扩展 MVC 模式在空管系统的应用①

The Application of Extend MVC in ATC System

张 芳　刘淑芬　（河南理工大学计算机科学与技术学院 河南 焦作 454000）

摘　要：空中交通管制系统属“专用、实时、使命重大”型系统，对系统的实时性和稳定性要求很高，选择怎样的软件体系结构对于设计出良好的空管人机界面系统很重要。MVC 这种软件体系结构可以很好的解决空管系统的这些问题，MVC 软件体系结构被广泛地应用于图形化用户系统。本文介绍了 MVC 软件体系结构，并对 MVC 的三层体系结构进行了扩展，开发了适合空中管理特点的空中交通管制系统。

关键词：空中交通管制系统 MVC Model View Controller

1　空中交通管制系统

1.1　空管概述

空中交通管制（ATC）就是防止航空器之间相撞，防止航空器与地面障碍物相撞，维持空中的交通秩序，保证有一个快速高效的空中交通流量。空中交通管制包含区域管制、进近管制、塔台管制和空中交通报告服务四部分。为有效地保护和促进空中交通安全，维护交通秩序，保障空中交通畅通，提供指挥调度、管理监测手段而建立的系统就称为空中交通管制系统。空中交通管制人员必须严格掌握飞行动态、及时通报飞行情况、调配飞行冲突、有效地监督空中飞行活动、妥善安排飞机起飞和着陆顺序、减少飞机延误和等待、提高飞机和机场的利用率、防止飞机与各种飞行器或地面障碍物相接、防止飞行器擅自进入飞行禁区和飞越国界，以及为制定飞行计划和国防安全提供可靠的资料和情报[1]。因此，空中交通管制系统是一个对人机交互要求较高的专用实时系统，它通过多个台位的相互协作来共同完成对空中交通的管制过程，管制员指挥空中交通的过程就是与系统进行交互的过程。为了达到指挥空中交通的目的，管制员需要向系统查询多种航行信息、编辑飞行计划、操作命令处理等，必要时还应与其他系统通信，这些都需要通过操作空管系统软件来实现，所以对空管系统的稳定性、健壮性、易操作性和实时性的要求就特别高。空中交通管制系统属 SRM（Special purpose，Real－time，Mission Critical——“专用、实时、使命重大”型系统。因此对整个软硬件系统的实时性、可靠性、稳定性、安全性的要求很高。

1.2　空管特点

由于空中交通管制系统是时实专用的系统，它有别于其他一般的管理系统，有以下几方面的重要特点：

（1）准确，时实，高效，任何不准确的或延迟的飞行情报数据将会导致管制员的判断失误，带来灾难性的后果。

（2）飞行数据多样化，信息量大，处理各类复杂的飞行计划信息需要具有复杂的业务逻辑。

（3）管制员控制作业流程的各个环节都需要频繁的人机交互，系统开销大。

2　MVC 模式

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，这在大型的图形化程序中严重降低了程序的灵活性，增加了程序开发与维护的工作量。MVC 很好的解决了这个麻烦。

MVC 是一种软件体系结构，它很好的实现了模型和视图的分离。它把软件结构抽象为三个层次：Model 层，View 层和 Controller 层[2]：

Model 层封装了软件的核心数据结构和逻辑功

① 项目基金号：2004BA907A20 国家科技攻关计划项目

能，是独立于外部表象的内在抽象。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[2]。

View 层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，并触发相应应用逻辑运行；另一方面它又将逻辑运行的结果以某种形式显示给外界。

Controller 层是 Model 层和 View 层的纽带，它解释从 View 层上传过来的用户操作意图，作相应解释处理并交给 Model 层去执行，Controller 层还负责确保 View 层和 Model 层之间的对应关系，协调 View 层和 Model 层的工作。

Model，View 和 Controller 的关系如图 1 所示。

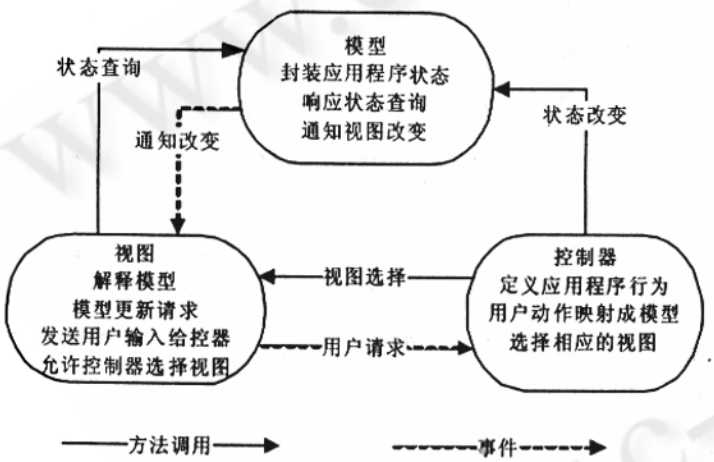


图 1　MVC 模式各部分的关系和功能

3　扩展的 MVC

3.1　问题的提出

MVC 把对象结构抽象为三个层次：Model 层，View 层和 Controller 层，可以方便的维护业务流程，缩短业务应用系统的开发周期，当各个层次任务均匀的时候，这样的层次划分可以很好的实现各层之间同步，保证整个系统的协调统一，但是当 Model 层，View 层和 Controller 的某个层次由于处理业务繁重而变得十分臃肿的时候，那么这个层次就会由于负载过会导致整个系统时实失效。

对于空管系统的特点有两个主要问题：1、View 层人机交互频繁，此层的业务处理繁重；2、Model 层数据处理的业务量也很大。这些问题的解决办法是怎样将负载严重的层次分割开来，进行分布式的处理，以更好的提高系统的效率。

3.2　扩展的 MVC 模式

对于空管系统来说将 View 层分割开是不符合业务逻辑的，Model 层是最复杂的层次，它为整个系统完成各种数据处理功能，提供各部分所需要的各种类型的数据。然而在空管系统中许多和显示操作相关的状态和功能并不和底层的商业逻辑一致[3]，因此结合空管系统的特点，在系统的开发时将和显示操作相关比较密切的一层单独提出来称为 Presentation Model，将和底层数据处理相关的操作抽象出来形成 Concept Model。

Presentation Model 直接存放和 View 层显示相关的数据及一些数据处理操作。

Concept Model 和底层数据交互，主要的数据处理放在这个对象层次上，可以利用分布式系统的特点，把这个层次的商业业务逻辑分布到分布式系统的各个客户端，分散系统开销，实现系统的高效性。

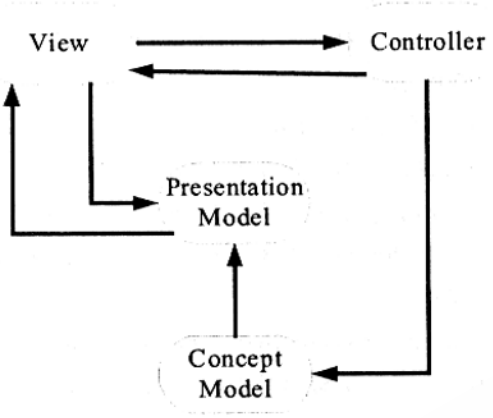
View 层与 Presentation Model 通讯，View 层读取 Presentation Model 层的数据，Concept Model 层的具体数据处理业务逻辑对于 View 层是透明的。具体到系统的应用层上，和系统显示相关的应用层面并不和底层的异构的体系相关联，降低了应用层面的开销。

Controller 层控制 View 层的显示，并响应 View 层的用户请求，通知 Concept Model 层将需要的数据进行处理并移交到 Presentation Model 层。

3.3　扩展的 MVC 模式在空管中的实现

空管系统中 View 层即人机界面层，是一系列图形用户界面，它把空管系统的各种数据表现为不同形式的数据表现形式，方便用户查看，主要包括：触摸屏，表页屏和态势屏。触摸屏是用户操纵系统各个逻辑功能的主菜单，用户通过触摸不同菜单项来实现对整个系统的操控，通过 TouchMain 来实现；表页屏以用户看得

懂的各种数据形式用来显示各种数据信息，主要有 Form 类来实现，；态势屏则是真实世界在屏幕上的模拟显示。



Concept Model：概念模型　Controller：控制
presentation Model：表现模型　View：视图

图 2　扩展 MVC 模式各部分关系视图

空管系统 Controller 负责对用户请求事务和后台运行任务的逻辑控制，实现系统管理（参数设置、状态监控和记录重演）、交通指挥、目标管理、通信管理、态势生成（目标数据融合与综合识别）等。Controller 对象又以下几方面组成：

Controller 层对象主要包括：表页控制，态势生成，报文和控制台命令分发控制等。

表页控制：处理与表页屏信息显示相关的操作。

态势生成：处理态势屏相关信息的显示。

控制台命令分发控制（Dispacher 类）则用来分发收到的报文，注册台位信息和相关的菜单类，注册报文和控制类的对应关系。

Controller 类是所有控制类的基类，它有两个虚方法：void ProcessMessage（CMessage ＊ pmessage）和 virtual void Execute（long menuID），ProcessMessage 方法负责处理和报文消息相关的消息，Execute 方法负责处理触摸屏发送的控制命令。

Presentation Model 层是一系列和显示密切相关的实体的集合，其基类是 PresentationObject class。

Concept Model 层是和数据处理相关的操作的集合，其基类是 ConceptObject class。

3.4 Presentation Model 和 View 层的同步

利用发布－订阅（Observer）模式可以很好的解决模型和视图的同步问题，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（Publish2 Subscribe）[5]。目标（Subject）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（Observer），可以有任意数目的观察者订阅并接收通知。在 Subject 中保存 Observer 的引用，当需要更新时，调用 Notify（）通知所有 Observer。

3.5 Presentation Model 和 Concept Model 层的交互

PresentationObject 定义了 Declare（）方法用来注册 PresentationObject 和 ConceptObject 的相关属性的对应关系列表，当 ConceptObject 的属性发生改变时 PresentationObject 得到通知改变与其对应的属性，ConceptObject 的 Getter（）和 Setter（）方法分别处理当读取或设置 ConceptObject 属性时所触发的操作。PresentationObject 的 Draw（）方法和系统的图形接口 API 交互。

3.6 Controller 和 View 层的交互

根据“迪米特法则”：在系统中各个对象之间应尽量减少直接调用，如果一定要发生调用关系，应尽量通过第三方进行通信。空管系统在 View 和 Controller 层之间加上了一个分发器用来充当第三者身份，交互图如下图所示。

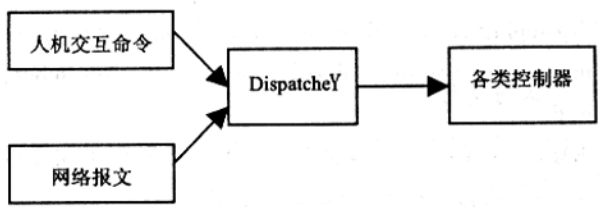


图 3　分发器交互图

Dispatcher 类的 void RegisterMCmap（QString mcMapFileName）方法用来注册触摸屏菜单的 menuID 和控制类之间的映射和报文和控制类之间的映射；void

ExecuteCommand(long menuID) 用来在此方法中实现执行触摸屏菜单项命令;void ExecuteMessage(short messageID,CMessage * pMessage) 用来在此方法中实现处理报文。

在 Dispacher 类的 ExecuteMessage (short messageID,CMessage * pMessage) 方法会根据报文的 ID 来执行相应的报文控制类的 ProcessMessage (CMessage * pmessage) ;在 Dispacher 的 ExecuteCommand (long menuID) 方法会根据传来的 menuID 来执行相应的控制台命令控制类的 Execute (long menuID) 方法,于是就实现了控制类和控制类之间的通信。

4 总结

在实时性要求特别高的空管系统中应用 MVC 模式,保证了数据的同步性和统一性,是实践证明的高效的软件体系结构。基于 MVC 模式开发的系统软件结构清晰,缩短了开发周期,提高了软件的可维护性和代码复用率。

MVC 模式的层次结构只是一个概念上的抽象,并不是一个开发软件的约定,其中的框架结构和层次划分可根据所开发的系统的功能特点不同,进行灵活的设计。本文介绍的这种分层方式迎和了空管系统的特点,是对 MVC 模式的扩展,是一种扩展的 MVC 模式。

这种扩展的 MVC 模式更贴近空管系统的要求,对于设计其他系统是一个借鉴和启发。但是 Concept Model 层与底层的交互有待进一步完善的,可以把现时流行的数据访问中间件技术应用在其中,解决异构分布系统的数据源操控访问问题。

参考文献

1 王秀林,软件重用技术在空管系统中的探讨,现代电子工程,1998(1).

2 任中方等,MVC 模式研究的综述[J],计算机应用研究,2004,20(10):1~3.

3 陈乐、杨小虎,MVC 模式在分布式环境下的应用研究[J],计算机工程,2006.10, Vol.32 No.19.

4 袁梅冷、黄烟波、黄家林等, 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).

5 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,2001.

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

7 王明军,空管系统基于面向对象技术的开发及实时系统界面模型的应用与研究[硕士论文],2004.

8 Matt Raible. Comparing Web Frameworks: Struts, Spring MVC, WebWork, Tapestry&JSF DB/OL. http://www. Raibledesigns. com. 2006 -05 -06.

9 Burbeck S. Applications Programming in Smalltalk - 80: How to Use Model - View - Controller (MVC) [EB/OL]. http://st - www. cs. uiuc. edu/ users./ smarch/st - docs/mvc. html,1992.

10 吴宏森、宋顺林,MVC 架构在工程项目管理系统中的应用[J],微计算机信息, 2006 Vol.22 No.22.