[文章编号] 1003－4684(2019)01-0069-04

AOP 语言在 GUI 编程中的应用

徐承志，黄振兴
（湖北工业大学 计算机学院，湖北 武汉 430068）

[摘 要] 面向 Agent 编程是一种更高抽象级别的编程范型，常见于人工智能、交互式仿真等研究领域，却很少应用于工程实践领域，其中一个原因是缺少对图形用户接口的支持。为了解决这个问题，一种命令式的编程语言－－CAOPLE，将底层图形库封装成 Agent 组件，并在编程过程中遵循三层结构的设计，即对外接口层、枢纽层和表示层，以适应面向 Agent 编程的特点。将这种编程模式应用于网络聊天室的设计中，用清晰的逻辑和简洁的代码实现了在分布式环境下的图形交互应用。CAOPLE 的图形接口编程证明了，面向 Agent 编程思想在工程应用领域的可行性和适应性，并将推动下一代面向 Agent 的软件工程早日进入实用阶段。

[关键词] Agent；面向 Agent 编程；图形用户接口；GUI 编程
[中图分类号] TP311 [文献标识码] A

继云计算、物联网和移动互联网之后，大数据也成为信息技术计算机产业发展的新方向.面对如此庞大的市场和复杂的需求，工业界迫切地希望寻求软件的新概念、新模型和新技术来支持这类新系统的工程化开发。作为早期人工智能领域内的一个关注点，智能体(Agent)，由于其的固有特征符合目前软件发展趋势，因此被成功引入软件工程领域.面向 Agent 编程(AOP)的思想也迅速成为研究热点，并成为大规模网络计算的一种可行的解决方案。

1 AOP 发展现状

近年来人们已经提出了不少面向 Agent 的建模工具、编程语言和运行环境。但是学术界、工业界呈现出冷热不均的势态。目前工业界采用的实现手段主要还是面向对象编程(OOP)，以及围绕 OOP 形成的软件理论和编程框架[1]，而真正采用面向 Agent 编程而实现的工业产品并不多。出现该问题的原因，很大一部分是因为研究者把研究的精力过于投入到 Agent 系统的智能性上去了，而忽略了 Agent 开发中实际体现出来的代码问题和工程问题，使得面向 Agent 的软件开发缺少必要的工程标准和实践理论支持。

自从 Agent-0 被首次提出，已经涌现了大量的 AOP 语言，文献[2-3]将其分为三大类，分别是声明式语言、命令式语言和混合型语言。其中，声明式语言占了大多数，大多数以 Lisp 式的函数语言或 Prolog 式的逻辑语言形式存在。这些语言在人工智能领域内非常成功，但并不适用于工程领域。原因除了执行效率有待提高外，声明式语言更适用于逻辑推理，而并不适合表达实际应用中的业务流程、UI 交互、批量数据处理等。因此，在进行工程类软件开发时，采用命令式语言更为合理。CAOPLE 就是命令式 AOP 语言的典型代表，该语言实现了对图形用户接口(GUI)的封装，使其更适合开发应用型项目。

2 CAOPLE 简介

Caste-centric Agent-Oriented Programming Language(CAOPLE)是牛津布鲁克斯大学的形式化应用研究组提出的一种命令式的 AOP 编程语言，并经历了从形式化[4]、建模[5]、编程环境[6]，到编译运行时环境[7]等多个阶段的发展[8]，逐渐成为具有实用性的开发工具。CAOPLE 将 Agent 之间的交互行为(进程间通信、网络通信)直接映射到语言的基础设施中，从用户角度来看，网络是透明、环境是透明的、交互也是透明的。CAOPLE 语言符合

[收稿日期] 2018－09－03
[基金项目] 湖北省教育厅科学技术研究计划项目(B2017048)；欧盟 FP7 项目(PIRSES－GA－2011－295222)
[第一作者] 徐承志(1976－)，男，湖北武汉人，工学博士，湖北工业大学副教授，研究方向为多 Agent 系统，智能计算
[通信作者] 黄振兴(1982－)，男，湖北黄梅人，湖北工业大学硕士研究生，研究方向为信息技术应用

Agent 的弱定义，即自治性、反应性、能动性和社会性，而 Agent 的强定义特性，如知识、信念、愿望、意图等，则可以通过上层代码来实现。这样既能满足 Agent 基本特性，又可以兼容其它各类 Agent 模型。

3 CAOPLE 的 GUI 编程

支持图形用户接口(GUI)是高级语言进入实用阶段的重要标志。只有具备 GUI 编程的能力，AOP 语言才能开发具有图形化界面的应用软件，而对于大多数的普通用户来说，图形化的应用软件才是最易接受和最易使用的软件。

面向 Agent 编程的思想是将程序中的所有基本单元都视为 Agent 的实例，运用到图形化软件界面中，所有可视化控件也可以被视为 Agent。传统的面向对象编程思想在实现控件交互时，控件对象的成员函数(Method)被动地被外界调用；而 AOP 的思想则是要求，控件实例自主采取行动(Action)。Agent 的独立性使得控件不受外界对象的操控，控件之间的交互完全依靠与环境进行信息交换实现。CAOPLE 语言在 GUI 编程方面做了有益的尝试，可以对常见的控件进行 Agent 级别的封装，使之具备 Agent 的基本特性，即自治性、主动性、反应性和社会性。下面将围绕一个具体的应用(网络聊天室)来阐述 CAOPLE 的 GUI 编程解决方案。

3.1 聊天室模型

为了更清晰的说明 GUI 编程的解决方案，在设计聊天室时采用了极简设计。整个软件只包含 2 个非可视化元素和 4 个可视化元素(表 1)。

其中，Chatter 不是 GUI 控件，它是聊天用户在网络空间中的软件代理。用户上线后，系统会为每一位聊天者在网络空间里驻留一个 Chatter 实例，该实例的主要作用就是向外部环境(由网络中所有的 Chatter 共同构成)输出自己的发言，同时也从外部环境中获取别人(其它 Chatter)的发言。同时，Chatter 的另一个作用，就是创建一个 GuiController 实例，并与之建立起 1∶1 的联系。

表 1 聊天室中的 Agent

	Agent 名称	Agent 描述	封装 GUI
1	Chatter	聊天者代理	否
2	GuiController	GUI 构建者	否
3	Frame	UI 面板	是
4	TextArea	内容显示区	是
5	TextField	发言输入框	是
6	Button	发送按钮	是

GuiController 也不是 GUI 控件，它的作用是创建聊天室的 GUI 控件组，并作为交互中心建立起 GUI 控件与 Chatter 之间的联系。所有的控件实例都将 GuiController 视为消息的起点或终点，而控件实例之间没有额外的通信。这样的设计简化了交互方式，代码的逻辑结构也相对清晰。

图 1 是聊天室的 UI 设计图，显示了所有 Agent 封装过的 GUI 控件组合。其中，Frame 是对话框面板，它的作用是整个 UI 界面的静态容器。TextArea 显示了环境中所有 Chatter 的发言。TextField 则是当前用户的输入区。Button 的点击动作则是响应用户的发言操作。

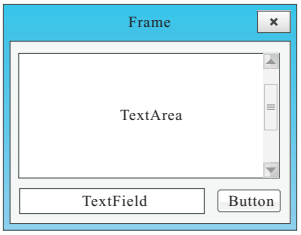

图 1 聊天室界面布局

3.2 多 Agent 系统的交互

本文中的聊天室是一个纯 Agent 系统，其中的每一个对象都被封装为一个 Agent 实例。实例的创建顺序如图 2 中的实线箭头所示。首先被创建的是 Chatter，然后由 Chatter 创建 GuiController，最后由 GuiController 创建所有的 GUI 控件，Frame、TextArea、TextField 和 Button。

由于每个 Agent 实例都要保持自主性，实例之间不能直接调用对方的代码，而实例之间的交互是依靠消息传递实现的，所以该 Agent 系统是基于消息驱动的。图 2 中的圆角矩形框将整个系统分为内部环境与外部环境。对于某个聊天客户端来说，其所属的所有 Agent 实例都在本机运行，他们共同构成了内部环境，而该客户端之外的所有实例(其它的聊天客户端)共同构成了外部环境。GUI 控件的消息只在本机的内部环境中传递，不必经过网卡进入外部环境，而该客户端只与外部环境进行必要的消息传递。

消息时序流程如图 2 中虚线箭头所示。当用户点击 Button 后就会向内部环境发送①click 消息。GuiController 接收到该消息后，分别发送了②enter 和③clearField 两个消息。enter 消息被 Chatter 接收到，并触发其向外部环境发出携带着发言内容的④speak 消息，即将发言内容通知其它 Chatter。同时，clearField 消息被 TextField 接收到，并触发了清除 TextField 的行为。接着，Chatter 也会接收来自外部环境中其它 Chatter 的⑤speak 消息，在获取别人的发言内容后，再向内部环境发送⑥updateGUI 的消息，GuiController 获取该消息后随即向

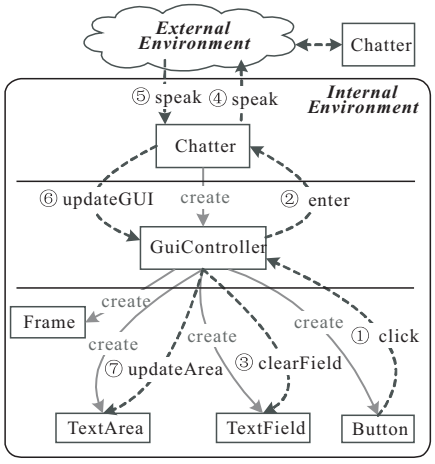

图 2　多 Agent 交互协作图

TextArea 发送⑦updateArea 消息。TextArea 接收到该消息后,即刻更新聊天记录。至此,一次聊天记录的发收循环结束。

从图 2 可以看出,聊天客户端里的 Agent 被分为了三个层次。中间为枢纽层,由 GuiController 构成,顶层和底层都需要通过它进行消息传递。顶层是对外接口层,由 Chatter 构成,它专门负责与外部环境进行信息交换。底层是表示层,由各个 GUI 控件构成,主要负责界面显示。这种分层设计职责明确、分工合理、内外分开,可以推广到其它的 GUI 程序设计中。

3.3　Agent 封装

每个 Agent 的代码被封装到以 cpl 为扩展名的源文件中,下面将展示聊天室中各 Agent 的主要代码。其中,注释中的序号代表从用户点击按钮开始,到发言显示到窗口截止,整个过程中消息产生的时序。

```
Chatter.cpl
caste Chatter(…)
{ observe all chatter in Chatter;
    observe gc in GuiController;
    …
action speak(str: string){ … }
action updateGUI(str: string){ … }
init
{ create gc of GuiController(…); }
body
{ when gc:enter(rcv sentence)
   { speak(sentence); } //④
    when exist chatter:speak(rcv sentence) //⑤
    { updateGUI(sentence); } //⑥
}
}
GuiController.cpl
caste GuiController (…)
{ observe ct in Chatter;
observe bt in Button;
…
var myFrame: Frame;
var myTextArea: TextArea;
var myTextField: TextField;
var myButton: Button;
action enter(str: string){ … }
action updateArea(str: string){ … }
action clearField(){ … }
init
{ create myFrame of Frame(…);
  create myTextArea of TextArea(…);
  create myTextField of TextField(…);
   create myButton of Button(..);
   …
}
body
   { when bt:click( )
  { content:=myTextField.text;
     enter(content); //②
     clearField( ); //③
     }
     when ct:showSentenceInGUI(rcv content)
      { updateArea(content); } //⑦
  }
}
Frame.cpl
caste Frame(…)
{ init{ … }
   body{ … }
}
  TextArea.cpl
caste TextArea(..)
{ init{ … }
   Body { … }
}
TextField.cpl
caste TextField(…)
{ init{ … }
   body { … }
}
  Button.cpl
caste Button(…)
{ action click( ){ … }
  init{ … }
  body
  { when self:clickEvent()
     { click( ); } //①
}
}
```

从上述代码可以看出,程序与外界交互的部分被封装到接口层里的 Chatter 中,软件内部的 Agent 交互被封装到枢纽层里的 GuiController 中,软件的可视化部分被封装到显示层的 GUI 控件中。

3.4　实例演示

CAOPLE 语言支持基于 CLR 技术的.Net 运行时环境和基于 JVM 技术的 Java 运行时环境。本文的网络聊天室将演示运行在 Java 虚拟机上的效果(图 3)。

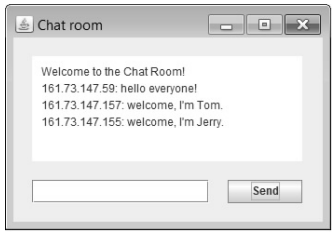

图 3　系统运行截图

CAOPLE 源文件被编译后,生成扩展名为 cst 的目标代码文件。目标代码文件可以提前部署或自动分发到网络中的不同虚拟机上。同时,CAOPLE

不仅支持本地创建实例,还支持远程创建实例,下面的启动代码展示了远程启动效果。第一个 Chatter 实例被创建到本地(IP 为 161.73.147.59),而第二、第三个 Chatter 实例则被创建到远端(IP 分别为 161.73.147.155 和 161.73.147.157)。

```
caste BootChatter( )
{ init
{ create Chatter( ); // @ "localhost"
create Chatter( ) @ "161.73.147.155";
create Chatter( ) @ "161.73.147.157";
}
body{ }
}
```

图 3 展示了实际运行的效果。在网络不同节点上,客户端都显示了相同效果,不同客户端的发言都显示到各自的 GUI 界面上。

4 总结

本文采用面向 Agent 编程的 CAOPLE 语言,实现了网络聊天室的 GUI 设计,证实了 AOP 技术有能力进行常规应用型软件的开发。同时,针对 GUI 类型的软件,提出了两个环境、三层架构的设计思想。两个环境是指内部环境与外部环境,三层架构是指对外接口层、交互枢纽层和界面显示层。两个环境提高了通信的效率,三层架构简化了代码的逻辑。这套设计思想可以有效的移植到其它面向 Agent 编程的 GUI 软件开发中。随着 Agent 技术应用在更多的实用领域,AOP 编程必将推动下一代面向 Agent 的软件工程早日来临。

[参 考 文 献]

[1] Vakilian M , Chen N , Zilouchian Moghaddam R , et al. A compositional paradigm of automating refactorings[C]// European Conference on Object-oriented Programming. Springer Berlin Heidelberg, 2013. Bordini R H, Braubach L, Dastani M, et al. A Survey of Programming Languages and Platforms for Multi-Agent Systems[J]. Informatica, 2006, 30(1):33-44.

[2] Bădică C, Budimac Z, Burkhard H D, et al. Software agents: Languages, tools, platforms[J]. Computer Science & Information Systems, 2011, 8(8):255-298.

[3] Wang J , Shen R , Zhu H . Agent oriented programming based on SLABS[C]// International Computer Software & Applications Conference. IEEE, 2005.

[4] Zhu H , Shan L . Caste-centric Modelling of Multi-agent Systems: The CAMLE Modelling Language and Automated Tools[M]. Springer Berlin Heidelberg, 2005.

[5] Zhou B, Zhu H. A Virtual Machine for Distributed Agent-oriented Programming[C]// Twentieth International Conference on Software Engineering & Knowledge Engi-neering. DBLP, 2008:729-734.

[6] Xu C, Zhu H, Bayley I, et al. CAOPLE: A Programming Language for Microservices SaaS[C]// Service-Oriented System Engineering. IEEE, 2016:34-43.

[7] Liu D, Zhu H, Xu C, et al. CIDE: An Integrated Development Environment for Microservices[C]// IEEE International Conference on Services Computing. IEEE, 2016:808-812.

Application of AOP Language in GUI Programming

XU Chengzhi, HUANG Zhenxing

(*School of Computer Science, Hubei Univ. of Tech., Wuhan* 430068, *China*)

Abstract: Agent-oriented programming is a higher level of abstraction programming paradigm, commonly found in artificial intelligence, interactive simulation and other research areas, but rarely used in engineering practice. One of the reasons is the lack of support for graphical user interface. To solve this problem, we propose an imperative programming language, CAOPLE, which encapsulates the underlying graphics library into Agent components, and follows the design of three-layer structure in the programming process, namely, external interface layer, hub layer and presentation layer, in order to adapt to the characteristics of Agent-oriented programming. Applying this programming mode to the design of web chat room, the graphical interactive application in distributed environment is realized with clear logic and simple code. CAOPLE's graphical interface programming proves the feasibility and adaptability of Agent-oriented programming ideas in engineering applications, and will promote the next generation of Agent-oriented software engineering to enter the practical stage as soon as possible.

Keywords: Agent; Agent-oriented programming; graphical user interface; GUI programming

[责任编校: 张岩芳]