计算机科学2000Vol. 27№. 2

基于多视点的需求工程方法*)

Multiple-Viewpoints Based Requirements Engineering Approaches

何炎祥 黄 浩 石 莉 张 戈 李 超

(武汉大学计算机科学系 软件工程国家重点实验室 武汉430072)

Abstract Requirements specification is one of the most important problems in software development. The main objective of the requirements engineering is to provide a model of what is needed in a clear, precise, unambiguous and consistent statement of the system to be specified. Viewpoints are seen as a means for separating concerns in software development in accordance with a variety of criteria, which has a significant role in achieving a successful system. The paper is a survey of the current multiple viewpoints based requirements approaches; a simple example of a distributed multimedia conferencing system is used to demonstrate them.

Keywords Requirement engineering, Viewpoint, Distributed multimedia conferencing system

1 引言

视点其实是集体工作的一个普遍特性:不同的人对于要协作完成的工作,很自然地,会从他们所处的角度有自己的不同于他人的看法。视点是指主体(agent)和主体的视图(view)。软件开发过程中,比如在建立或描述一个复杂的系统模型时,通常需要许多主体的参与。同时,也要求主体对系统相关信息的收集尽可能完全,特别是那些关系到系统成败的关键信息。视点技术在定义好的系统结构的基础上有机地分配给开发人员以不同的职责,从而使得不同的参与者都能够从一个适当的角度观察这个系统,即论域,得到各自的透视(perspective)或视图,例如安全视图、体系结构视图、性能视图等。虽然每个透视或视图是对整个系统部分且不完全的描述,但是因为视点技术允许从不同的视点收集系统信息,丢失系统关键信息的可能性已经大大地减小,并且它把一个复杂、难于管理的任务适当地分解成为若干简单、易于管理的子任务。

近年来,国际上对视点在需求工程中的应用作了许多理论和实际的工作。需求工程研究的是具有复杂结构的大型系统以及若干系统的结构和行为之间的内在限制。因此,需要多个视点来进行分解、分析和控制工作。事实上,在软件开发整个过程中都会遇到视点这个问题,现有的许多设计方法,例如数据流模型或实体-关系模型,都隐含地使用了视点。

视点先是在SADT(Structured Analysis and Design Techniques)[1~3]中被提到,然而第一次正式提出视点是在CORE(Controlled Requirements Expression)[4]中。随后,又出现了许多适用于软件开发不同阶段的视点模型,其中有:Finkelstein[5~7]提出了一个基于视点的软件开发方法,以此作为分布式软件开发的一个框架;Leite[8]设计了一个视点模型,用来识别和解决在需求分析时的完全性和正确性问题;Kotonya 和 Sommerville[9]主要研究的是需求工程的视点模型。下面将对这些方法依次做简要的介绍,并以一个分布式多媒体会议系统作为要进行需求分析的系统。

2 基于多视点的需求工程方法

2.1 SADT(结构化分析和设计技术)

SADT 方法基于如下假设:系统被看成一个人们所关心的活动的集合;只要能够将其分解成易于掌握的子部分,人们就能够处理任何复杂的系统。SADT 的目的就是找到这些子部分,对它们和它们之间的关系模型化。

SADT 主要由两部分组成:(1)用于结构化分析(SA,Structured Analysis)的 box-arrow 图表。见图1。其中矩形框(SA box)代表系统最抽象的活动,左右上下四个箭头分别代表输入,输出,控制和机制。机制在控制的作用下将输入转变为输出。例如,在代数式 A+

*)国家教育部重点项目资助的课题。

B＊X＝Y 中，SA 框是整个公式，输入是参数 X，输出是结果 Y，控制是变量 A 和 B，机制是运算符＋和＊。(2)设计技术(DT，Design Techniques)，为了高效地使用图表语言，就必须学习和掌握设计技术，它是指导思维和行动的原则。

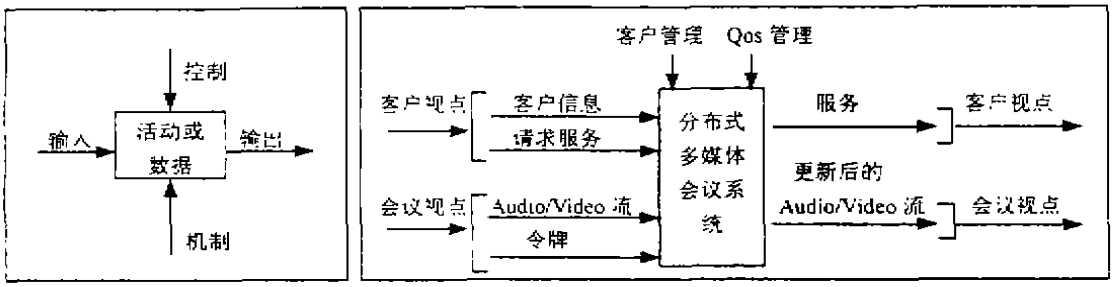

图1　结构化分析图表

图2　一个分布式多媒体会议系统的模型

SADT 并没有定义出"视点"这个概念；视点只是需求分析者的主观活动，通过对数据的输入端和输出端进行分析，从中得出视点。实际上，被看成是数据流的源或目的地的视点，是 SADT 设计技术的一个很自然的扩展。图2给出了一个最抽象的多媒体会议系统的模型，其中有两个视点："客户视点"和"会议视点"。"客户视点"描述了和与会者有关的输入和输出数据，包括与会者身份信息，请求和服务，而"会议视点"描述了会议信息的改变。其实，控制："客户管理"和"Qos 管理"，与输入一样，会对输出数据产生影响，但 SADT 没有为控制与视点的联系提供应有的框架。

2.2　CORE(受控的需求表达)

CORE 方法是由 System Designers 在七十年代后期为英国航天局开发的。CORE 方法诞生之后，取得了很广泛的应用，影响很大。

CORE 是一个基于视点的功能化需求分析方法。需求可以分为两大类：功能化需求和非功能化需求。功能化需求的目的是把握部件(可以是一个系统、一个软件包或者一个硬件设备)和部件所处的环境之间相互交互的特性；非功能化需求则对那些可能的解决方案给出一定的限制条件[10]。CORE 方法将视点分为两层：第一层由与系统交互的所有实体组成，其中，CORE 提供了区分功能化视点和非功能化视点的准则；第二层主要研究的是定义视点(defining viewpoint)和边界视点(bounding viewpoint)，其中，定义视点是与系统直接交互的系统的子系统，而边界视点是那些与系统间接交互的实体。

CORE 方法有七个步骤，这里，本文只简单介绍与需求说明相关的主要内容：

1)识别视点。CORE 没有为如何识别视点规定固定的规则，而是建议，系统分析人员和用户一道使用"脑风暴"方法得到所有可能的视点。然后，将得到的视点分为功能化视点和非功能化视点(图3)；再把功能化视点又分为定义视点和边界视点(图4)。

2)结构化视点。在这个阶段，目标系统被分解为由功能化的子系统组成的层次结构，其中，每个子系统都是一个视点。这样做的好处是，把分析任务分成若干不同抽象层次上的需求说明，便于对分析进行控制。对于如何选择和结构化功能化视点，CORE 提出了一些建议，如功能化视点应是信息的改变者，在每个视点层上的功能化视点应不超过五个等。

3)收集表格。表格的收集就是收集视点的相关信息，并解决视点间信息遗漏和冲突问题。对于每个视点，需要考虑五个方面：它执行的动作；输入的数据；数据源；输出的数据；数据目的地。

4)结构化数据。

5)对单个视点建模。

6)对复合视点建模。

7)分析约束条件。

CORE，和 SADT 一样，都不支持多视点的需求说明。但是，CORE 支持间接视点。CORE 中的间接视点有点象负责发送输入数据到进程，或从进程接受输出数据的外部实体。只不过，CORE 把主要注意力放在定义视点上，作为进一步分解的基础，定义视点负责将输入转化为输出。

2.3　VOSD

VOSD(面向视点的软件开发，Viewpoints-Oriented Software Development)中提出，在开发大型软件系统中，需要许多不同软件开发方向和不同应用领域的专家参与，而且，他们的职责和注意力有可能随着软件的开发发生变化。因此，在 VOSD 中，使用视点来管理软件开发过程中各个阶段每个参与者所肩负的职责。

识别视点的依据是参与者的职责，他们感兴趣的领域和领域相关的知识。VOSD中，每个视点由五个槽

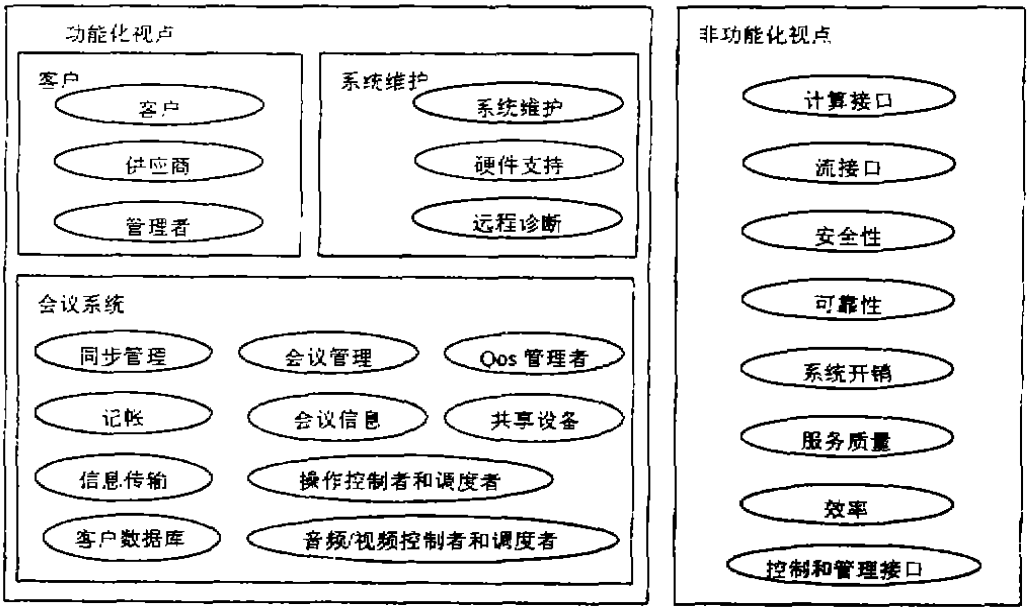

图3 功能化视点和非功能化视点

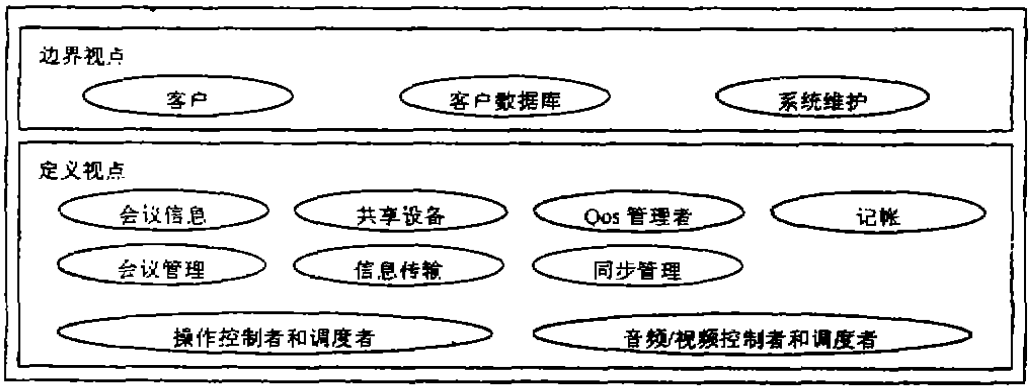

图4 定义视点和边界视点

(slot)组成：

1)风格，即用以描述视点的表达模式；

2)论域，即视点所关注的领域，它是视点的部分标识符；

3)工作计划，由生成说明的动作的集合，和控制动作执行的处理模型组成。视点的所有者负责执行视点工作计划中的处理模型。视点的所有者通常是开发人员，但也可以不是开发人员，可以是某种智能工具或专家系统；

4)需求说明，即按风格槽中描述的表达风格和工作计划槽中描述的开发策略来描述论域；

5)工作记录，即视点所执行动作的历史记录，用以保存视点说明的开发状态和动作记录。

VOSD借用了CORE的思想，整个需求说明分成了若干开发阶段，每个阶段使用一种视点模板。视点模板是指仅含有表达风格和工作计划的视点。通过初始化一个视点模板可以生成一个对特定论域进行说明的视点，同时，由视点的工作计划生成视点的说明。最后，视点的配置，即视点的集合，一起构成系统需求说明。图5是由CORE方法中表格收集阶段的视点模板生成

的一个视点，它描述了多媒体会议系统中客户的活动。

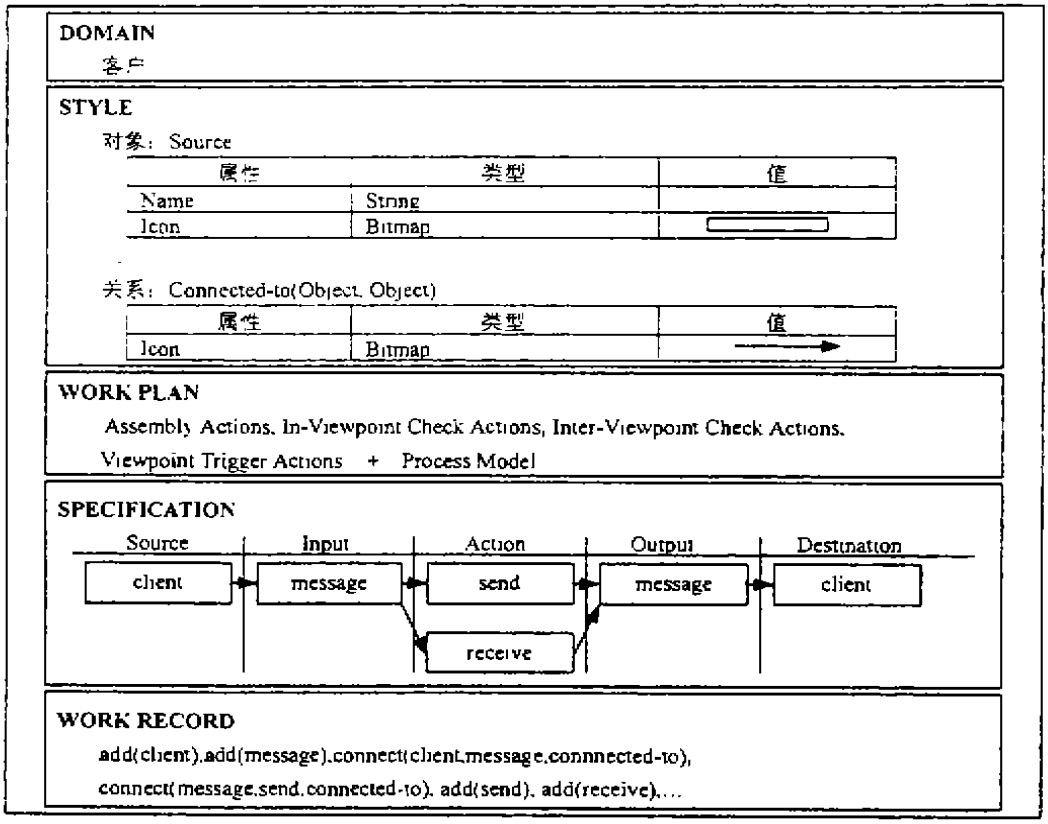

图5 表格收集阶段的视点

2.4 视点分析

Leite 提出的视点分析方法，旨在对于需求分析早期过程中的完整性和正确性问题加以识别和分类。他认为从多个角度收集信息能够更好地理解需求，从而可以更好地保证完整性和正确性。为此，Leite 给出了如下的概念。

视点，即需求分析者在分析整个论域时所处的位置。

透视，即需求分析者观察得到的一组事实，分为数据透视、分析者透视和操作透视。

视图，即通过一个视图构造过程而得到透视的集成。注意，这里 Leite 将透视和视图看成是两个不同的概念。

层次，即论域中概念的 is-a 和 parts-of 层次。

此外，Leite 还提出了一个视点语言(VWPL)，用以描述视点。Leite 提出的视点方法中，至少要有两个分析者，即至少要有两个视点。分析者用 VWPL 表达他们对整个系统的理解，然后使用数据透视、分析者透视和操作透视，以及 is-a 和 parts-of 层次的概念形成和改进各自的透视，最后消除透视之间内在的冲突，将它们集成为一个视图。

2.5 VORD

Kotonya 和 Sommerville 在其它学者研究工作的基础上，开发出面向视点的需求定义 VORD(viewpoints-oriented requirements definition)方法。VORD 覆盖了需求工程的全过程，从发现需求直到系统建模。视点，即需求源，由终端用户、风险承担者、相邻系统和系统环境中可能会对系统产生影响的其它实体组成；视点和系统之间的关系建立在视点对系统的要求和视点与系统的交互上。

VORD 分为三个步骤：(1)视点的识别和结构化；(2)书写视点文档；(3)视点需求分析和说明。在第一步中，VORD 将“系统特权者”构造成一个视点类的树图，以此作为识别视点的起点。所谓“系统特权者”，是指对应用领域有兴趣或专家知识的人或文档，包括系统的终端用户，系统的中介者，系统工程师和现存系统的文档。图6是视点类的一个示例。当然，这个类的层次并不是通用的，每个系统都应该在自己的需求上建立

属于自己的视点类的层次。

VORD 将视点的识别分为五个阶段：

(1)去掉视点类层次图中与系统无关的视点类；

(2)系统的风险承担者，即那些将受到系统影响的人，如果他们没有被包括在视点类层次中，那么在层次中填加这些类；

(3)通过系统结构模型来识别于系统的视点；

(4)通过区分经常使用系统的用户，偶尔使用系统的用户和间接使用系统的用户，得到系统潜在的视点；

(5)对于那些间接视点类，考虑与它们相关的分析人员的视点。

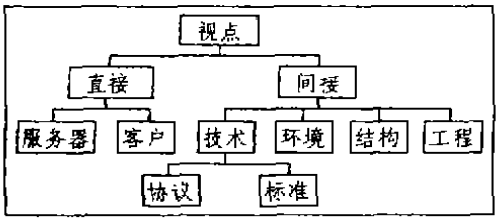

图6 一个视点类的例子

结束语 本文中我们集中介绍了一些比较典型的基于多视点的需求工程方法，同时，我们用一个简单的分布式多媒体会议系统的例子，说明了它们是如何进行视点的识别，视点的分类，视点的说明，视点的集成的。这个领域还有许多问题，如视点间或视点内不一致性和冲突的管理等，都亟待更好地解决，目前已有一些学者开始了这方面的研究工作。

参 考 文 献

1 Ross D, Schoman K E. Structured analysis for requirements definition. IEEE Trans, 1977,3(1):6～15

2 Ross D T. Structured analysis: a language for communicating ideas. IEEE Trans., 1977,SE-3(1):16～34

3 Ross D T. Applications and extensions of SADT. Computer, 1985,18(4):25～34

4 Mullery G P. CORE—— a method for controlled requirement specifications. In: 4th IEEE Computer Socty Int Conf. On Software Engineering. Munich, Germany, 1979. 126～135

5 Finkelstein A, et al. Viewpoint: a framework for integrating multiple perspectives in system development. Int. J. Softw Eng. Knowl. Eng., 1992,2(1):31～57

6 Finkelstein A, Fuks H. Multi-party specification. In 5th Int Workshop on Software Specification and Design, IEEE Computer Society, 1989. 185～195

7 Nuseibeh B, et al. A framework for expressing the relationships between multiple views in requirements specification. IEEE trans. on Softw. Eng., 1994,20(10):760～773

8 Leite J C S P. Viewpoint analysis: a case study. In Proc. 5th Int. Workshop on Software Specification and Design (IWSSD-5). IEEE Computer Society Press, 1989. 111～119

9 Kotonya G, Sommerville I. Requirements engineering with viewpoints. Softw. Eng. J. 1996,11(1):5～18

10 Roman G-C. A taxonomy of current issues in requirements engineering. Computer, 1985(April):14～21

(上接第17页)

到抽象的程序；最后，利用精化演算的求精规则由抽象程序得到可执行程序代码。

目前从形式化规约到程序代码的求精过程主要由人工实现，自动化程度较低。如何在求精过程中尽可能多地应用机器辅助技术是形式化方法能否成功的关键问题之一。例如：利用机器辅助技术进行数据精化的正确性证明，机器辅助实现从 Z 到精化演算表示体系的转换，机器辅助实现操作精化。但在实际应用中还有大量工作要做，如对各求精步骤具体、实用策略的深入研究、完备的谓词转换定理系统的建立及其选用策略的研究等。

参 考 文 献

1 袁晓东. Z 的面向对象扩充和程序转换系统的研究：[南京大学博士毕业论文]. 1997. 12

2 Spivey J M. Understanding Z: a specification language and its formal semantics. Number 3 in Cambridge tracts in theoretical computer science. Cambridge University Press, Cambridge, 1988

3 Morgan C C. Programming from Specifications. Prentice-Hall International series in computer science/C. A. R Hore, series editor. Prentice-Hall International, Englewood Cliffs, N. J.; London, 1990

4 Hayes I J. Specification case studies. Prentice-Hall International series in computer science/C. A. R. Hore, series editor. Prentice-Hall International, Englewood Cliffs, N. J.; London, 1980

5 King S. A refinement calculus case study: [Technical Report PRG-TR-7-90]. Programming Research Group, 1990