一种软件体系结构设计决策的建模工具

肖 赛 崔晓峰 孙艳春 黄 罡
(北京大学信息科学技术学院软件研究所 北京 100871)
(高可信软件技术教育部重点实验室 北京 100871)

摘 要 体系结构设计在整个软件生命周期中起到关键作用,而设计知识的蒸发会导致系统演化花费代价高、涉众之间交流出现障碍、体系结构制品的复用受到限制等问题,为此需要在软件体系结构层次对设计决策进行显式化的建模。基于一种以决策为中心的体系结构设计方法,实现了一个软件体系结构设计决策的建模工具。该工具帮助架构师对体系结构设计中的问题、方案、决策、理由等核心概念进行建模,完成从需求到体系结构的设计过程,并实现了自动化的候选体系结构方案的合成和部分设计理由的捕捉。该工具还提供了体系结构设计模型与设计决策之间的相互追踪性,以及帮助实现体系结构设计过程中设计决策知识的复用。

关键词 软件体系结构,软件体系结构设计,软件体系结构设计决策
中图法分类号 TP311 **文献标识码** A

Modeling Tool for Software Architecture Design Decisions

XIAO Sai CUI Xiao-feng SUN Yan-chun HUANG Gang
(Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)
(Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing 100871, China)

Abstract Architectural design plays a crucial role in the whole lifecycle of software. The vaporization of the design knowledge causes a lot of problems, for instance, the cost of evolution will be huge, the communications among stakeholders will be difficult, and the reuse of software architectural artifact will be limited. Therefore it is a demanding trend to explicitly modeling the design decisions at architectural level. We developed a modeling tool for software architecture design decisions, based on a decision-centric software architecture design method. The tool helps architects model the core notions of issue, solution, decision, and rationale in architecture design, accomplish the design process from requirements to architectures, and implements the automated synthesis of candidate architecture solutions and capture of partial design rationale. Furthermore, the tool provides tracing between architecture models and the design decisions, and helps to implement the reuse of design decision knowledge during the architecture design processes.

Keywords Software architecture, Software architecture design, Software architecture design decisions

1 引言

软件体系结构(software architecture)已经越来越受到研究者和实践者的重视,并成为软件工程的一个重要的研究领域。在软件开发过程中,软件体系结构提供系统的高层抽象、支持开发人员之间的交流、支持软件复用等[1],因而体系结构的设计在软件生命周期中起到关键作用。体系结构设计本身是一个不断做出决策的过程,产生大量的推理信息。当前的问题在于仅仅关注体系结构设计的结果,而设计中的决策信息会随着时间的推移逐渐丢失,即知识蒸发(Knowledge Vaporization)[2]。

体系结构知识的蒸发会导致一系列后果[3]:(1)系统演化的代价:设计者为了满足新需求做出的决策很可能违反、覆盖、忽略原来的决策,因为他们并不能有效地识别原来决策的存在。(2)涉众(stakeholders)之间交流的障碍:不同的涉众具有不同的背景,对于体系结构编档需要记录的关注点也有所不同。如果体系结构决策没有显式化,涉众即便共享了体系结构制品也并不清楚体系结构背后的理由,难以解决冲突和达成一致的目标。(3)体系结构制品复用的困难:当体系结构决策都隐藏在体系结构中没有显式表示时,就无法有效实现体系结构的复用。

目前的软件体系结构建模中,典型地使用构件/连接子模型对体系结构进行描述。这样的体系结构模型可以清晰表达软件体系结构设计的结果,但是这种仅关注制品而未能显式

到稿日期:2009-01-16 返修日期:2009-03-02 本文受国家重点基础研究发展计划(973计划)项目(2002CB312000),国家自然科学基金项目(60503028),国家高技术研究发展计划(863计划)项目(2007AA01Z127, 2007AA010301)资助。

肖 赛(1984-),女,硕士研究生,主要研究方向为软件工程、软件体系结构,E-mail:xs.xiaosai@gmail.com;崔晓峰(1971-),男,博士研究生,主要研究方向为软件工程、软件体系结构;孙艳春(1970-),女,博士,副教授,主要研究方向为软件工程、软件复用及构件技术、计算机支持的协同工作;黄 罡(1975-),男,博士,副教授,主要研究方向为软件工程、分布式计算、中间件。

表达设计决策及其相关理由的方式还不能让涉众充分理解设计，即常常存在“为什么要设计成这样?”、“为什么某种设计未被采用?”、“能否对当前的设计进行某些改动而不破坏整体的一致性?”等等疑惑。可见，在构件/连接子模型中缺少对设计决策的“一阶(first-class)”表示，是引起设计知识蒸发的主要原因。

本文基于一种以决策为中心的体系结构设计方法[4]，实现了一个软件设计决策的建模工具。该工具帮助架构师对体系结构设计中的问题、方案、决策、理由进行显式化的建模，完成从需求到体系结构的设计过程，并在此过程中实现了自动化的候选体系结构方案的合成和部分设计理由的捕捉。该工具还提供了体系结构设计模型与设计决策及相关方案、问题之间的相互追踪性，并帮助实现体系结构设计过程中设计决策知识的复用。

本文第 2 节是方法概述；第 3 节介绍工具的设计与实现；第 4 节通过一个指挥显示系统进一步对工具的应用进行说明；最后总结全文并给出未来的工作。

2 方法概述

提出了一种以设计决策为中心的体系结构设计方法[4]，对体系结构设计中的决策等核心元素进行建模，并提供了一个半自动化的体系结构设计过程。

2.1 元模型

设计决策的元模型(如图 1 所示)包含了几个重要概念：(1)问题(issue)，作为需求与设计之间的一个桥梁，是从体系结构设计角度考虑的需求，一个问题关注于一个或者多个需求；(2)方案(solution)，分为问题方案(issue solution)和体系结构方案(architecture solution)，分别是对于单个问题的解决方案和对于整个体系结构的解决方案；(3)决策(decision)，是对一个方案的选择结果，即采纳或放弃。决策也分为：对问题方案的决策，称为问题决策(issue decision)；对体系结构解决方案的决策，称为体系结构决策(architecture decision)；(4)理由(rationale)，是决策背后的原因，也分为问题理由(issue rationale)，即问题决策的原因，和体系结构理由(architecture rationale)，即体系结构决策的原因。

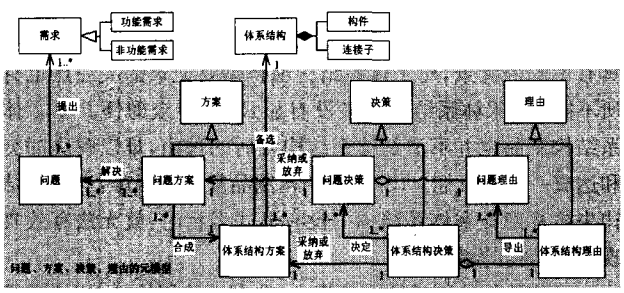

图 1 设计决策元模型

2.2 设计过程

以设计决策为中心的体系结构设计过程(如图 2 所示)以软件需求和(可能的)原始体系结构为输入，经过问题导出(issue eliciting)、方案发掘(solution exploiting)、方案合成(solution synthesizing)和体系结构决策(architecture deciding)4 个步骤，得到决策的体系结构解决方案，并且通过一个理由捕捉(rationale capturing)活动，得到体系结构相关的各种设计理由。其中，问题导出、方案发掘和体系结构决策是由架构师等参与者完成，方案合成和理由导出则能够自动实现。

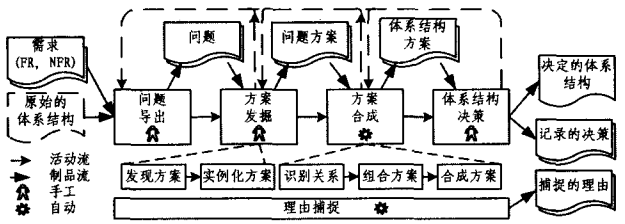

图 2 以决策为中心的设计方法过程

2.3 设计决策的复用

软件体系结构层的复用如果仅仅关注软件体系结构的制品，而设计者不知道软件体系结构背后的设计理由，往往会导致违背原来的设计假设和依赖关系，因此复用过程将存在很大风险。本文把设计决策也作为一种可复用的资产进行管理，建立了设计决策知识库[5]。设计决策知识库包含了问题、方案、决策和理由 4 类设计决策知识。本文为 4 类设计决策知识提供了多种检索方案，支持问题、方案、决策、理由的复用。本文提供了两种复用方式：(1)工具自动提供复用建议；(2)用户手工检索并复用。由于设计决策知识受到设计环境等因素影响，可能复用的设计决策在当前的设计环境下并不完全适用，因此对决策和理由只提供检索以供参考。本文提供了对设计决策修改的接口，从而设计者有足够的空间对设计决策适配。

3 工具的设计与实现

开发的工具 ABC/DD 用于实现体系结构设计决策的建模。ABC/DD 是作为基于体系结构的构件组装工具集 ABC-Tool[6]的一个功能插件，使用 EMF (Eclipse Modeling Framework)[7]开发实现。

图 3 是工具的系统结构图。从设计过程的角度看，用户通过与设计决策编辑器的交互完成设计过程的问题导出、解决方案的挖掘和对体系结构做出决策这 3 个设计活动。另外，工具自动化实现了解决方案的合成和理由的导出。从工具的功能角度看，工具实现的主要功能有：(1)增加体系结构决策；(2)修改体系结构决策；(3)浏览体系结构和设计决策；(4)管理设计决策知识库；(5)检索设计决策知识；(6)复用设计决策知识；(7)追溯设计决策；(8)追踪体系结构模型制品；(9)完整性检查；(10)检查设计决策的各个元素逻辑关系的一致性；(11)自动合成解决方案；(12)自动导出理由。

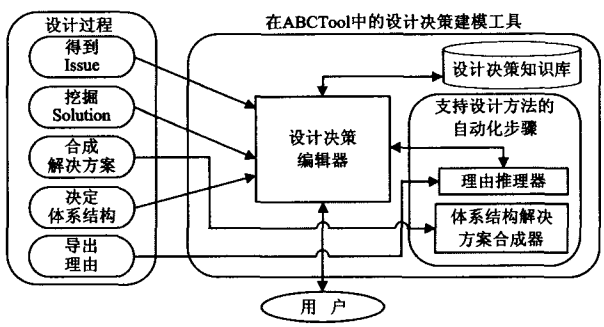

图 3 工具的系统结构图

4 工具应用实例

本节结合一个指挥显示系统(Commanding Display Sys-

tem)说明本文设计决策建模工具的应用。指挥显示系统为指挥者提供丰富的数据信息,并支持数据的监视、分析和指挥。一个指挥显示系统通常有很多的监视器来显示由后端实时传来的数据,同时也需要存储大量的历史数据以便随需查询。为了满足传输现场数据、保证实时性等各方面的需求,迫使设计面临了很多关键的问题,如数据库使用普通的还是实时的、是否需要增加服务器等,这些问题相互交织在一起,很难直接做出抉择。通过以决策为中心的设计过程支持,找到了最终的体系结构解决方案。

以下按照设计指挥显示系统的步骤,介绍工具表示、复用设计决策等功能,展示工具如何支持设计过程中自动化的体系结构方案合成和设计理由捕捉以及在工具易用性和实用性方面的特点。

4.1 体系结构设计

(1)导出问题

涉众(包括用户、架构师等)讨论和确定体系结构关键的问题,基于软件需求、原始体系结构以及其它相关考虑。在后续的各个步骤中,问题也可能被再次提出或者修改,因此整个过程是一个重复迭代过程。在该例中,包含了 7 个需求,如表 1 所列。

表 1 指挥显示系统的需求

需求	描述	类型
R 1	显示现场数据	FR
R 2	显示历史数据	FR
R 3	数据持久性	FR
R 4	实时性	NF
R 5	非超载	NF
R 6	可视化	NF
R 7	可维护性	NF

从以上需求导出 5 个体系结构设计问题,如表 2 所列。

表 2 设计者导出的 5 个 Issue

Issue	描述	关联的需求
Issue 1	现场数据实时获取	R1,R4
Issue 2	历史数据快速获取	R2,R4
Issue 3	数据库装载	R3,R5
Issue 4	可视化	R6
Issue 5	可维护性	R7

例如由 R1“显示现场数据”和 R4“实时性”,导出了第一个问题“现场数据实时获取”。工具为问题提供了描述模板,记录问题的描述、关注的需求和关键词。

(2)发掘解决方案

架构师为问题导出候选的解决方案。这通过两个子步骤实现:首先是发现解决方案,根据可复用的设计知识或者新开发的技术等;其次是将这些非形式描述的解决方案进行实例化,获得用构件和连接子描述的具体设计片断,也就是建立解决方案的配置图。例如,为 Issue 1 发掘了两个解决方案 IS 1.1 和 IS 1.2,表 3 是对它们的简要描述。

表 3 问题方案 IS1.1 和 IS1.2 的描述

问题方案	描述
IS 1.1	描述:数据源将数据直接传给监视器。优点:实现简单;普通的数据库技术。缺点:缺少现场数据的管理。
IS 1.2	描述:数据源把数据传给实时数据库;监视器从实时数据库中读取数据。优点:专门支持实时数据库,现场数据和历史数据可以得到管理。 缺点:需要一个实时数据库,增加开发成本。

工具提供了对发掘的问题方案记录的模板,以及对这些方案建立实例化模型,即配置图的环境。图 4 是对 IS1.1 建模的实例化模型。

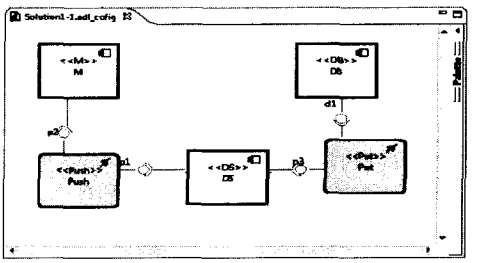

图 4 利用工具对问题方案建立的配置图模型

(3)合并解决方案

当设计者为每个问题发掘了解决方案后,已有的问题方案可以被自动地合成为候选的体系结构方案。

这通过 3 个子步骤实现:首先,问题方案之间的关系被识别,从而确定哪些问题方案可以被组合在一起。例如,IS4.1 和 IS5.2 的数据库构件属性一个是实时的,一个是普通的,它们的构件属性冲突,所以 IS4.1 和 IS5.2 不能合成。其次,通过识别的关系,对组合树进行裁剪,所有可行的问题方案组合被找出;最后,每个可行组合中的问题方案被融合在一起,形成一个候选的体系结构方案。最终,工具生成 8 个可行的候选体系结构解决方案,分别是 AS1 到 AS8,并自动生成各个候选的体系结构解决方案的配置图,图 5 是候选的体系结构解决方案 AS8 的配置图。

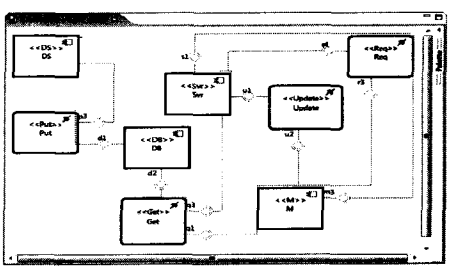

图 5 工具自动合成出的候选体系结构解决方案 AS8

(4)决策体系结构

从合成的候选体系结构解决方案中选择最终的体系结构。为此,架构师需要根据需求的满足程度以及其它的权衡等对这些候选的体系结构解决方案进行评估和比较。最终用户选择采纳 AS8,并给出“可维护性良好”的理由。图 6 是工具提供的对 AS8 进行决策的界面。

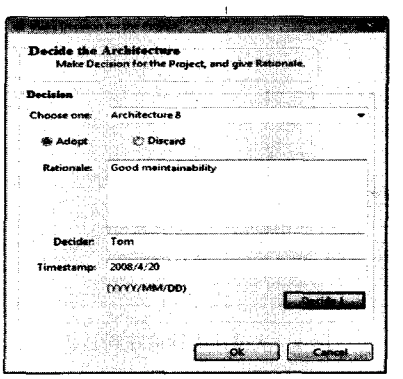

图 6 工具提供的编辑决策信息的界面

(5)捕捉理由

工具自动地从确定的体系结构决策和理由导出各个问题的决策和理由，这是基于问题方案与体系结构方案之间的合成关系实现的。导出的问题决策和理由，以及主要由人决定和给出的体系结构决策和理由，共同构成了全面完整的设计决策和理由信息。由于 AS8 的决策是采纳，意味着 IS1.2，IS2.3，IS3.2，IS4.2，IS5.2 被采纳，而其它的问题方案被放弃。以 IS5.2 为例，工具为它自动导出的理由是它的优点和设计者为 AS8 添加的额外理由"Good Maintainability"。图 7 显示了工具为问题方案 IS5.2 导出的决策和理由。

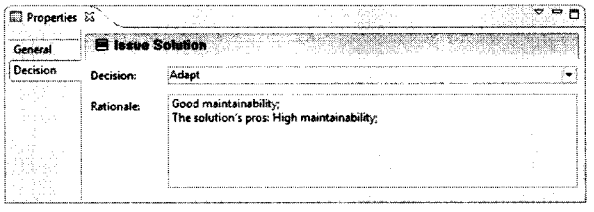

图 7　工具为问题方案 IS5.2 导出的决策理由

4.2　查看追踪性

经过设计的整个过程，设计决策已经和体系结构模型建立了关联，两者存在追踪关系。维护这种追踪关系并显式化追踪关系对于系统的演化和后期设计者、开发者理解系统的设计具有很大的指导和帮助。用户可以利用工具查看追踪关系。

工具提供了在体系结构解决方案的配置图中维护体系结构制品与设计决策的追踪能力，分别有两个方面：

(1)从制品追溯到设计决策。在体系结构的配置图中，当用户选择一个构件或者连接子时，可以追溯它属于哪些问题的实现，了解到这个构件和连接子背后的理由。另外，当用户想删除某个构件或者替换某个构件时，可以知道这个操作会对哪些问题造成影响。如图 8 所示，选择某个构件，其所属的问题加粗显示。

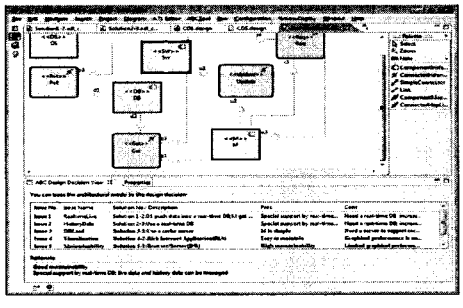

图 8　从制品追踪到设计决策

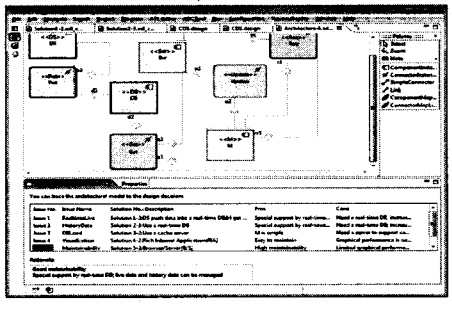

图 9　从设计决策追踪到制品

(2)从设计决策追踪到它对应的制品。这样可以追踪这个问题最终在体系结构模型中如何实现，具有哪些构件和连接子。如图 9 所示，双击某个问题，实现它的构件和连接子高亮实现。用户从而可以知道这个问题的实现对整体的体系结构模型具有哪些影响。

4.3　复用设计决策知识

在设计者为新的项目导出问题时，工具根据该项目的需求，查询以往体系结构设计中的问题，找到那些所关注的需求与当前需求具有相同关键词的问题，作为复用建议提供给用户。

例如根据输入的需求"non-overload"的关键词" DB "检索到编号为 IU0003 的问题，如图 10 所示。点击"复用"按钮，对话框复现这个问题的属性，用户可以在此基础上修改或者直接复用。

在设计者为问题发掘方案时，工具查询以往体系结构设计中的问题，找到那些所解决的问题与当前问题具有相同关键词的方案，作为复用建议提供给用户。

例如，当用户为以上名为" DBLoad "的问题发掘方案时，通过问题的关键词" DB "，可以检索到编号为 IS0001 和 IS0002 的两个方案，最终用户选择 IS0002，复用它的基本属性和配置图，如图 11 所示。

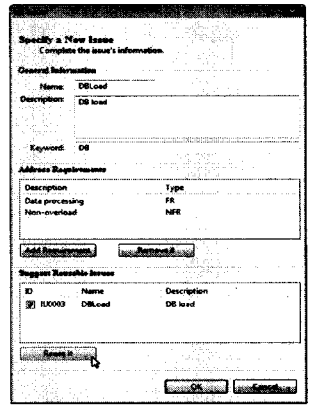

图 10　新增问题时自动提供复用建议

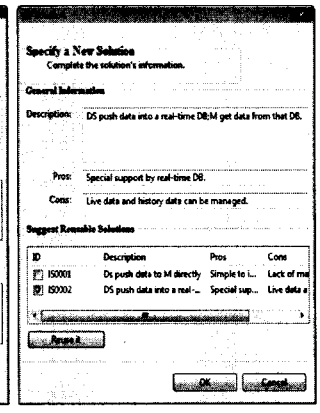

图 11　新增问题方案时自动提供复用建议

另外，工具还提供了检索和复用视图，对问题、方案、决策和理由提供了多种手工检索方式。图 12 是工具为问题方案和体系结构方案提供的手工检索和复用视图。用户通过提供问题描述关键词和问题方案描述中的关键词检索到 IS0004 的方案。

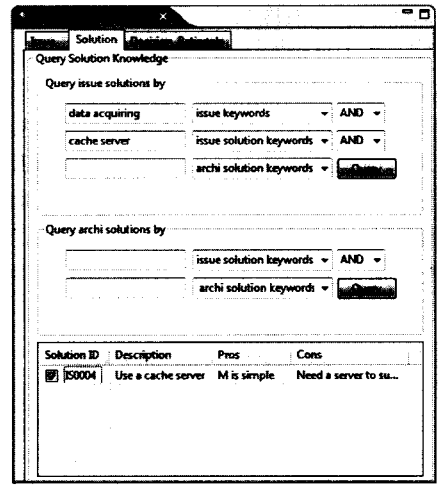

图 12　检索和复用视图

(下转第 173 页)

一点是，软件复用使得专门为软件复用而产生的构件开发和销售成为一种可能。软件模块化和复用的非常重要的问题就是如何对一个提供服务的对象的行为进行描述。这种描述既包括对对象内部接口的描述，还包括对对象外部行为的描述。在面向对象程序中，类和对象通过接口方法对外提供服务，体现对象的外部行为。由于对象是状态、数据和内部及外部行为的封装体，单独依靠对象接口的描述对于正确地使用该对象往往是不足的，还需要描述对象外部接口之间时序关系的对象行为协议。

在软件工程和再工程的活动中，行为协议在对程序进行理解、开发、测试和重用等方面都有十分关键的作用。在现今新的技术层出不穷的情况下，准确地把握系统当前的行为，能够让开发人员和维护人员尽快地进行技术上的调整，而能保持原有的行为效果不变。这对于节省开销，保持程序的稳定性都有重大的意义。提出了一种静态的从遗产系统中恢复出程序接口行为协议的方法。最终的接口约束以抽象状态图的形式表示出来。通过观察，发现对象状态的变化是由于对象属性值发生了改变，而属性值的改变引起状态改变的原因是使用属性的方法受到了影响，因此，需通过考察对象中属性变量的共享使用来推测方法之间隐含的依赖关系，恢复出方法调用的行为序列。这是一个自动的静态分析过程，而且，恢复出的状态图在保留协议的约束行为的同时，减少了状态图的复杂度和冗余性。

与现有的其他一些研究方法相比，方法直接对源代码进行解析，在解析过程中不涉及到对源代码的修改或其它辅助代码的生成等。比如动态方法就需要对程序的源代码进行插装工作，并且需要生成一定数量的动态测试代码。静态分析方法避免了这种对于源程序的依赖性，使得能够更普遍地被采用，并且分析结果相对比较稳定。而对比传统的静态分析的状态生成方法，我们将程序的状态抽象成为对象的行为能力，这种抽象更符合人们直观地理解对象状态，同时也能减少冗余状态的产生和状态图过于复杂等问题的出现。通过实验证明，根据我们的方法开发的原型工具可以提供完全自动化的对象行为协议的恢复工作。

结束语 对象行为协议对于理解对象接口、正确实现模块集成以及类代码的复用都有着重要的意义。本文在前期所提出的基于静态分析的对象行为协议抽取方法基础上，进一步介绍了自动抽取工具的实现技术，并进一步对行为协议描述的验证进行了探讨。在今后的研究工作中，将着力于改善方法中对于接口信息的定义形式，使其能提供更强的描述功能。此外，还将尝试引入一定的动态分析工作，这有助于为我们方法提供对象在动态运行时的变量信息，根据这些信息对对象的行为方式进行更深入的分析。

参 考 文 献

[1] Yang Jinlin, Evans D. Dynamically Inferring Temporal Properties[C]// Proc. the ACM-SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. 2004:23-28

[2] Yuan Hai, Xie Tao. Automatic Extraction of Abstract-object-state Machines Based on Branch Coverage[C]// Proceedings of the 1st International Workshop on Reverse Engineering To Requirements at WCRE 2005 (RETR 2005). November 2005:5-11

[3] 黄洲，彭鑫，赵文耘. 基于依赖性分析的对象行为协议逆向恢复[J]. 计算机科学，2008，35(8)：265-268，276

[4] Tang Mei-huei, Wang Wen-li, Chen Mei-hwa. A UML Approach for Software Change Modeling. cs. albany. edu

[5] http://compilers. cs. ucla. edu/jtb/jtb-2003

[6] https://javacc. dev. java. net

[7] Mohamed G. Gouda Closed Covers: to Verify Progress for Communicating Finite State Machines Technical Report[R]. CS-TR-82-191 Year of Publication: 1982

[8] http://www. graphviz. org

(上接第 164 页)

结束语 本文介绍了一个软件设计决策的建模工具。该工具帮助架构师对体系结构设计中的问题、方案、决策、理由进行显式化的建模，完成从需求到体系结构的设计过程，并在此过程中实现了自动化的候选体系结构方案的合成和部分设计理由的捕捉。该工具提供了一个综合的设计环境，包括体系结构设计过程、设计决策的可视化、设计模型的追踪性、设计决策知识的复用等功能，从而更好地帮助架构师提高体系结构设计的效率和质量，使设计结果能够更好地支持体系结构的理解和演化。

下一步将对问题之间和决策之间的相互关系进行发掘和建模；在工具方面，将继续对易用性进行加强。

参 考 文 献

[1] Shaw M, Clements P. The golden age of software architecture [J]. IEEE Software, 2006, 23(2): 31-39

[2] Shaw M, Clements P. The golden age of software architecture [J]. IEEE Software, 2006, 23(2): 31-39

[3] van Gurp J, Bosch J. Design Erosion: Problems & Causes[J]. Journal of Systems and Software, Elsevier, 2002: 61(2): 105-119

[4] Jansen A, van der Ven J, Avgeriou P, et al. Tool support for Architecture Decisions[C]// Proc. 6th IEEE/IEIP Working Conference on Software Architecture (WICSA'07). 2007

[5] Cui X, Sun Y, Mei H. Towards Automated Solution Synthesis and Rationale Capture in Decision-Centric Architecture Design [C]// Proc. 7th IEEE/IFIP Working Conference on Software Architecture (WICSA'08). Feb. 2008

[6] Cui X, Sun Y, Xiao S, et al. A Decision-Centric Architecture Design Method Facilitating the Contextually Capture and Reuse of Design Knowledge[C]// Proc. 12th International Conference on Software Engineering and Knowledge Engineering (SEKE'08). July 2008

[7] Mei H, Huang G. ABCTool: A Tool for Architecture Centric Engineering of Component based Systems[C]// Tool Demonstration. Track of ICSE. 2008

[8] Eclipse Community. Eclipse Modeling Framework[OL]. http://www. eclipse. org/emf

[9] 梅宏，陈锋，冯耀东，等. ABC：基于体系结构，面向构件的软件开发方法[J]. 软件学报，14(4)：721-732