

40-44/50 协作设计系统交互环境中的解释*)

Explanation in the Interactive Environment of Collaboration Design System

陈建国 潘云鹤

(浙江大学人工智能研究所 杭州 310027)

Abstract In the collaboration design systems, a designer and a knowledge system may cooperative-ly solve a design problem. Explanation is an important issue in building an effectively interactive design environment of collaboration systems. This paper considers the two related problems of explaining the system's reasoning and the design generated by the system. In particular, the content of the explanations of design reasoning and design solutions is analyzed.

Keywords Interactive design environment, Explanation, Modeling methods, AI

一、引言

在协作设计系统中,内部表示和推理机制是重要的。然而,外部表达以及知识与设计者的交流同样重要。当研究设计问题的一个子系统时,对该子系统的解释有助于问题的其它子系统的构造,同样,对设计推理和设计方案的解释有助于在未来设计中重新使用部分推理或方案。

当设计者和知识系统合作解决一个问题时,系统必须精确地向设计者解释它正在做什么,如何做及为什么做。另外,系统必须能证明针对给定问题所提出的设计方法的正确性。没有这些功能,用户对设计没有把握,也很难发现设计中潜在的缺陷。因此,构造实用的交互设计环境需要设计解释的理论以及接口的建立。

这样,问题就变成了知识系统如何解释其推理和提出的设计方案。推理和方案的解释是当今 AI 研究的主题。所谓“解释”是指系统自动说明,而不是外部数据的展开说明,解释包括方案和在生成方案时所用知识的证明^[1]。交互系统中的解释包括几个相关但又有所不同的方面,例如解释的内容、生成、表达等。本文主要研究解释内容。

设计方案的解释内容取决于设计的阶段和领域,例如,初始设计结果的解释不同于构造设计结果的解释。前者是关于设计的功能和结构,而后者往往

涉及几何信息。同样,一个图案设计的证明内容不同于建筑物,因为两者的功能和结构之间的关系根本不同。

本文首先描述了任务-方法-知识模型和功能-行为-结构模型,然后说明这两种模型分别在设计推理和设计方案中的解释作用,最后提出了需在方案的环境中解释推理,方案的解释应立足于引出方案的推理环境。

二、任务-方法-知识模型

知识系统中的 AI 研究已发展了几个关于问题求解、知识获取、解释的面向任务的理论。虽然不同的理论在许多细节上各不相同,但它们都用独立于领域的目标类型(任务)和任务特定的推理模式(方法)确定问题求解的内容和构成。为更精确地描述知识的内容、形式、构成、以及在问题求解过程中的功能作用,本文运用分治技术提出了任务-方法-知识模型(TMK 模型),并将它用于解释设计问题的推理。

2.1 分解与合成

分治技术是一种分解-合成技术,即对于复杂的不能直接求解的问题,将该问题分割成几个子问题,分别加以求解,然后合成子问题的解,以获取整个问题的解。若分割出的子问题仍不能求解,则继续分解子问题,一直到可以直接求解为止。

*) 国家自然科学基金资助课题。陈建国 博士生;潘云鹤 中国工程院院士、浙江大学校长,博士生导师。

根据任务、用于完成任务的方法、方法产生的子任务以及方法使用的知识, TMK 模型提供了设计推理的功能和策略的解释。因为子任务由适用于推理的方法产生, 故 TMK 模型使用产生子任务的方法的同时也证明了推理所关注的特定任务。同样, 因为方法为任务服务, 且是由合适的知识提供, 因此 TMK 模型用当前处理的任务和支持方法的知识证明了推理系统对特定方法的使用。

2.2 TMK 模型的构成

设计推理的 TMK 模型有三个主要元素^[2]。第一个元素是任务。它由接收的输入和产生的输出的信息类型表示, 任务输入是所需设计的功能规范, 其目标是输出满足所需功能的结构规范。TMK 模型的第二个元素是方法。方法的表示有: (1) 它使用的知识类型; (2) 它建立的子任务 (如果子任务存在); (3) 它对子任务处理所施加的控制。模型的第三个元素是知识。特殊类型的领域知识由内容表达方式和它的结构组成。

TMK 模型中任务-方法的分解是递归的: 用于任务的方法生成子任务, 对每个子任务重复同样的任务-方法分解。对一个给定的任务, 这个递归分解一直进行, 直至到达“元素”层。在这一层中, 可用领域知识直接解决子任务, 即领域知识直接将子任务

的输入映射至所需的输出, 而无需任何其它方法。用术语“过程”表示这种方法。一个过程不再产生子任务。递归任务分解导致了一个任务-方法-子任务树。

2.3 TMK 模型生成步骤

生成 TMK 模型的步骤为:

- 任务识别: 首先, 根据输入输出的通用信息类型确定任务。
- 知识识别: 根据领域中适用的知识分析领域。
- 方法识别: 识别由不同种类知识提供的各种方法。这一步包括每个方法建立的子任务的识别。
- 方法选择: 因为可能不只一个方法适用, 因此要确定选择特殊方法的准则。这些准则可包含所需方案的性质和方法的计算特性等因素。
- 递归任务-领域分析。最后, 对所选方法建立的每个子任务重复执行上述步骤。

浙江大学计算机系人工智能研究所研制开发的“美术图案创作智能 CAD 系统”就是一种运用 TMK 模型的图案创作系统, 如图 1 所示。根据原始设计需求信息, 确定图案创作的任务 (根图案)。然后, 确定和选择构成最终图案的方法, 例如画面的布局推理、元素搭配推理、元素变形、元素选取等。根据所选方法, 产生相应的子任务, 最后递归分解直至元素层。

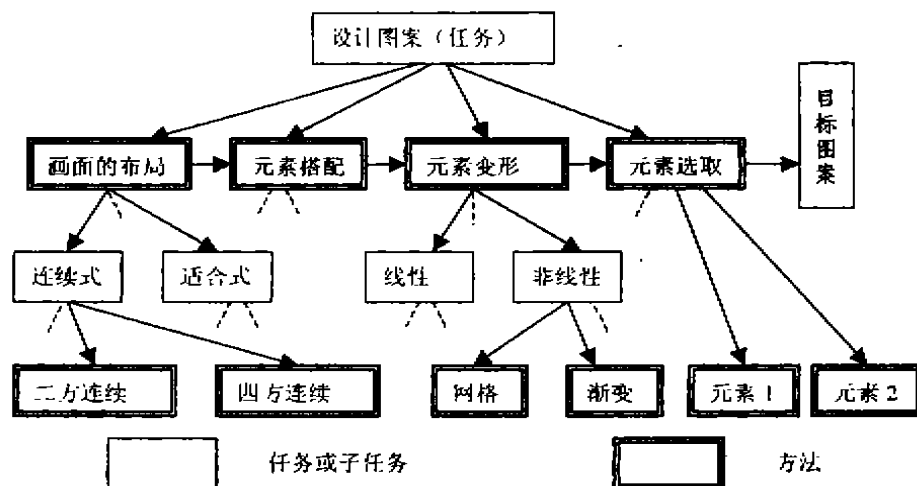


图 1 美术图案创作智能 CAD 系统的 TMK 模型

三、功能-行为-结构模型

功能-行为-结构模型 (FBS 模型) 根据设计对象的功能、行为和结构之间的关系, 分层次地描述设计对象, 如图 2 所示。这里, 使用功能-行为-结构模型

(FBS 模型) 解释设计方案。FBS 模型提供了对象如何工作和结构如何获得功能的功能和因果解释。解释明确了设计中每个结构元素所起的功能和因果作用。因为 FBS 解释了结构如何达到所需功能, 因此它也证明了设计。

3.1 功能

功能是一种依赖于设计者意图的直觉概念。通常,功能是指输入和输出之间的能量、材料、信息之间的关系。根据功能与行为的关系,功能又可定义为:为了利用行为,设计者通过对行为的认识,通过抽象而得到的行为描述^[1]。因此,功能确定了输入的行为状态,设计产生作为输出的行为状态,以及取得这种转换的内部因果行为序列。功能的表达可根据功能变量、功能与对应的行为之间的关系,用一个表示对象目标的标号和一个行为集来表示。由于功能作为行为的索引,一个功能可由多个行为完成,而单个行为对应多个功能,因此功能与行为的关系是一种多对多的关系,表示为 $F \leftrightarrow B$ 。

3.2 行为

行为是一种推断结构含义的方法,由行为类型、行为变量和定性因果关系表示,它将子结构的功能构成整个设计的功能。行为类型有两种:空间类型和时间类型。空间行为在 2D 或 3D 空间中描述有序物体,表示为行为状态;时间行为服从时间约束,表示为行为状态间的转换序列。状态转换上标注因果、结构和功能的环境。在此环境下,状态转换才能发生,状态变量才能转换。因果环境指明了在前后状态中变量间的因果关系。结构环境指定了组件、物质、对象不同空间位置之间的不同的结构关系。功能环境指出对象中哪个组件的哪个功能对转换起作用。

行为变量有结构性变量和外部变量。结构性行为是从结构本身直接派生得到的,而外部行为是外部物体作用于结构时所表现出来的特殊行为。

行为与结构的关系也是一种多对多的关系,即 $B \leftrightarrow S$ 。在链接结构和功能方面,行为起着重要的作用。

3.3 结构

结构表示了构成设计的组件、组件的属性、组件之间的关系以及在组件之间所传递的物质。组件和物质可以在结构和行为上相互作用。

每个组件和组件间的关系对设计的功能直接或间接地起作用。结构的变量可以是组件、属性、关系、操作或过程。结构分成两个部分:数据结构和过程结构。其中,数据结构由组件、属性、关系构成;过程结构由操作和过程组成。

任何一个设计结构都是由组件组成。系统中不能再细分的组件称为原子组件。有些组件组合在一起形成一个子结构或结构组件。原子组件和结构组件是被动的,设计者可以查询、访问或操作它们。在

设计过程中最重要的步骤是选择组件。

一个组件有许多特征或属性。属性值可以是连续的,也可以是离散的。一些设计在设计结构中结合了操作和过程。操作是一种动态组件,它从被动组件获取信息,采取适当的控制行为调节结构的行为。过程是由一系列操作和子过程组成。

使用 FBS 模型解释对象与 Simon(1981)^[2] 的关于实体的功能解释的观点相符。Simon 认为实体的解释是关于实体的目的并为目的所引用,因此,我们认为 FBS 模型具有设计者理解的具有恰当抽象程度的设计解释。

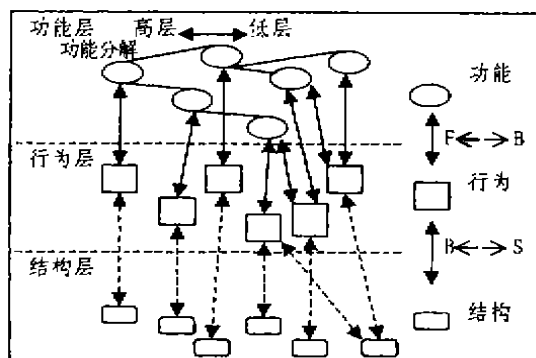


图2 功能-行为-结构模型

四、模型与交互设计

在 AI 和设计的交叉处,有三个重要的主题:设计方法和过程模型、设计知识和对象模型、交互设计环境。

4.1 设计方法和过程模型

在设计中,AI 研究的一个主要目标是为设计提出计算方法和过程模型,这引起了几个计算方法的发展,例如启发性搜索、启发性联想、计划说明和扩展等。近期在基于事例设计的研究中导致了设计的多策略处理模型的发展,紧密相关的研究方向也涉及到确定计算方法和过程模型的语言。

任务级或知识级的说明更明确地区分了知识系统中基于知识的推理与实现。在 80 年代中期,Chandrasekaran 提出了通用任务语言用于对基于知识的问题求解的分析和建模,这种语言允许更清楚的解释。80 年代后期,他将通用任务与任务结构相连接,描述了用于设计的高层任务结构^[3]。Goel 等人亦为基于事例设计描述了一个精确处理的任務结构。

TMK 模型是基于通用任务的任務结构的概

括,为解释基于知识的推理提供了适当的抽象。与基于通用任务方法解释的早期模型相比,TMK 模型更明确地说明了特定知识类型所起的作用。因此,当任务结构以任务-方法交互形式适用于推理的解释时,TMK 模型对解释方法-知识交互也有用。特别地,它们允许对不同种类知识、任务的知识类型、方法的构成和索引进行解释。

4.2 设计知识和对象模型

对象的解释不仅一直是 AI 和设计研究的主要话题,而且也是认知工程的主题。80 年代早期,Chandrasekaran 等人提出了功能表达(FR)方法^[6],FR 方法不仅明确表达了功能,而且功能可用于查询对完成功能有用的因果行为。这个因果行为反过来又可查询对象子结构的功能。因为子结构的功能与导致它的因果行为相关,这样便产生了对象模型的层次结构,即功能→行为→功能…。在 80 年代中期,Bylander 提出了部分基于 Hayes 的组件-物质存在论的原子行为分类,他也描述了一种将原子行为组成更复杂行为的方法。FBS 模型从 Chandrasekaran 功能表达法和 Bylander 的行为存在论进化而来。它使用 FR 的构造方法,即功能充当因果行为的索引,因果行为索引对象子结构的功能。FBS 模型中功能、行为和结构的规范是基于 Bylander 的定义良好的行为存在论。

在认知工程中,Rasmussen 提出了表示对象知识的分层结构,他的对象模型还指明了层次中各层的结构、行为和功能。

在设计研究中,Chandrasekaran、Gero、Goel、Maher、Umeda 等人都纷纷提出了相应的 FBS 模型,尽管在这些模型中描述方案的细节不尽相同,但是它们均将设计过程需要的变量分组为三个组:功能组、行为组和结构组。设计中,将它们连接起来,并从其中一个转换成另一个,这表明了功能、行为、结

构三者的紧密相关性。

4.3 交互设计环境

交互设计环境中的一个核心问题是设计者和知识系统如何分担设计职责。AI 对交互设计环境的研究覆盖了广泛的设计者和系统的职责分担。

一个极端是,系统作为知识源,而将几乎所有的推理留给了设计者。用于设计的知识库通常包含设计组件和材料等知识。人完成几乎所有推理将受到人的基本的认知局限的限制,例如人的短时记忆能力、遗忘、对长期记忆的回忆较慢等^[7]。

另一个极端是,自动执行几乎所有推理的自动知识系统。设计者与这些系统的交互局限于设计问题的形成,将问题提交给系统,接收系统生成的方案。这种系统的局限性包括难于获得足够的完整的领域知识、专家决策制定过程的不透明性、规范问题等。在这些问题中规范问题最难解决,因为在开始设计时,规范可能不完整(即设计的某些特征没有指明),另外某些特征值不知道。

要解决上述两种极端系统的局限问题,必须建立交互的设计环境,允许设计知识与设计者进行交流,允许设计者和系统协作解决设计问题。因此,在这两个极端之间存在大量的设计者和系统分担职责的设计环境。

交互设计环境应对系统的设计推理和设计方案进行解释。通常,设计系统有两个部分:完成设计任务的子系统和用户解释接口子系统。解释接口不仅要解释设计推理和设计方案,而且要允许用户发展和探索设计,并反映设计的推理。

以基于事例的设计系统为例,通过接口子系统,用户可访问所有的设计知识,包括设计事例及其 FBS 模型。利用 FBS 模型可图示说明和解释设计方案的功能,使用 TMK 模型可图示说明系统生成新设计的过程,如图 3 所示。推理踪迹存贮在事例之

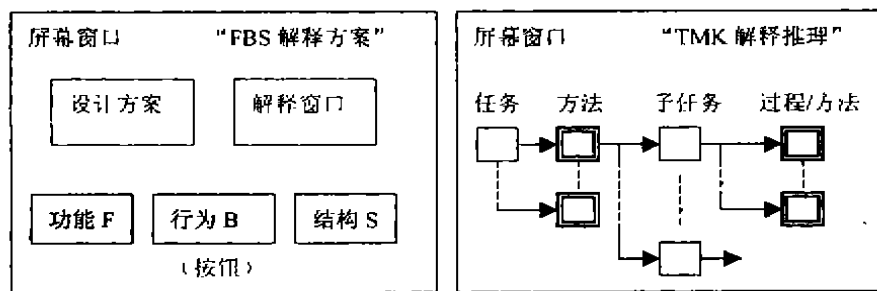


图 3 FBS 模型解释设计方案以及 TMK 模型解释设计推理

中, TMK 模型使用户能检查任务、方法、知识和推理状态, 这使用户能对设计推理作出反映, 例如用户可检查 TMK 推理踪迹, 探测其中的缺陷。用户也可要求系统证实一些推理选择。

近期对设计知识库的研究集中在使设计者可以访问设计事例库, 从而产生了许多相应的基于事例推理的设计 (CBD) 系统。主要的 CBD 系统有 CADSYN, CADET 等。但在这些设计系统中, 没有一个提供了解释接口, 这些系统允许设计者访问不同领域中的设计事例库, 事例中标注提供了设计的解释, 但它们不提供事例推理本身的恰当解释。

Fischer 等人的 JANUS^[1] 和 Mostow 的 BOGART^[2] 系统是两个例外。它们以推理踪迹的形式提供解释, 推理踪迹是对象事例的一部分。Fischer 认为交互设计环境不应仅仅访问先前设计的类别, 还应提供导致特定设计的推理。JANUS 系统采纳了分组问题求解的基于问题的观点, 向用户提供了在先前设计问题求解情节中提出的问题踪迹, 支持或反对各种设计选择的证据, 以及对设计决定的证明。Fischer 认为先前设计问题求解情节的基于问题的踪迹在新问题的环境中可使用户作出支持或反对特定的设计选择, 从而使用户创造出更有效的设计。

Mostow 对交互设计环境中设计事例的知识内容采纳了相似的观点。基于 Carbonell 的派生类比框架, Mostow 的 BOGART 系统用目标、操作符和启发式原则为用户提供了先前设计问题求解情节的踪迹。他认为先前设计事例的问题求解的派生记录可使用户更有效地从先前事例向新问题传递知识, 同时他证实了当交互设计环境中的知识系统提出设计问题的方案时, 系统应向用户提供引出方案的推理解释。

我们认为在这两个系统中使用的推理踪迹的有用性和可用性是有限的, JANUS 方案的不足在于它使用非正式语言描述踪迹。这种非正式规范在记录设计合理性方面, 即在一个组中所作决策的踪迹, 可能是最佳的。但是, 交互系统的问题求解的解释可能要求自动解释。BOGART 方法的困难在于它表达踪迹的抽象程度太低, 即它使用操作符、操作选择、操作选择的启发式原则, 这形成了一个较差的解释接口。因此, JANUS 用于描述设计问题求解的踪迹不正规, 以至于不能自动解释; 而 BOGART 的语言过于低层, 在交互环境中用处不大或不可用。使用 TMK 模型相信可以克服这两个缺点, TMK 模型具有恰当的抽象程度, 在交互设计环境中可使知识系

统清楚地解释推理。知识系统除了对推理进行解释外, 还能证实它提出的设计方案。

结论 在交互式基于事例的环境中, 设计的解释是一个重要问题。事实上, 要使设计系统实用化, 问题求解的解释是一个关键问题。

典型的交互设计环境将知识系统作为主要组件, 设计者可使用交互环境进行设计构造和试验, 知识系统可自动执行特定的和经选择的部分过程, 从而导致了人机协作设计。这引出了知识系统的适用性和学习性的问题。如果设计者不能方便地使用它们, 以及不能方便地学习如何使用它们, 则设计者就不会使用它们。如果设计者能够对系统如何工作、如何推理有一个形象模型, 且如果他们对系统生成的方案具有一定的信任, 则他们更愿意使用这些系统。

因此, 问题变成了知识系统如何为用户生成一个系统推理的形象模型, 系统如何解释推理和证实设计方案。知识系统的解释需捕捉推理的功能和策略内容, 以及系统的知识内容。TMK 模型允许这种任务层和知识层的解释, 这促进了系统和用户间的有效交流, 设计系统的解释需捕捉系统的功能、因果行为关系以及系统的结构, 而 FBS 模型在促进系统和用户间的有效交流的抽象层次上允许这种解释。

更进一步, 我们认为, 设计推理的解释应立足于发展的设计方案的环境。同样, 发展的设计方案的解释应立足于引出方案的设计推理的环境。首先, 在发展设计方案的环境中进行设计推理的解释使解释更有意义, 这是因为解释术语可以从它们当前涉及的设计的特定部分中得到它们的含义。其次, 在设计推理的环境中解释设计方案使解释更完整, 原因是在设计方案发展中以前解释所具有的适用性。

参 考 文 献

- 1 Goel A K, Murdock J W. Meta-cases: Explaining Case Based Reasoning. In: Third European Workshop on Case Based Reasoning, EWCBR '96 Lausanne, Switzerland, 1996
- 2 Goel A K, et al. Explanatory interface in interactive design environments. In: Fourth Intl. Conf. on Artificial Intelligence in Design, AID'96, 1996
- 3 Umeda Y, et al. Supporting conceptual design based on the function-behavior-state modeler. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 1996(10): 275~288
- 4 Simon H. The Sciences of the Artificial (2nd ed.), MIT Press, 1981 (下转第 50 页)

从功能角度看,基于线程的模型有两点优势:

1 基于小粒度的数据分解可以被无效率损失地实现。这对通信间计算量较小的问题如树搜索算法、整数计算、数据库查询系统格外重要。2 这种模式与 Client-Server 计算很相近,服务可通过使用线程登记机制输出、用类似 RPC 的方式调用;这种办法对非数值计算应用程序很有用、尤其那些数据库和事务处理领域。

性能方面,多线程能屏蔽计算-通信延迟。在传统的进程中,通信-计算-通信循环导致了空闲周期,但用多线程则可以高效率地使用 CPU,即一个线程阻塞了,又转而执行另一个线程,且这种开销比进程少得多。Sundraw 等人的初步实验表明,采用多线程可提高通信性能 35%^[1]。

3.2 定量分析

鉴于以上的定性分析,我们在由运行操作系统 AIX 4.1 的工作站 RS6000 组成的网络上做了对照实验:一台节点机 master 向另一台节点机 slave 发 n 次消息,slave 分别 1) 在 n 个进程中;2) 在一个进程中创建 n 个线程响应各消息,针对任务计算粒度分别为 $10^2, 10^3, 10^4, 10^5, 10^6$ 的情况,分别测得程序执行时间,如表 3-1 所示。

表3-1 不同执行方式的程序执行时间 (单位:秒)

执行方式 计算粒度	多进程并行	多线程并行
100次整数乘法	0.405821	0.075107
10^2 次整数乘法	0.448177	0.083401
10^4 次整数乘法	0.461189	0.087426
10^5 次整数乘法	0.586142	0.204409
10^6 次整数乘法	1.486253	0.585102

结果表明,采用 $n(=10)$ 个进程的并发执行时间是采用 n 个线程的 3~6 倍,进而定量地证明:多线程对提高并发计算的性能意义重大。

参考文献

- (美)Hwang 著. 高等计算机系统结构: 并行性, 可扩展性, 可编程性. 王鼎兴, 等译. 北京: 清华大学出版社, 1995
- Tanenbaum A S. Distributed Operating Systems. PRENTICE HALL, 1996
- Chandy K M, Kesselman C. CC++, a declarative concurrent object oriented programming notation. In Research Directions in Object Oriented Programming, MIT Press, 1993
- Gannon D, et al. HPC++, A Draft White Paper. In preparation, 1995
- Lopes L M B, Silva F M A. Thread-and Process-based Implementations of the pSystem Parallel Programming Environment. Software Practice and Experience, 1997, 27(3): 329~351
- Foster I, et al. Nexus: Runtime Support for Task Parallel Programming Languages. In: Intl. Workshop on Parallel Processing-1994. India, 1994
- Draft Version, TPVM: A Threads-based Interface and Subsystem for PVM. 1994. Available at: <http://uvcs.cs.Virginia.edu/~aifj/tpvm.html>
- Muller F. A Pthreads Library Interface. [Tech. Rep.] Department of Computer Science, Florida State University, July 1993

(上接第44页)

- Chandrasekaran B. Design Problem Solving: A task analysis. AI Magazine, 1990, 11: 59~71
- Chandrasekaran B. Functional representation and causal processes, in Advances in Computers, Vol. 38, M. C. Yovits, Ed. Academic Press, 1994, 73~143
- Lemke A C, Fischer G. A Cooperative Problem Solving System for User Interface Design. AAAI-90, 1990, 479~484
- Fischer G, et al. Supporting Indirect Collaborative Design with Integrated Knowledge-Based Design Environment. Human-Computer Interaction, 1992, 7(3): 281~314
- Mostow J. Design by Derivational Analogy. Issues in the Automated Replay of Design Plans. Artificial Intelligence, 1989