文章编号:1000-5862(2017)03-0308-06

# 面向消息驱动框架的实体管理模块设计与实现

王永近[1],李　廉[2*]

(1. 合肥工业大学工业与装备技术研究院,安徽 合肥　230009;2. 合肥工业大学计算机与信息学院,安徽 合肥　230009)

摘要:探讨了一种基于消息驱动的框架(MDF,Message-driven Framework)的软件开发方法,旨在一些互联网多用户的应用系统中进一步提高软件开发效率. MDF 设计被分为 3 个部分:实体管理、消息管理和数据显示管理. 主要研究了实体管理模块,在该模块中,系统提供实体的规范和模板,用户需要根据表单注册信息,然后 MDF 会根据注册信息自动将实体模板转换为实体实例. 从设计模式和开发方法的角度看,MDF 具有更便捷、更高效率的特点,并开发了相应的中间件来支持系统的运行,最后通过 1 个案例说明该方法的特点

关键词:消息;消息驱动,多用户应用系统;互联网应用;软件开发方法

中图分类号:TP 311　　文献标志码:A　　DOI:10.16357/j.cnki.issn1000-5862.2017.03.18

## 0　引言

自 1946 年 2 月 14 日,世界上第一台电子计算机 ENIAC 在美国宾夕法尼亚大学诞生以来,计算机的发展经历了 70 年的历史. 软件开发方法伴随着计算机程序语言的诞生而诞生,是实现编程人员和机器之间进行对话的重要工具. 软件的开发模式可以分为基于静态结构的开发方法、基于动态结构的开发方法和基于演化结构的开发方法. 较为传统的程序开发属于基于静态结构的开发,这种程序的内部语句逻辑是固定的,不随外部运行环境的变化而变化;在互联网时代,程序开发方法大多属于基于动态架构的开发,这种程序考虑外部运行环境,根据环境的变化调整运行逻辑,网络环境中的应用程序多属于这种架构. 基于演化结构的开发方法是更新的一种,软件系统投入执行之后,根据环境条件和需求状况的改变可以对软件本身进行修改[1-4]. 文献[5]提出反应式编程在事件驱动和交互式应用程序开发中的“六轴”评价,其中分布式和并行性具有极其重要的地位. 文献[6]提出通过设计不同的引擎和模块,响应用户对于不同编程粒度的需求,通过使用消息驱动方式实现了一种细粒并行处理机系统.

由于软件开发面临越来越多元化的要求和越来越复杂的环境,因此软件应用架构的概念随之出现. 目的是为一类应用程序提供简便的解决方案,它为应用软件的开发提供了整体的框架和统一的标准,因此在较大程度上节约了开发的成本和减少了开发的错误. 软件应用架构是一个整体的结构,配备有相应的环境描述要求,以及实现这些要求的中间件. 用户按照一定的要求补充具体应用的细节,就形成了具体的应用软件.

一般而言,软件应用框架采用面向对象的技术,将事先设计好的操作和通信模式封装在对象里,通过实际开发人员的具体指定实现这些对象的实际操作定义,从而成为在给定环境可用的软件. 这种软件开发的模式具有较强的可扩展性、可一致性和可维护性,因此是当前软件开发中主流模式.

具体地讲,软件应用框架就是一组具有较好逻辑性、并且被相互关联的类和类库,但是它并不包含实际的对象. 类和类之间彼此关联或包含,并且彼此交互和合作,所以它们都不能单独使用,而必须共同使用. 此外,用户不能修改框架中的任何类,而只能从框架中的基类派生出自己的具体应用类,当程序运行起来后由框架按需要自动创建这些类的对象并调用它们,或者由用户手工创建并调用它们[7].

本文在文献[4-8]的基础上,探讨基于消息驱动的软件开发框架(MDF,Message-driven Framework). MDF 从概念上已经有较多讨论,并且提出了较多实现方式,这些方式是针对不同的开发需求提出的,适

收稿日期:2016-10-22

基金项目:国家自然科学基金(61370219)和广东省佛山市创新团队课题(2015IT100095)资助项目.

通信作者:李　廉(1951-),男,山东阳谷人,教授,博士生导师,主要从事机器学习、计算机网络和无线传感器网络等方面的研究. E-mail:llian@hfut.edu.cn

用于各种不同的应用目标. 本文中,提出一种三模块结构的 MDF 模式,在该模式中,架构被分为 3 个模块,即实体模块、消息模块和数据及显示模块. 实体类似于对象,封装了数据和操作,形成独立的运行单元. 但是实体之间没有直接联系,所有的实体都是基于消息控制. MDF 提供一种在网络环境下,共同开发的语言,遵守事先规定好的规范,所有的开发人员完全不需要有沟通联系,根据规范自行开发应用模块(实体),然后加入到系统中运行,并且还可以随时退出,这些操作都无需与系统其它用户和管理人员交互. 所有这些实体(用户)行为决定了系统的行为,个别实体的消亡或变化不影响整个系统的功能. 每个对象的行为具有确定性,但是系统的演化却具有不确定性,这种不确定性是由交互的环境影响而产生的. 在 MDF 中,程序的运行由消息决定,每个实体发生事件将产生相应的消息[6],以此确定了整个系统的行为和功能. MDF 开发方法让软件开发工作变得更简单便捷、灵活、可靠. 特别是在网络众多用户的背景下,使用 MDF 方法开发这类应用提高了软件的开发效率和可维护性,与其类似的开发方法在文献[9-16]中列出.

在软件开发架构中 MDA[8] 是通过各种模型的建立,定义一种全过程的模型控制编程模式,MDA 有一套完整和系统的工程规范. 而本文中的 MDF 只是一种通过消息来驱动对象的程序架构,其内容和规范还在发展之中,这 2 个概念还是有着许多的不同之处.

与已有的事件驱动开发架构(EDA,event-driven architecture)相比[7],2 者具有相似性,理论上 EDA 涵盖了 MDF. 2 者都是通过消息传递信息,但是却有本质的区别. EDA 需要用户有目的性和主动性的去完成一次消息触发. 例如点击一次鼠标、一次键盘输入等,Web 服务属于 B/S 模型,同样是需要用户主动去请求一次操作,后台服务器会根据请求返回相应数据,MDF 则是实体间随机性的消息触发. MDF 可以看作是 EDA 在网络应用环境下的轻量级版本,由于 EDA 的体系比较庞大,因此在一些不需要精确响应和快速协同的应用场合,EDA 显得有些笨拙,而这些场合可以尝试本文提出的 MDF 方法. 例如,对于网络环境下,众多互不相识的用户需要协同完成任务(组织旅游、召开会议、社会选举等),以及模拟一些自然系统行为(生物系统、多体系统等). 其中共同的特点是,一个用户的行为可以受其他用户行为影响,但是这些用户之间却没有任何交互,甚至不知道对方用户的存在,在这些情况下,都可以采取 MDF 来设计软件. 但是 MDF 也有缺点,由于采用了消息交互作为实体的运行触发(trigger),因此对于需要高度时间同步,或者快速时间响应的系统,MDF 并不合适.

## 1　MDF 开发模式

到目前为止,MDF 只是一个概念,其具体结构仍在讨论之中. 当前在软件开发体系中,已经有一些框架形式,比如针对网络服务类应用程序开发的架构有 CORBA 和 ACE 等; 针对一般窗口类应用程序开发的框架有 MFC 和 OpenClass 等; 多媒体应用软件的框架有 Microsoft 的 DirectX 等系列. 这些框架结构各具特点,应用于不同的开发目标.

本文提出一种非常轻型的框架结构,即三模块的 MDF 结构(简称 TMDF,见图 1),将 TMDF 分为 3 个部分: 实体管理模块、消息管理模块和数据显示模块. 这 3 个模块分别管理实体的运行、消息的传递和最终数据的存储和显示. 所有希望加入系统运行的用户,只要根据要求(下面称为规范)宣布自己的实体参数和属性,就可以加入系统,或者宣布退出系统. 这种系统模式适用于具有广泛实体(对象)参与,这些场合包括了很多社会计算系统、非数值计算应用、企业管理与商业应用. 但是对于需要快速计算、图像处理以及频繁信息交互的场合并不适用.

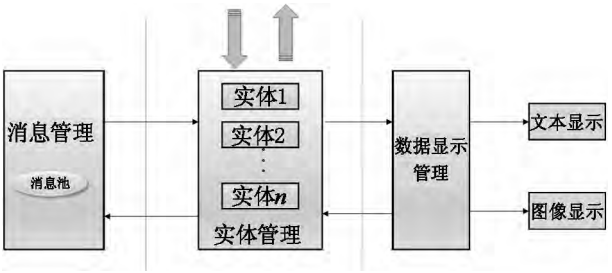


图 1　TMDF 系统架构图

三模块的 TMDF 具有以下几个优点: 1) 去中心化. TMDF 开发没有传统意义上的主程序,因此也没有一个开发中心,只要定义好了系统规范,所有的开发都可以由分散在各地的人员自行开发,甚至支持用户自行开发,用户只需要填写一些信息,就可以获取相应的实体软件,从而加入系统参与运行. 这行模式特别是适合与互联网背景下的众筹式开发. 如大型社区、网络购物等. 2) 开放性. TMDF 的规范是公开的,所有希望使用系统的用户都可以了解其内容,因此软件的开发具有完全的开放性,任何一个实体在软件方面的错误都不会影响其他实体(病毒不在考虑范围内). 3) 轻型化. TMDF 具有非常简单的规范描述,该框架提供了 3 个模块的组件和中间件,在对象关系、通讯联系和网络连接都采用了最基本的

协议形式,从而使得三模块的修改和维护也十分简单,这为轻型快速的应用软件开发提供了便捷的环境. 4) 个性化. TMDF 支持所有的用户根据自己的需要来开发软件,只要符合规范,可以定义对于外部消息的响应,而且这种响应模式外人无法知道,因此对于个性化和私密性具有较好的保护作用.

## 2 实体管理模块

实体是 TMDF 系统的基本概念,像是生命系统里的细胞一样,细胞作为整个生命系统中的实体,是独立运行并且相互之间通过交换信息(也包括物质)实现各自的演化和彼此的合作. 整个生命系统的状况与每一个细胞的表现都有关系,但是任何单独细胞的诞生和死亡却不会影响整个生命系统的存在. 把这种机制引进到软件开发,作为 TMDF 的基本原理. 在 TMDF 中,系统的性能也是由实体决定的,并且实体通过信息交换决定各自的行为. 但是,实体又是独立于系统存在的,实体的加入和退出虽然会影响系统的运行结果,但是不会影响系统的运行.

TMDF 中最重要的是实体管理模块,实体管理模块维护一个实体管理规范(ES,entity specification). ES 描述了系统中实体的属性和行为,实体管理模块负责实体的注册和维护. TMDF 中的实体分为实体类型与实体实例. 实体类型声明了系统中不同的实体形式. 而实体实例是同一类型中不同的实体. 在本文实例的描述太阳与行星之间相互位置变化的系统中,只有一种实体类型(即行星),其中太阳也定义为行星类型,因为在这个系统中,仅仅描述的是位置关系,没有必要区别太阳和行星. 但是这个实体类型有 5 个实体实例,分别是太阳、水星、金星、地球和火星. 而在一个投票系统中,实体类型就会有主持人(负责发布候选人)、票箱(负责计算票数并报告计算结果)和投票人(负责提出候选人和投票),其中主持人实体类型只有一个实体实例,票箱实体类型也只有一个实体实例,但是投票人实体类型可以有任意多的实体实例.

一个系统的实体类型是确定的,不能变动的. 但是实体实例却可以随时变化,加入系统或者退出系统. 所有这些实体实例的运行都服从于实体规范,发布和接受消息,并且根据消息确定动作. 一般地,系统开发者在发布实体规范时,会同时提供各种类型实体的模板,以便于用户开发实体,只要填写一些必要的参数就可以形成实体实例加入系统的运行.

实体是实体规范的主要内容. 实体是独立的、封闭的、可扩展的载体,对外设置有接口,用来发布和接受消息. 图 2 给出了实体与实体、实体与消息管理模块的关系. 图 3 展示了实体生成一条完整消息传输的过程,所有这些都是在高度并发状态下运行的. 实体之间互相没有直接联系,因此一个实体的存在和消失对于其他实体是未知的. 这些措施保证在众多用户情形下,系统运行的可靠性和鲁棒性.

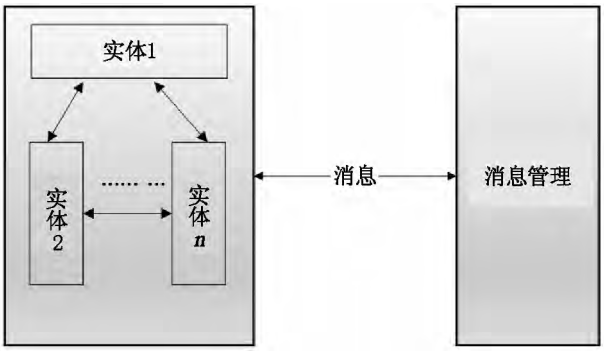


图 2 实体管理模块设计

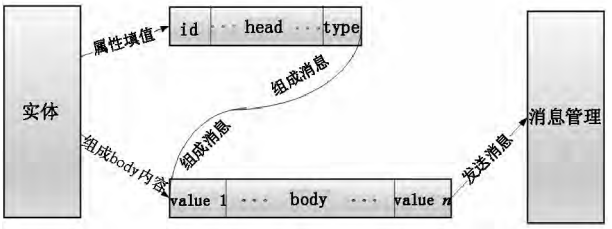


图 3 消息组成与发送

### 2.1 实体规范(ES,entity specification)

实体规范,即对实体的规定和说明,需要遵循一定的格式. 文献[7]提出面向规范的编程方式,将规范直接看作高层次的代码. 用户根据实体规范填写表单,中间件获取填写的信息,生成具体的代码,省去了手写代码的过程. 文献[8]指出,消息驱动的协调程序,其性能超过了在某些情况下传统的运行模式. 在 TMDF 中开发中间件来实现系统对规范的有效识别和维护功能. 实体规范用于系统开发者描述实体格式. 这些描述包括系统说明、实体类型、实体实例(包括可选项实例模板)、实体登录格式、实体消息发送、实体消息接受、消息响应格式、实体输出格式、实体显示模式,所有实体规范采用文本格式描述.

下面逐一介绍相关内容: 1) 系统说明. 说明系统的功能,目标和质量要求. 2) 实体类型. 描述实体的类型,实体类型在整个系统生命期内不能变动; 实体类型允许是空类型,即没有实体实例的类型. 3) 实体实例. 根据实例类型开发实体实例的说明,由系统开发者提供实体模板,以方便用户使用. 有些实体是共享的或者单一的,则由系统开发者开发. 4) 实体登录格式. 定义用户使用实体时,需要在系统上登

录的信息,URL 是必有项,用户登陆后,取得进入系统的许可. 其中用户的 URL 会进入消息管理模块,进入用户名单. 5) 实体消息发送. 描述实体发送消息的内容格式和语义说明. 6) 实体消息接受. 描述实体消息接受的解析. 7) 消息响应格式. 实体在收到消息后必须响应的内容和格式. 不是所有消息都需要响应. 8) 实体输出格式. 描述实体向数据显示模块的输出内容. 9) 实体显示模式,描述实体显示的内容和格式,一般是文本型和图像型,显示软件一般由系统开发者开发.

### 2.2　系统表单

系统表单,即友好的用户界面,系统开发者根据系统规范发布规范. 规范以文本说明和表单的形式给出,其中文本说明描述系统的功能、内容以及必要的定义,以便于用户理解系统. 表单提供用户登录参与系统运行. 表单的内容有以下几部分: 1) 用户注册信息. 用户通过在表单上填写个人信息,实现系统注册,表单会把用户注册信息发送到相应模块,例如用户的 URL 会发送到消息管理模块和数据显示模块,以确认用户的登录. 2) 用户实体实例信息. 用户通过表单声明自己的实体类型,并且提交实体实例软件,这个软件即可以是用户自己开发,也可以借助系统提供的实例模板填写必要参数后使用. 用户实例信息填写的内容在表单中明确规定. 3) 用户数据信息. 用户对于需要永久存储的数据进行声明,向数据显示模块进行登录. 系统公共数据部分由系统开发者定义,不能由用户改动. 4) 用户显示信息. 用户对于需要显示的数据和形式进行声明,由数据显示模块响应.

### 2.3　实体动作定义

实体动作是系统中最重要的部分,它是整个软件的基本内容,在 TMDF 中,系统开发者可以使用文本形式描述实体的动作,动作采取函数式语言描述,每一个动作的描述包括: 触发动作的消息内容、动作的输出数据及处理、动作结束后发出的消息内容.

动作通过接受消息触发,结束后发出消息,因此在一个动作执行期间,不再接受任何消息. 由于实体可能是支持并行操作,因此一个动作执行期间,其他部分可以继续接收消息,并同时触发其他动作.

TMDF 与传统程序编制的区别是,系统本身并不规定具体的动作实现和程序,所有参与开发人员只是根据规范提供符合要求的消息送到消息管理模块,以便于其他用户使用. 同时也从消息管理模块接受消息,运行自己的实体,至于实体内部是如何进行运算的操作的,TMDF 并不规定. 这部分的内容或者由系统提供的实体模板实现,或者由用户自己实现. 也就是说,仅就 TMDF 本身而言,只关心消息之间的响应关系,而不关心如何实现这些响应. 这就给予了用户较大的自由发挥空间. 因此这种模式支持网络环境下众多用户共同维护系统的模式. 每一个用户开发的程序是“私有的”,因此整个系统只能看到“自己”程序的运行,看不到“其他”程序的运行.

但是从实际应用,强烈建议系统开发者提供一个实体模板,这样用户可以免于编制软件的烦恼,这对于只是希望使用系统的用户来说是方便的. 用户通过填写必要的参数来把模板具体化为实体实例.

## 3　开发实例

作为一个表现 TMDF 特点的实例应用,本文开发了太阳系的行星(包括小行星)系统,该系统具有演化性、独立性和相互影响的特点. 某一行星某一时刻的运动轨迹(位置、速度)与当前时刻位置和加速度有关,同时也受到其他行星的引力影响,引力的大小和方向时刻变动,这正是 TMDF 擅长处理的问题.

行星运行受到万有引力的作用,引力大小由星体质量和距离决定. 太阳系中行星和小行星数量众多,所以受到来自各个方向的引力作用,图 4 和图 5 说明了地球的受力分析.

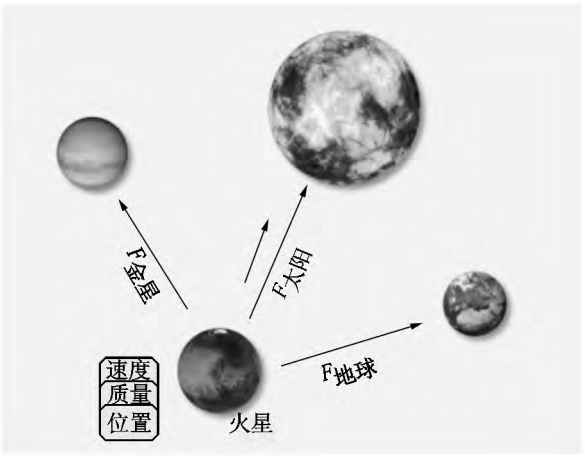


图 4　火星引力分析

文献[9]采用一种逼近算法计算出各行星公转过程中单位时间内的偏角、顶点坐标,达到了精确表示各行星轨道及公转的仿真效果. 本次实验旨在验证 TMDF 合理性和效率,所以实验比较简单. 实验中根据万有引力来确定行星的运动轨迹,行星在注册表单中需要填写初速度和自身质量,时间间隔 $t$ 为默认值. 万有引力公式为 $F = Gm_1m_2/r^2$,其中 $m$ 为行星质量,$r$ 为距离,$G$ 为引力常量. 经查阅资料,表 1 中给出了行星参数.

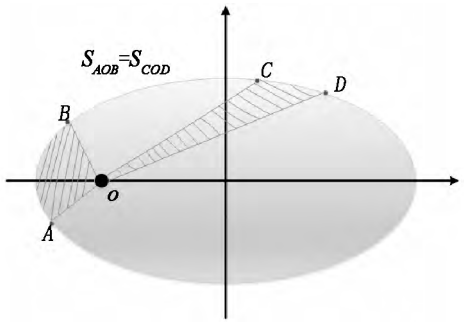


图5 开普勒第2定律

表1 行星参数

| 行星 | 质量/kg | 位置 $\{(x,y)\}$/m | 速度 $\{(x,y)/(\mathrm{m}\cdot\mathrm{s}^{-1})\}$ |
|---|---|---|---|
| 金星 | $4.869\times10^{24}$ | $(0,1.07\times10^{11})$ | $(-35\ 000,0)$ |
| 地球 | $5.980\times10^{24}$ | $(1.496\times10^{11},0)$ | $(0,29\ 783)$ |
| 火星 | $6.418\times10^{23}$ | $(-2.496\times10^{11},0)$ | $(0,-24\ 000)$ |

在确定了引力、速度和时间后就可以确定下一时刻行星的位置. 实体将计算出的位置放入 body 部分,然后和 head 组成一条完整消息发送出去,这样,一次完整的通信就完成了. 作为演示的例子,本文只选取了太阳、地球、金星、火星这 4 个星体组成的系统,本文开发的例子可以随时添加任意的星体,也可以随时删除任意的星体.

该系统的一些基本规范内容如下: 1) 系统说明. 行星运行系统,每个行星提交即时的位置(空间) 3 维坐标,速度(3 维矢量),质量,时刻数据; 系统模拟该行星系统整体运行情况; 2) 实体类型. 一类,即行星(包括太阳); 3) 实体实例. 本规范提供实体样例,用户根据界面窗口填写具体的参数加入系统运行; 4) 实体登录格式. 见窗口; 5) 实体消息发送. 发送消息包括位置(空间) 3 维坐标、速度(3 维矢量)、质量、时刻. 数字类型,8 个数据顺序排列,逗号隔开,前 7 个为浮点数,第 8 个为时间格式,广播形式发送; 6) 实体消息接受. 接收消息,根据引力公式计算行星下一时刻的位置,时刻值可变,默认为 600 s(10 min); 7) 消息响应格式. 不需要响应; 8) 实体输出格式. 输出每一时刻的位置数据到数据库; 9) 实体显示模式: 图像显示.

实体管理模块中设置了系统管理员 admin,作为对于整个系统的管理. 如改变时刻值、改变计算规则等. 系统管理员通过注册获取管理权限.

普通实体用户在注册时按照表单中要求进行填写,在本系统中,需要用户填写一些必要的信息,例如用户名、密码、URL 以及实体的一些初始信息. 注册完成后中间件会自动生成实体实例,并且向系统管理模块发送必要的消息,进入系统参加运行. 用户可以随时通过该界面添加新的实体(相当于发现新的星体),或删去星体(相当于该星体已经消失),如图 6 所示.

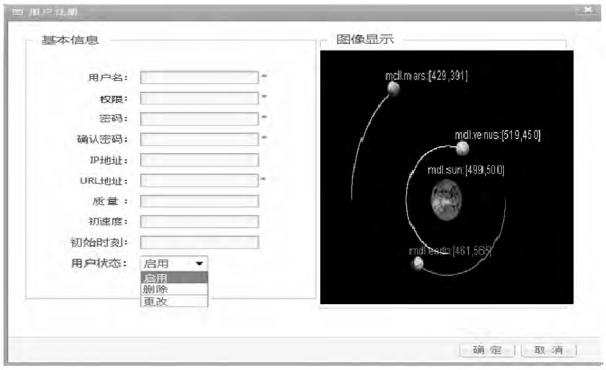


图6 实体注册表单

## 4 结束语

在互联网时代,不仅需要重量级的软件开发框架,如 J2EE、MFC、CORBA 等涵盖大型应用的架构,也需要提供轻量级的用于众多小型和分散应用的软件开发架构. 这种要求在许多新兴应用中具有很大的市场. TMDF 带来了一些新的架构特点,如易于并行开发、便于维护、系统支持演化特性和灵活性等,提供了便捷和快速的开发形式. 在互联网一些特定的应用中,具有一定的优势. 作为一种较新的开发概念,还有许多问题需要解决,特别是对于 TMDF 的结构、实体描述规范语言、消息传递机制、支持大流量消息的策略,以及相应的开发工具都需要继续大量的研究这些方面还期待于未来的发展.

## 5 参考文献

[1] Bainomugisha E, Van Cutsem T, Mostinckx S, et al. A survey on reactive rogramming [J]. Acm Computing Surveys, 2013, 45(4): 1-34.

[2] 李玉龙,李长云. 软件动态演化技术 [J]. 计算机技术与发展,2008,18(9): 83-86.

[3] De Paoli F, Tisato F. On the complementary nature of event-driven and time-driven models [J]. Control Engineering Practice, 1996, 4(6): 847-854.

[4] 惠长江,吴江. 软件进化研究综述 [J]. 计算机技术与发展,2007,17(4): 196-200.

[5] 郭先超,林宗缪,姚文勇. 基于规则库与消息驱动机制的工作流 [J]. 计算机应用,2014,34(1): 270-272.

[6] Singh K, Both A, Diefenbach D, et al. Towards a message-

driven vocabulary for promoting the interoperability of question answering systems [EB/OL]. [2016-07-11]. http://eis.iai.uni-bonn.de/upload/paper/2015_Towards_a_Message_Driven_Vocabulary_for_Promoting_the_Interoperability_of_Question_Answering_Systems.pdf.

[7] 杨志义,杨刚,张海辉. 一种面向服务的事件驱动架构信息集成平台构造方法 [J]. 计算机研究与发展,2008,45(10):1799-1806.

[8] Object Management Group. MDA: the architecture of choice for a changing world [EB/OL]. [2016-07-05]. http://www.omg.org/mda/.

[9] 张天,张岩,于笑丰,等. 基于 MDA 的设计模式建模与模型转换 [J]. 软件学报,2008,19(9):2203-2217.

[10] 于俊清,余华飞,魏海涛,等. 多核环境下编译器辅助消息驱动的动态调度 [J]. 计算机学报,2014,37(7):1633-1637.

[11] 何成万,何克清. 基于角色的设计模式建模和实现方法 [J]. 软件学报,2006,17(4):658-669.

[12] Kunzman D. Runtime support for object-based message-driven parallel applications on heterogeneous clusters [EB/OL]. [2012-10-12]. http://charm.cs.illinois.edu/newPapers/12-45/paper.pdf.

[13] 董红斌,黄厚宽,印桂生,等. 协同演化算法研究进展 [J]. 计算机研究与发展,2008,45(3):454-463.

[14] 曾一,周吉,孙政,等. 支持 MDA 的设计模式建模与模型转换方法研究 [J]. 计算机工程与应用,2012,48(1):76-80.

[15] Vasudevan R,Vadhiyar S S,Kal,et al. G-Charm: an adaptive runtime system for message-driven parallel applications on hybrid systems [C]. International ACM Conference on International Conference on Supercomputing,2013:349-358.

[16] Phillips J C,Stone J E,Schulten K. Adapting a message-driven parallel application to GPU-accelerated clusters [EB/OL]. [2016-05-11]. http://130.126.143.33/sites/default/files/papers/393/SC08_NAMD.pdf.

[17] Kantorowitz E,Tadmor S. Toward specification-oriented frameworks [EB/OL]. [2016-10-12]. http://www.research.ibm.com/haifa/info/ple/papers/TowardSpec-Subm.pdf.

[18] Penczek F,Herhut S,Scholz S B,et al. Message driven programming with s-net: Methodology and performance [EB/OL]. [2016-10-12]. https://staff.fnwi.uva.nl/c.u.grelck/publications/PencHerhScho+P2S210.pdf.

[19] 李万万. 基于 Java3D 的太阳系仿真的原理与实现 [J]. 哈尔滨师范大学学报: 自然科学版,2013,29(6):53-56.

# The Design and Implementation of Entity Management Module Based on Message-Driven Architecture

WANG Yongjin[1],LI Lian[2*]

(1. Institute of Industry & Equipment Technology,Hefei University of Technology,Hefei Anhui 230009,China;
2. School of Computer and Information,Hefei University of Technology,Hefei Anhui 230009,China)

**Abstract:** A message-driven Framework(MDF) for software development method is explored,which tends to improve the efficiency of software development in the application system on internet. MDF is divided into three modules,which are,respectively,entity management(EM),message management(MM) and data display management(DM). The mainly studies is the entity management module. In EM,users need only enrolling some information,then EM will create a real entity automatically according to the information from entities sample. From the point of view of design pattern and development method,MDF has more convenient and higher efficiency. At the same time,some corresponding middleware are developed to support the operation of the system,and a practice case is used to illustrate the characteristics of this method.

**Key words:** message; message-driven; multi-access application system; internet use; software development method

(责任编辑: 冉小晓)