

3 Couclelis H. People manipulate objects (but cultivate fields): Beyond the raster-vector debate in GIS In: Frank A U, et al. eds. Theories and Methods of Spatial-Temporal Reasoning in Geographic Space, Berlin: Springer-Verlag, 1992. 65～77

4 Egenhofer M J, Herring J R. High-level spatial data structures for GIS In: Maguire D J, et al. eds. Geographical Information Systems. Principles and Applications, Longman. 1991. 227～237

5 Ishikawa H, et al. An object-oriented database system jasmine: Implementation, Application, and Extension. IEEE Transactions on Knowledge and Data Engineering, 1996, 8(2)

6 Kemp K K. Environmental modeling with GIS. A strategy for dealing with spatial continuity In. Proc. of GIS/LIS Annual Conf. Bethseda, MD: ASPRS and ACSM, 1992. 397～406

7 Peuquet D J. A conceptual framework and comparison of spatial data models. Cartographic, 1984, 21(4): 66～113

8 Peuquet D J. Representation of geographic space: toward a conceptual synthesis, Annals of the Association of American Geographers, 1988. 375～391

9 Roberts S A, Gahegan M N. An Object-Oriented Geographic Information System Shell. Information and Software Technology, 1993, 35(10): 561～572

10 Samet H. The Design And Analysis Of Spatial Data Structures, Addison-Wesley. 1990

11 Tomlin C D. Geographic Information Systems and Cartographic Modeling, Prentice-Hall. 1990

12 Worboys M F. GIS: A Computing Perspective, Taylor & Francis, 1995