MVC 模式研究的综述*

任中方，张　华，闫明松，陈世福
（南京大学 计算机软件新技术国家重点实验室，江苏 南京 210093）

摘　要：随着面向对象技术的发展，MVC 的含义和用途变得更加广泛，不仅可以用于组件的构造，也可用于类似于电子商务应用等大型面向对象系统的软件设计。从 MVC 模式起源开始，讨论了 MVC 模式的结构、设计方法、实现技术、优缺点及其应用，最后介绍了由 JSP，Servlet 和 JaveBeans 实现的 MVC2 结构。

关键词：MVC；MVC2；设计模式
中图法分类号：TP311.52　　**文献标识码：**A　　**文章编号：**1001-3695(2004)10-0001-04

Overview of the Research in Model-View-Controller Pattern

REN Zhong-fang，ZHANG Hua，YAN Ming-song，CHEN Shi-fu
(*State Key Laboratory for Novel Software Technology*，*Nanjing University*，*Nanjing Jiangsu* 210093，*China*)

Abstract：As the object-oriented technique develops，the use of MVC became wider and wider，and no longer limited to the construction of component，it can also be used in the software design of large object-oriented system，such as electronic commerce. This paper starts with the origin of MVC pattern，discusses about the structure，design method，implementation techniques，good side and bad side，and the application of it. At end of this paper，MVC2 structure is discussed，which is construct of JSP，Servlet and JavaBeans.

Key words：Model-View-Controller(MVC)；MVC2；Design Pattern

早期的图形化程序设计常常围绕着事件驱动的用户界面来组织，这样的直接后果就是数据处理、程序功能与显示代码等部分完全纠结在一起[1]。大型的图形化程序中一个数据通常对应多种表示与处理方式，把特定界面绑定到应用程序上严重降低了程序的灵活性，使得一个很小的改动也牵涉到大量的代码，增加了程序开发与维护的工作量。

20 世纪 70 年代，MVC(Model-View-Controller)模式在 Smalltalk-80 的 GUI 设计中被提出。MVC 模式把数据处理、程序输入输出控制以及数据表示分离开来，并且描述了不同部分的对象之间的通信方式，使它们不必卷入彼此的数据模型和方法中，使程序结构变得清晰而灵活。由 JSP，Servlet 和 Java Beans 实现的基于 J2EE 的 MVC2 结构的出现使得 MVC 模式广泛地应用于大型的 Web 项目的开发中。本文重点介绍了 MVC 模式的结构、设计方法和实现的关键技术。

1　MVC 模式的结构

MVC 模式包括三个部分：模型(Model)、视图(View)和控制器(Controller)，分别对应于内部数据、数据表示和输入输出控制部分。一个更为合理的缩写应该是 MdMaVC[2]，其中，Md 指 Domain Model，是分析员和设计师所面对的部分，是对问题的描述；Ma 指 Application Model，用来记录存在的视图，获取视图信息和向视图发送消息。MVC 模式的一般结构[3]如图 1 所示。

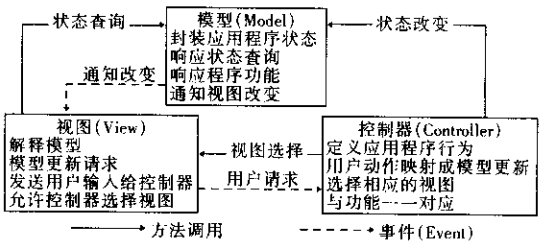

图 1　MVC 模式各部分的关系和功能

1.1　模型(Model)

模型是与问题相关数据的逻辑抽象，代表对象的内在属性，是整个模型的核心。它采用面向对象的方法，将问题领域中的对象抽象为应用程序对象，在这些抽象的对象中封装了对象的属性和这些对象所隐含的逻辑。模型的作用如下：①抽象应用程序的功能，封装程序数据的结构及其操作；②向 Controller 提供对程序功能的访问；③接受 View 的数据查询请求；④当数据有变化时，通知对此数据感兴趣的 View。

1.2　视图(View)

视图是模型的外在表现，一个模型可以对应一个或者多个视图，如图形用户界面视图、命令行视图、API 视图；或按使用者分类：新用户视图、熟练用户视图等。

视图具有与外界交互的功能，是应用系统与外界的接口：一方面它为外界提供输入手段，并触发应用逻辑运行；另一方面。它又将逻辑运行的结果以某种形式显示给外界。当 Model 变化时，它作出相应变化，有两种方法：Push(推)方法，让 View 在 Model 处注册，Model 在发生变化时向已注册的 View

收稿日期：2003-10-22；修返日期：2003-11-26
基金项目：江苏省高新技术研究资助项目(BG2003026)

发送更新消息;Pull(拉)方法,View 在需要获得最新数据时调用 Model 的方法[4]。View 的作用如下:①对数据的表现部分进行抽象;②将数据展现给用户,获得用户输入;③将用户输入转发给 Controller;(4)当接到来自 Model 的'数据已更新'通知后,更新显示信息。

1.3 控制器(Controller)

控制器是模型与视图的联系纽带,控制器提取通过视图传输进来的外部信息,并将用户与 View 的交互转换为基于应用程序行为的标准业务事件,再将标准业务事件解析为 Model 应执行的动作(包括激活业务逻辑和改变 Model 的状态)。同时,模型的更新与修改也将通过控制器来通知视图,从而保持各个视图与模型的一致性。Controller 的作用如下:(1)抽象用户交互和应用程序语义的映射;(2)将用户输入翻译成应用程序的动作,并转发给 Model;(3)根据用户输入和 Model 对程序动作的输出,选择适当的 View 来展现数据。

1.4 MVC 三部分之间的关系

如同一般的程序结构一样,MVC 有输入、处理、输出三个部分:Controller 对应于输入,Model 对应于数据表示和数据处理,View 对应于输出。其中,控制器与平台和操作系统的关系最为密切,View 与之部分相关(View 并不在意事件是来自 Microsoft Windows 还是 X Windows),Model 与平台无关[2]。

MVC 模式通常定义一个受限连接集合,用以描述模型、视图与控制器之间的通信,以及数据和控制信息的传递方式与方向。MVC 模式中三个部分之间的通信都按照这个集合的规定进行。针对图 2(a),这个受限集合通过图 2 中的箭头表现了出来,可以简要描述如下:

(1)模型—视图。模型处理数据,并根据其状态变化的情况将要显示的数据提供给视图,视图将数据组织成各种显示样式表现给用户。它们之间是一种典型的 Observer 模式[5]。

(2)控制器—视图。控制器根据用户输入直接调用不同视图改变响应流程,或与模型交互后获得需要显示的数据,再调用视图改变响应流程。它们之间是一种典型的 Strategy 模式[5]。

(3)控制器—模型。控制器与模型交互,控制器将输入数据传递给模型处理,控制器也可以从模型中抽取数据。

对于这个描述 MVC 通信的受限连接集合的定义,可以在实际应用中根据具体应用环境的不同作合理调整,如在很多应用中采用图 2(b),图 2(c)所示的两种通信形式。图 2(b)中,视图承担的角色与图 2(c)及图 2(a)中有重要区别,它直接接收用户输入并向用户提供数据表示;图 2(c)与图 2(a)的重要区别是,取消了视图与模型的通信,由控制器与模型交互,抽取需要显示的数据并传递给视图,视图只负责处理数据表示。MVC ++[6] 就是这样的结构。

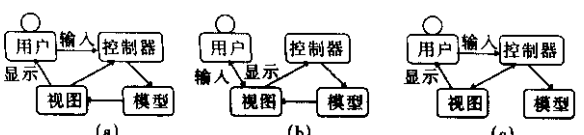

图 2 MVC 的几种结构

2 MVC 模式的优缺点

2.1 MVC 模式的优点

分离稳定的代码和易变的代码是面向对象设计的一个基本原则。通常负责控制部分的对象要比负责表现部分的对象稳定,而负责业务逻辑和业务数据的对象比前两类对象更稳定。MVC 设计模式分离了程序的表现、控制和数据,具有设计清晰、易于扩展、运用可分布的特点,因此在构建 Web 应用中具有显著的优势,可适用于多用户的、可扩展的、可维护的、具有很高交互性的系统,如电子商务平台、CRM 系统和 ERP 系统等。它可以很好地表达用户与系统的交互模式以及整个系统的程序架构模式。MVC 模式有如下优点:

(1)将数据建模、数据显示和用户交互三者分开,使得程序设计的过程更清晰,提高了可复用程度;

(2)当接口设计完成以后,可以开展并行开发,从而提高了开发效率;

(3)可以很方便地用多个视图来显示多套数据,从而使系统能方便地支持其他新的客户端类型;

(4)模式中各组件的分界线就是很自然的分发接口点,使得应用程序的发布更容易,并且支持渐进式升级;

(5)各部分的责任划分得很清楚,从而简化了测试工作,维护人员很容易了解程序的结构,便于维护工作的进行;

(6)提高了系统灵活性,因为数据模型、用户交互和数据显示等部分都可以设计为可接插组件;

(7)可以用于分布式开发,只要给 Model,View 和 Controller 使用代理(Proxy)[4],就可以封装不同计算机之间的通信。

2.2 MVC 模式的缺点

MVC 模式的缺点主要集中在两个方面:

(1)由于实施 MVC 模式过程而产生的开销。设计 MVC 模式需要有经验的分析人员对系统进行分析,类的数量及文件数量的增加(像C ++ 这样的编程语言,一个类对应两个文件)。

(2)由于设计 MVC 模式时分析不够、设计不当而引起相反的效果,把属于一个模块的分开,把不相干的模块聚在一起。对属于一个实体不同方面的严格区分导致了一个紧凑结构,使得测试和维护的工作量大幅度增加,每一次变动牵涉到许多本不相干的模块的变动。

3 MVC 模式的设计方法[3]

MVC 模式将目标系统分为三个部分:模型、视图和控制器,并要定义描述三个部分之间通信的受限连接集合。相应地,MVC 模式的设计包括四个部分:MVC 通信的设计、模型的设计、视图的设计和控制器的设计。

3.1 MVC 通信的设计方法

在 MVC 模式中,模型、视图和控制器这三个部分之间的通信是通过定义一个受限连接的集合来描述的。如图 1 和图 2 所示的那样,可以采用带箭头连线对在各个部分之间进行的信息和控制传递作出形象的表示,并具体描述每个带箭头连线的

含义。在实现上,模型、视图和控制器这三个部分之间的通信是通过相应的模型对象、视图对象和控制器对象之间的消息通信完成的。原则上应该尽量减少三个部分之间的结合,增强隔离性,使模型部分可以不必了解数据表示和与用户交互的细节就可以完成数据处理,使视图不必了解模型进行数据处理的内部实现就可以完成数据表示的功能。

这个受限连接的集合可以表示为 {((P1,P2),D,(M,…)),…},其中,(P1,P2)表示从 P1 部分到 P2 部分的通信,P1,P2 代表模型、视图或控制器;D 是对从 P1 到 P2 方向的通信的详细描述;(M,…)表示一个有限方法集合,M 是实现数据或控制传递的方法;((P1,P2),D,(M,…))三元组完整地描述了从一个部分到另一个部分之间的通信。

3.2　模型的设计方法

模型是进行数据处理的地方,表达了实际应用中的业务处理逻辑。模型由许多模型对象及这些对象之间的各种各样的关系构成。模型对象是根据问题域中的对象构建的,带有一些有意义的属性和方法,它们之间有这样或那样的关系,如一般特殊关系、整体部分关系、实例关联、消息关联等,它们相互交互完成数据处理功能。

3.3　视图的设计方法

视图是表达数据显示逻辑的地方,通常完成或辅助完成 GUI 功能。视图对象通常用于保存一些从模型对象中得到的信息,但视图对象并不完全与模型对象相同。模型中有的对象在视图中可能没有,与模型对象对应的视图对象一般也具有与之不同的属性和方法,反之亦然。主要有两种方式实现视图对象:(1)视图对象是模型对象的一个代表,持有对相应模型对象的一个引用,相应地,在视图对象的方法中通过这个引用将调用都传递到模型对象中去。(2)视图对象不必实现与模型对象相同的接口方法,它拥有的是一些属性,反映了满足显示需要的数据,并可以从模型对象中获取相应的值。在构建视图对象的时候就使用实际模型对象初始化视图对象的状态,也就是给视图对象的属性赋值,实现数据传递。

3.4　控制器的设计方法

控制器在用户和视图之间交互,完成接收用户输入并控制显示流程转换的功能,它充当了模型与视图之间的去耦层。控制器中的对象用于控制对用户输入的响应方式,是根据对响应方式和响应流程的定义而获得的对象。根据不同的 MVC 通信设计,控制器对象可以与模型对象、视图对象进行不同形式的通信。

针对不同的情况,控制器对象通常有两种实现方法:(1)采用的设计开发环境提供了对控制器的支持,对控制器对象的实现形式以及它与模型对象和视图对象的交互方法提供了专门的支持,如在采用 JSP/Servlet/Java 方式进行开发时,Servlet 对象充当了控制器对象的角色,并且有相应的方法支持与 JSP 和用 Java 实现的模型对象进行通信。(2)控制器作为模型和视图之间的一个去耦层实现,表现为一些普通对象,但完成去耦功能。

4　MVC 的实现技术

实现 MVC 模式时面对的主要问题是 Model 与 View 的关系,这可以使用 Observer 模式[4] 来实现。为了满足分布式的要求,可以使用 Proxy 模式[4] 来封装本地操作和远程。

4.1　相关模式:Observer(观察者)

Observer 模式用于描述 Model 与 View 的更新、访问关系。它定义了对象间的一种一对多的依赖关系,当一个对象的状态发生改变时,所有依赖于它的对象都得到通知并被自动更新,所以又称为依赖(Dependents)。这一模式中的关键对象是目标(Subject)和观察者(Observer),一个目标可以有任意数目的依赖它的观察者。一旦目标的状态发生改变,所有的观察者都得到通知,作为对这个通知的响应,每个观察者都将查询目标以使其状态与目标的状态同步,这种交互也称为发布 - 订阅(Publish-Subscribe)。目标是通知的发布者,它发出通知时并不需知道谁是它的观察者,可以有任意数目的观察者订阅并接收通知。Observer 模式的一般结构如图 3 所示,在 Subject 中保存 Observer 的引用,当需要更新时,调用 Notify() 通知所有 Observer。

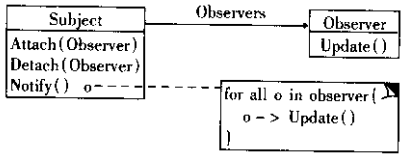

图 3　Observer 模式的结构

当目标和观察者间的依赖关系特别复杂时,需要一个维护这些关系的对象 ChangeManager。ChangeManager 将一个目标映射到它的观察者并提供一个接口来维护这个映射,这就不需要由目标来维护对其观察者的引用,反之亦然。使用它的目的是尽量减少观察者反映其目标的状态变化所需的工作量。例如,如果一个操作涉及到对几个相互依赖的目标进行改动,就必须保证仅在所有的目标都已更改完毕后,才一次性地通知它们的观察者,而不是每个目标都通知观察者。另外,观察者通常并不对所有事件都感兴趣,可以扩展目标的注册接口,让各观察者注册为仅对特定事件感兴趣,以提高更新的效率。当一个事件发生时,目标仅通知那些已注册为对该事件感兴趣的观察者。支持这种做法的一种途径是使用目标对象的方面(Aspects)的概念,这样就可以实现不同的观察粒度。可以在将 Observer 添加到 Subject 上时使用 Aspect 作为参数,将观察者对象注册为对目标对象的某特定事件感兴趣,通知时将这方面的改变作为 Update 操作的一个参数提供给它的观察者。

4.2　相关模式:Proxy(代理)

为了实现分布式体系,需要区分是远程操作还是本地操作,这种区分如果体现在功能代码中就会影响代码的清晰度,而且不便于维护。面向对象设计的一个思想就是“封装变化”,即把易变化的部分封装在一个类中,这样结构清晰。Proxy 就使用这样一个机制,使得功能代码编写时可以不考虑是本地操作还是远程操作。Proxy 又名 Surrogate,Proxy 模式的

结构如图 4 所示。Proxy 和 RealSubject 都继承自 Subject，Proxy 中组合了 RealSubject（本机或者远程的）。当 Client 调用一个 Request() 的时候，不需要关心 Subject 是 RealSubject 还是 Proxy，这样就可以实现远程调用的封装。

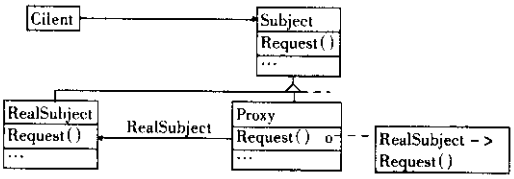

图 4 Proxy 模式的结构

5 MVC 模式的应用

由于具有使用户界面（View）、控制（Controller）与数据（Model）分离开来的特性，MVC 很适用于大型的 GUI 软件的开发。目前许多 GUI 软件的结构都是基于或者是部分地基于 MVC 的。一些比较常见的例子如下：

（1）Microsoft Foundation Class（MFC）的基本结构是文档/视图结构，它是 MVC 结构的变体。它把 View 和 Controller 集成在视图内，文档负责数据的表示以及存取（Model），视图负责显示文档内容（View）以及处理用户界面上的操作（Controller）。一个文档可以对应多个视图，每个视图可以采取不同的表现方式，如同一份数据，可以在一个视图中使用文本方式显示，一个视图中使用表格方式显示，另一个视图使用图表的方式显示。当用户在某一个视图中改变了文档的内容，可以使用 UpdateAllViews() 方法发消息给该文档的所有视图，使得各个视图保持一致。

（2）Java 的 JFC 同样使用了 MVC 结构，其中，Model 对应于数据类或者是用户定义的结构，如 JList，JTable，JTree，JTextPane 等，View 对应于 JFC 部件的子类，Controller 对应于 Listener 的子类，如 ChangeListener。在 Swing 开发包所提供的组件中，一些组件将视图和控制器联合成一个对象叫做 Delegate（代表）。Delegate 像视图一样可以表示模型，又像控制器一样把用户输入传递给模型，它具有视图和控制器的双重功能。

（3）Velocity 是 Apache-Jakarta 的项目计划之一（Apache-Jakarta 是 Apache Software 的一个研究开发 Java 产品的工程，主要为 Java 开发者提供各种开发工具及软件的框架），它是一种基于 Java 的模版引擎。在使用 Velocity 的 J2EE 环境中，使用模板作为视图，将 EJB 作为模型，采用 Servlet 作为控制器，可以实现良好的 MVC 模型。由于 Velocity 将 Java 代码从页面模板分离，实现了对 MVC 软件设计模型的良好支持，具有比 JSP 等后台技术更多的优点[7]。

（4）Struts 是 Apache-Jakarta 的项目计划之一，它是 Jakarta 工程提供的一个用于开发 Web 应用软件的框架，它采用 MVC2 模型作为基本结构，并在 MVC2 的基础上加进了 Tag Library 用以分离 HTML 与程序控制语言。Struts 采用 EJB 作为模型，JSP 和 Custom Tag Library 配合作为 View，Servlet 作为控制器[8,9]，提供了对开发 MVC 系统的底层支持。

6 MVC2 的结构

在早期的 Web 项目的开发中，程序语言（CGI 和脚本语言等）和 HTML 的分离一直难以实现，而且由于脚本语言的功能相对较弱，缺乏支持 MVC 设计模式的一些必要的技术基础。所以尽管 MVC 设计模式很早就提出，但在 Web 项目的开发中引入 MVC 却是步履维艰，直到基于 J2EE 的 JSP Model 2（MVC2）的问世才得以改观。MVC2 使用 JSP 技术实现视图的功能，使用 Servlet 技术实现控制器的功能，用 EJB 技术实现模型的功能[9~11]。

MVC2 是面向 Web 应用软件开发的 MVC 模型，它有三个主要组成元素：JSP，Servlet 和 JavaBeans。其中，JSP 对应于 MVC 模型的 View，Servlet 对应于 Controller，Bean 对应于 Model。Model 的 Bean 又分为逻辑 Bean 和数据 Bean，逻辑 Bean 用于事务处理，数据 Bean 用于保存数据。MVC 模型 2 的结构如图 5 所示，其工作流程为：Servlet 接收客户端请求，Servlet 把接收到的数据保存到数据 Bean 中，逻辑 Bean 进行数据处理，Servlet 根据逻辑 Bean 的处理结果，调用相应的 JSP，JSP 生成 HTML 页面，并返回给客户端。

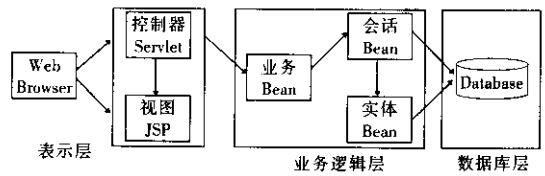

图 5 MVC2 的结构

MVC2 和 MVC 的主要区别是：在一般 MVC 模型中，View 和 Model 间是登录（Register）与通知（Notify）的关系，当 Model 对象的数据发生变化时，通知已登录的 View 对象，显示新数据，这就是所介绍的 Observer 的设计模式；而在 MVC2 中，View 和 Model 间没有采用 Observer 模型，这是由 Web 应用软件的特点决定的。因为在 HTTP 协议中，客户端发出请求，收到服务器返回的数据后，客户端和服务器之间的连接就断开了。

7 结束语

MVC 模式由于其结构清晰等特性，在当前大型软件开发中的应用日渐广泛，MVC2 的出现使得 MVC 模式很好地应用在 Web 项目的开发中，而 MVC2 结构中并没有采用 Observer 模式。我们在开发江苏省高新技术项目“面向集成应用软件的工作流技术”中对 MVC 模式和 MVC2 结构进行了深入研究，借鉴 MVC 模式和 MVC2 架构的一些优点，并在其基础上提出将消息管理机制从 Model 中独立出来的方法，以更通用的消息订阅机制来实现消息和数据的传递，从而更好地适应分布式系统的需要。

参考文献：

[1] 透明. Model-View-Controller 模式[J]. 程序员，2002，(10)：55-56.

[2] John Deacon. Model-View-Controller（MVC）Architeture[EB/OL]. http://www.jdl.co.uk/bridfings/MVC.pdf，2000.

[3] 姚延涛，王煜，沈钧毅. 采用增强的 MVC 模式提高 （下转第 8 页）

网络如图 5 所示。

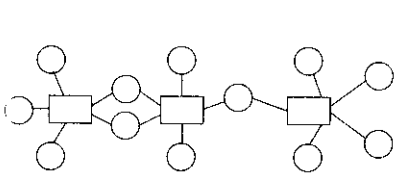

图 4 横向扩展共振系统

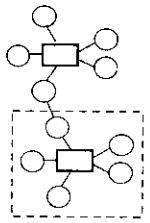

图 5 纵向扩展共振系统

图 5 中的虚线框中表示的是各组成部分的属性值与事物的某一属性(用粗线连接)之间的关系。图 5 中只给出一个属性与组成部分属性之间的关系,其他属性与组成部分之间的关系可同样处理。由于各类属性值(包括部分属性值)之间的关系是通过共振信息表达,共振信息的传递是双向的,所以在该网络系统中可实现整体到局部和局部到整体的自由推理,当然也实现逆向推理。

(4)可实现内共振。这里所说的内共振是指共振信息传到共振属性元后,共振属性元产生属性值,该属性值又传到共振网络,共振网络再产生共振信息,如此循环下去直至达到要求为止。特别是在多步推理时更是需要,可以利用共振信息传递的双向性,实现演绎推理和逆向推理。其实只需将共振属性元层次网络总算法中的第(5)步改为检测输出是否符合要求,是则结束算法,否则转第(2)步执行,这样即可实现属性值间的自由共振。

(5)不完全属性学习。在共振属性元层次网络中,不管输入样本含有多少个属性值(或数据),系统都可以正常运行。这是目前其他系统所无法比拟的。

(6)以属性值及其关系表征概念。在语义网的基础上发展起来的概念图及描述逻辑将概念和规则作为最小逻辑存储单元,不可避免地出现相同属性值的重复存储,当数据量不断增加时,系统的运行速度将受信息检索速度的严重制约。但在共振层次属性网络中,由于将概念的 n 个属性分到 n 个属性元处理,以属性值及其关系表征概念,不会出现相同属性值重复存储的现象,且各属性可并行处理,从而至少降低这种制约一个数量级。

4 总结

神经科学早已证实在人脑中有脉冲信号,有电波存在,脑中的信息是否通过电波传递?虽然没有肯定的答案,但也不能否定其存在。本文在假设脑中的信息是通过电波之间的共振来传递的基础上,以属性论为基础构建起一种集信息存储与推理于一体的共振属性元层次网络,该网络将对事物整体存储转换为属性分类存储,事物的各属性通过属性值之间的关系表现。不出现属性值冗余存储,是一种全新的知识表示和存储方法,由于将事物存储转换为属性值存储,从而有效避免了组合爆炸现象。由于推理通过共振信息传递,省去了烦琐的检索和逻辑推理,所以速度是其他系统不可比拟的。

参考文献:

[1] 史忠植. 高级人工智能[M]. 北京:科学出版社,1998. 126-180.

[2] Greene R L. Efficient Retrieval from Sparse Associative Memory[J]. Artifficial Intelligence,1994,66:395-410.

[3] Omori T, Mochizuki T, et al. Emergence of Sysbolic Behavior from Like Memory with Dynamic Attention[J]. Neural Networks, 1999,(12):1157-1172.

[4] Murdock Jr B B. TODAM2:A Model for the Storage and Retrieval of Item, Associative and Serial _ order Information[J]. Psychological Review,1993,100(2):183-203.

[5] Tru H Cao. A Formalism for Representing and Reasoning with Linguistic Information[J]. International Journal of Uncertainty, Fuzziness and Knowledge-based Systems,2002,10(3):281-307.

[6] Baader F, et al. The Description Logic Handbook:Theory, Implementation and Applications[M]. Cambridge:Cambridge University Press, 2002. 29-83.

[7] Ralf Kusters, Ralf Molitor. Approximating Most Specific Concepts in Description Logics with Existential Restrictions[J]. AI Communications,2002,15:47-59.

[8] 董占球,等. 按模式记忆[J]. 计算机学报,1991,14(4):316-318.

[9] 陈苒,董占球. 按模式记忆理论的记忆结构刻画[J]. 计算机研究与发展,2000,37(5):634-640.

[10] 宣士斌,冯嘉礼. 行为属性决定的事物间关系及其属性坐标表示法[J]. 南京大学学报(自然科学),2000,36(计算机专辑):8-13.

[11] 宣士斌,冯嘉礼. 属性神经网络模型[J]. 计算机研究与发展,2002,39(11):1442-1446.

[12] 周志华,陈兆乾,东世福. 基于域理论的自适应谐振神经网络研究[J]. 软件学报,2000,11(11):1451-1459.

作者简介:

宣士斌,副教授,主要从事人工智能及计算机应用研究。

(上接第 4 页)面向对象应用能力[J]. 小型微型计算机系统,2002,(12):23-25.

[4] Erich Gamma, Richard Helm, Ralph Johnson, et al. 设计模式:可复用面向对象软件的基础[M]. 李英军,等. 北京:机械工业出版社,2000.

[5] 胡文华,李建民,胡振鹏. 模式与设计模式[J]. 计算机与现代化,2002,(12):12-15.

[6] Ari Jaaksi. MVC ++ Application Architecture[EB/OL]. http://www.cs.uta.fi/~jyrki/ohto02/mvc.ppt,2003.

[7] 邢昊,张凌,张平. 基于 Velocity 的 J2EE 开发模式及其应用[J]. 计算机应用,2003,(1):47-48.

[8] 胡长城. Struts 的体系结构[J]. 程序员,2002,(10):57-60.

[9] 何成万,余秋惠. MVC 模型 2 及 Struts 软件框架的研究[J]. 计算机工程,2002,(5):274-275.

[10] 陆荣幸,郁洲,等. J2EE 平台上 MVC 设计模式的研究与实现[J]. 计算机应用研究,2003,20(3):144-146.

[11] 袁梅冷,黄烟波,等. J2EE 应用模型中 MVC 软件体系结构的研究与应用[J]. 计算机应用研究,2003,20(3):147-149.

作者简介:

任中方(1980-),男,硕士研究生,主要研究方向为人工智能;张华,硕士研究生,主要研究方向为人工智能;陈世福,男,教授,主要研究领域为人工智能理论应用。