基于场景的两种软件体系结构评估方法*

沈群力[1,2], 刘 杰[2]
(1. 上海商学院, 上海 200235; 2. 复旦大学 管理学院 上海, 200433)

摘 要: 基于场景的软件体系结构评估方法通过场景将系统的质量需求转换为风险承担者与系统的交互活动, 通过低代价的事前评估活动降低软件的质量风险。SAAM 及 ATAM 两种方法在具体的实际评估活动中, 它们在场景的生成、风险承担者的商业动机的表述、软件体系结构的描述等方面存在着很大的不同, 两种评估方法各有特长, 其评估方法在具体的场景执行环节上具有不确定性, 将定量的度量方法与定性场景结合是提高评估的有效途径。

关键词: 场景; 软件体系结构; 软件体系结构分析法; 体系结构权衡分析法; 效用树
中图分类号: TP311.5 **文献标志码**: A **文章编号**: 1001-3695(2008)10-3015-03

Two software architecture evaluation methods based on scenario

SHEN Qun-li[1,2], LIU Jie[2]
(1. *Shanghai Business School, Shanghai* 200235, *China*; 2. *School of Management, Fudan University, Shanghai* 200433, *China*)

Abstract: Software architecture evaluation methods switch quality demand to interaction between risk-receivers and the system. Advancing evaluation could reduce quality risk when software was designed. SAAM and ATAM were two different evaluation methods which have their merits when applied in creating scenario, describing business motivation and software architecture and so on. Neither methods can't ensure certainty when applied in particular performance, so it is an effective way to combine quantitative methods and scenario qualitative evaluation.

Key words: scenario; software architecture; software architecture analysis method; architecture tradeoff analyis method; utility tree

软件体系结构是指组成该软件系统的一个或多个结构, 它们构成软件的各个部分, 形成这些组件的外部可见属性及相互关系[1]。在大型复杂软件系统的设计中, 软件体系结构是软件设计阶段的早期产物, 它决定着软件产品能否顺利开发及最终软件产品的质量。因此对软件体系结构的分析评估就是事先通过代价低廉的评估活动来识别软件结构中存在的潜在风险, 降低项目开发中的不确定性, 找出软件体系结构中影响系统质量的主要因素及改进措施, 并在此基础上检验软件的质量需求是否在具体设计中得到体现, 并预测未来软件质量。

目前, 最著名并被广泛接受的基于场景的软件体系结构分析评价方法是由 SEI(Software Engineering Institute) 首先提出, 其最具代表性的方法便是 SAAM(software architecture analysis method, 软件体系结构分析法) 和 ATAM(architecture tradeoff analysis method, 体系结构权衡分析法)。两者都是通过分析软件体系结构对场景(系统的使用或修改活动) 的支持程度来判断该体系结构在多大程度上满足该场景所代表的质量需求。

1 场景内涵

软件体系结构评估中, 评估人员关注的是软件系统质量, 这些质量可用性能、可靠性、可用性、安全性、易修改性、功能性、可变性、集成性、互操作性等相关属性来表示。为达到目的, 评估人员首先要精确地提出具体质量指标, 并以这些质量指标作为软件体系结构优劣的评估标准, 为达到这一目的所采用相关机制方法便是场景。场景是软件体系结构分析评价中常用的一种技术, 由用户、外部激励等初始化, 场景包括系统中的事件和触发该事件的特定激励。

基于场景的评估方式就是将对系统质量需求转换为一系列风险承担者与系统的交互活动, 分析软件体系结构对这一具体活动的支持程度。在评估过程中将考虑所有与系统相关人员(包括风险承担者) 对质量的需求, 从而确定应用领域功能与软件体系结构之间的映射。在此基础上设计用于体现待评估质量属性的场景以及软件体系结构对场景的支持程度。例如用一系列对软件的修改来反映易修改性方面需求, 用一系列攻击性操作来代表安全性方面需求等。

2 基于场景评估方式的两种代表性方法简述

2.1 SAAM——软件体系结构分析法

SAAM 是一种相对简单的软件体系结构评估方法。最初用来分析软件体系结构的可修改性, 后来实践证明该方法不仅可用于可移植性、可修改性、可扩充性、可集成性等质量属性及系统功能进行快速评估, 还可用于对性能、可靠性等其他质量属性的启发式评估, SAAM 是最早形成文档并得到广泛使用的软件体系结构分析方法。具体评估步骤如图 1 所示。

SAAM 的评估包括六个活动, 分别为: a) 场景的生成, 风险承担者集体讨论, 反映自己的需求; b) 软件体系结构描述, 采用某种参评各方都能够充分理解的形式对即将评估软件体系

收稿日期: 2008-01-17; 修回日期: 2008-03-24 基金项目: 上海市政府重点学科建设基金资助项目(沪教委科(2006) 47)
作者简介: 沈群力(1973-) , 男, 安徽青阳人, 博士研究生, 主要研究方向为信息管理与信息系统(shenqunli@163.com) ; 刘杰(1964-) , 男, 江苏扬州人, 教授, 博导, 主要研究方向为信息管理与信息系统 .

结构描述; c) 场景分类并确定优先级, 保证在有限资源下优先考虑最重要的场景; d) 对间接场景的单个评估; e) 场景相互作用的评估, 即多个场景要求更改体系结构的同一组件; f) 总体评估, 根据相对重要性来确定每个场景的权重, 通过权重得出总体评价。

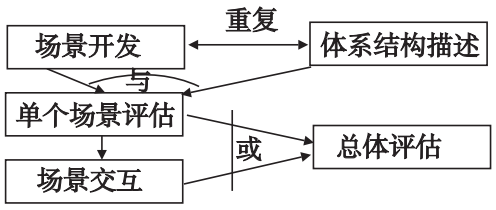

图1 SAAM的评估步骤

SAAM不仅可以对单个构架评估, 还可以对多个构架进行比较。不仅可以促进软件构架文档质量的提高, 还可以促进参与评估的风险承担者之间交流沟通。

2.2 ATAM——体系结构权衡分析法

ATAM 方法是 SEI 于 2000 年在 SAAM 方法基础上提出的, 它是考虑了可修改性、性能、可靠性和安全性等多种质量属性的软件体系结构评价方法, 并能确定这些相互制约属性之间的折中点。ATAM 方法包括四大部分, 共计九个步骤: a) 表述, 通过表述进行信息交流, 包括 ATAM 方法表述、商业动机表述、软件体系结构的表述, 共三个步骤。b) 调查分析, 包括确定软件体系结构的方法、生成质量属性效用树、分析软件体系结构方法, 共三个步骤。c) 测试包括集体讨论并确定场景的优先级、分析软件体系结构方法, 共两个步骤。d) 形成报告, 即评估结果的表述。

ATAM 通过用调查表来收集影响软件体系结构质量属性的要素, 描述质量属性的特征, 尽管九个步骤按编号排列, 但并不一定严格遵照这种瀑布模型, 评估人员可以在九个步骤中进行跳转和迭代。ATAM 能针对性能、实用性、安全性和可修改性这些质量属性, 在系统开发之前对其重要性进行评价和折中。当评估活动结束后, 将评估的结果与需求作对比, 如果系统预期行为与需求充分接近, 设计者就可以继续进行更高级别的设计或实现; 如果分析发现了问题, 就对所设计的软件结构、模型或需求进行修改, 从而开始迭代过程。

3 两种方法具体应用中的不同性比较

基于场景的软件体系结构评估方法用场景代替对质量属性的空洞描述, 提高软件测试的可行性。在具体的实际运用中, 两种方法又有各自不同的特点。

3.1 场景的生成方式不同

SAAM 方法采用头脑风暴(brainstorming) 技术构建场景, 要求风险承担者列举出若干场景, 并将场景分为直接场景和间接场景两类, 分别支持对体系的静态分析和动态分析。直接场景类似用例, 按照现有的构架开发出来的系统能够直接实现的场景。间接场景, 有时也叫更改案例, 在实际评估中, 如果软件体系结构不能直接支持某一直接场景, 就需要对现有的软件体系结构作修改, 对现有构架作一些修改才能支持的场景便是间接场景。评估中对于直接场景, 软件体系设计师需清楚表述该体系结构将如何执行这些场景; 对间接场景, 设计师应说明需要对体系结构作哪些修改才能适应间接场景的要求, 并估计出这些更改的代价。通过综合各间接场景对软件体系结构的影响, 可以测试出软件在生命周期内的适应性和可更改性。在具体评估中并给场景设置优先级别来保证在有限的时间内最先考虑最重要的场景。

ATAM 在具体评估中将场景分为三类: a) 用例场景(use case scenario) 。描述用户的期望与正在运行的系统交互, 用于信息的获取。b) 生长场景(growth scenario) 。预期的系统变更与质量属性关系。c) 探索场景(exploratory scenario) 。暴露当前设计的极限或边界, 显示可能隐含的假设, 即弄清这些更改的影响。场景的具体使用中评估小组请风险承担者对三类场景进行集体讨论, 根据风险承担者的意见将代表相同行为或相同质量属性的场景进行合并, 确定其中若干个场景, 最后通过投票的方式来确定这些场景的优先级别。这就要求软件设计师必须清楚地向风险承担者表达软件结构设计中所使用任何明确的体系结构方法, 风险承担者根据确定使用的体系结构方法, 采用 “刺激—环境—响应 ”来生成三类场景。

3.2 风险承担者商业动机表述方式不同

软件体系结构的评估参与人员主要有风险承担者及评估小组成员, 不同的风险承担者对软件的质量属性有着不同的组织目标, 不同的组织目标是由商业动机的出发点不一样而造成的, 在软件体系结构阶段就应该考虑他们的商业动机及目标满足程度。

SAAM 只是将不同组织的需求目标简单地联系在一起, 评估中由开发人员、维护人员、用户和管理人员等风险承担者将其所关心的目标问题以场景形式提出, 这样不同领域的风险承担者根据自己领域特点及对软件的需要确定若干场景, 评估小组根据他们确定的场景来评估软件系统特定方面的性能。对同一系统, 由于不同风险承担者确定的场景不同, 评估的结果也将不同。这种将商业动机不经过深入的数据处理就转换为场景的评估方式具有一定的不精确性, 只能是一种粗糙的评估方式。

ATAM 建立在 SAAM 的基础上, 借助于效用树(quality attribute utility tree) 将风险承担者的商业目标转换成质量属性需求, 再转换成代表自己商业目标的场景。对风险承担者不同的商业动机, 评估人员首先确定不同商业动机所代表的质量属性与其他质量属性比较, 得到相对重要性的矩阵, 并运用层次分析法来确定质量属性的优先组合, 将模糊的目标定量化, 生成质量属性效用树, 这样评估人员更容易关注效用树的叶节点, 更好地满足处在高优先级位置的场景而需采用的软件体系结构方法, 更为精确地定义质量需求。效用树的理论基础是管理学中的 “需要理论 ”, 即通过刺激及其所产生期望响应来描述场景, 根据期望的迫切程度确定场景优先级别, 在理想情况下, 所有场景都以 “刺激—环境—响应 ”的形式表述。ATAM 通过不同质量属性之间的交互及依赖关系, 寻求不同质量属性之间的折中机制, 因此 ATAM 方法在具体评估分析过程中要求记录系统风险、找出敏感点(sensitivity points) 及折中点(tradeoff points) , 同时也改进了体系结构的相关文档。

3.3 软件体系结构的描述方式不同

在评估之前, 首席软件设计师需要对软件体系结构作详略适当的讲解, 这种信息讲解的表达透彻程度将直接影响体系结构的分析质量。在讲解过程中, 评估小组还要询问更多相关信息, 只有选择合适的视图才能更清晰地表达相关信息。

ATAM 方法中软件体系结构的描述采用 Philippe Kruchten “4 + 1 ”视图模型, 即从五个不同的视角点描述系统的体系结

构, 四个视图模型从特定的不同方面描述软件的体系结构, 忽略与此无关的实体。描述系统的功能需求采用逻辑视图, 即系统提供给最终用户的服务; 描述系统的运行特性采用进程视图, 侧重关注非功能性的需求(如性能、可用性) , 该视图服务于系统集成人员, 方便后续性能测试; 硬件配置采用物理视图描述, 该视图服务于系统工程人员, 解决系统的拓扑结构、系统安装、通信等问题; 软件模块的组织与管理采用开发视图描述, 该视图服务于软件编程人员, 方便后续的设计与实现; 最后用场景视图刻画构件之间的相互关系。将四个视图有机地联系起来, 场景视图不仅可以描述一个特定的视图内的构件关系, 而且可以描述不同视图间的构件关系。在四个视图中, 逻辑视图、开发视图主要用来描述系统的静态结构; 进程视图、物理视图主要用来描述系统的动态结构。ATAM 在实际运用中并非每个系统都必须将五个视图都画出来, 而是各有侧重。

SAAM 则提倡使用非常单纯的体系结构要素。对体系结构的静态描述一般都要区别数据的连接(数据在组件之间的传递) 和控制连接(一个组件调用另一个组件执行某个功能) ; 对软件体系结构的动态描述主要描述系统在各个不同时间的行为, 给出软件的体系结构。软件体系结构的描述既可以用自然语言来描述系统的整体行为, 又可以采用某种形式的结构化的描述, 表述形式相对灵活。

4　结束语

SAAM 与 ATAM 方法相比较, 是一种相对简单的软件体系结构评价方法, 进行培训和准备的工作量较少。尽管 SAAM 评估步骤及细节较少, 但是涉及的内容更多, 在评估中没有提供体系结构质量属性的清晰度量, 评估过程过于依赖专家经验。总体评估时还需根据场景对系统功能的相对重要性设置权重, 确定总体评价, 权重的设置具有很强主观性, 故 SAAM 方法适于体系结构的粗糙评价, 用于边设计边评估的软件体系结构开发过程。

ATAM 方法是目前被验证有效并广泛使用的一种体系结构评估方法, 但没有对质量属性作深入分析, 同样缺少定量数据来支持分析结果。ATAM方法尽管揭示了体系结构如何满足特定的质量目标, 而且还提供了这些质量是如何交互的, 即它们之间如何权衡, 但这些设计决策将影响整个软件生命周期, 且软件实现后很难修改这些决策的影响。

目前对体系结构分析与评价大多采用基于场景的技术, 这种技术存在着一定的不确定性。如何根据某质量特征使确立的场景更有代表性和完备性, 如何为构造的场景确定合适的边界条件, 即停止准则等。这些在具体应用中都有很大的主观性及经验性, 而度量的评估方法在这方面做得较好, 将定量的度量方法与定性场景结合是提高软件体系结构评估的有效途径。

参考文献:

[1]　ASS L, CLEMENTS P, KAZMAN R. Software architecture in practice [M] . Reading, Massachusetts: Addison-Wesley, 2002.

[2]　张友生. 软件体系结构. [M] . 2 版. 北京: 清华大学出版社, 2006: 256-279.

[3]　刘霞, 李明树, 王青, 等. 软件体系结构分析与评价方法评述[J] . 计算机研究与发展, 2005, 42(7) : 1247-1254.

[4]　沈群力, 刘渊. 微支付系统的解决方案研究[J] . 安徽大学学报, 2002, 26(3) : 29-31.

[5]　沈群力, 唐思章, 刘杰. 基于微支付的网络认证协议[J] . 上海商学院学报, 2006, 7(1) : 27-30.

[6]　沈群力. 信息系统分析与设计实验教学探讨[J] . 实验科学与技术, 2007, 5(2) : 84-86.

[7]　GRIMANA A, PEREZA M, MENDOZA L, *et al*. Feature analysis for architectural evaluation methods[J] . The Journal of Systems and Software, 2006, 79 (2) : 871-888.

[8]　OLUMOFIN F G, MISIO V B. A holistic architecture assessment method for software product lines [J] . Information and Software Technology, 2007, 49(4) : 309-323.

[9]　LOPEZ M. Application of an evaluation framework for analyzing the architecture tradeoff analysis method^{SM} [J] . The Journal of Systems and Software, 2003, 68 (12) : 233-241.

[10]　ARES J, GARCIA R, JURISTO R, *et al*. A more rigorous and comprehensive approach to software process assessment[J] . Software Process: Improvement and Practice, 2000, 5 (1) : 3-30.

[11]　CLEMENTS P, KAZMAN R. Valuating software architecture methods and case studies [M] . [S. l.] : Addison-Wesley, 2002: 235-241.

(上接第 3011 页) 关理论, 对面向可信软件的风险管理模型进行研究。提出基于 BBN 的软件项目风险管理系统框架和影响软件可信性的风险因素 BBN 分析方法, 采用 BBN 对影响软件可信性的风险进行分析评估。在风险分析、风险评价的基础上, 建立基于多维约束满足的 HTOEP 风险控制模型, 从环境、人因、技术、组织、过程多角度控制可信软件开发过程中的风险。通过面向可信软件的风险管理研究为软件产品质量的提高, 软件可信性的实现提供新思路。

参考文献:

[1]　ARR M J, KONDA S, MONARCH I, *et al*. Taxonomy based risk identification, Technical Report CMU/ I-93-TR-6 [R] . Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, 2002.

[2]　BOEHM B W. Software risk management: principles and practices [J] . IEEE Software, 1991, 8(1) : 32-41.

[3]　CHARETTE R. Software engineering risk analysis and management [M] . New York: McGraw Hill, 2005.

[4]　FRIEDMAN N. A Bayesian approach to structure discovery in Bayesian networks[J] . Machine Learning, 2003, 5(1-2) : 95-125.

[5]　冯楠, 李敏强, 寇纪淞, 等. 基于贝叶斯网络的软件项目风险分析过程[J] . 计算机工程与应用, 2006, 42(18) : 16-18, 39.

[6]　HALL E M. Managing risk-methods for software system development [M] . [S. l.] : Addison-Wesley, 2002: 89-99.

[7]　HOUSTON D X, MACKULAK G T, COLLOFELLO J S. Stochastic simulation of risk factor potential effects for software development risk management [J] . The Journal of Systems and Software, 2001, 59(3) : 247-257.

[8]　HAN Wen-ming, HUANG Sun-jen. An empirical analysis of risk components and performance on software projects[J] . The Journal of Systems and Software, 2007, 80(1) : 42-50.

[9]　OSMUNDSON J S, MICHAEL J B, MACHNIAK M J, *et al*. Quality management metrics for software development [J] . Information & Management, 2003, 40(8) : 799-812.