

# 数字化医疗设备软件开发平台的设计和实现<sup>①</sup>

## Design and Implementation of Software Development Platform Oriented Towards Digital Medical Equipment

王晓楠 (江苏常熟理工学院软件工程系 215500)

**摘要:** 本文根据嵌入式软件的自身特点,从体系结构和框架结构的角度分析并提出了数字化医疗设备软件开发平台的实施方案,并且成功地验证了该方案的可行性和正确性

**关键词:** 嵌入式软件 框架 通信

### 1 引言

近年来,随着计算机技术、网络通信技术和软件开发技术的快速发展,基于嵌入式软件技术的数字化医疗设备在全球范围内得到了突飞猛进的发展。面向数字化医疗设备的嵌入式软件就是指计算机技术与医疗的硬件设备结合起来共同完成设备信息处理和智能控制要求的智能控制软件。它具有与底层硬件结合紧密、要求严格的实时性和极高的可靠性等特点。目前,国内在大型数字化医疗设备的软件研发方面还缺乏相关的标准和通用的平台。所以,迫切需要一个面向数字化医疗设备的软件开发平台,使得数字化医疗设备软件的开发能够规范化、高效化。

### 2 技术路线

由于数字化医疗设备软件开发平台是针对多种医疗硬件设备的软件开发平台,而不同医疗硬件设备的软件开发可能要求不同的软件和硬件环境,所以,我们必须选择一个标准框架,使这些使用不同语言、操作系统、和硬件开发出来的数字医疗设备的应用系统能够实现对象资源共享、代码重用以及对象之间的相互访问。因此,我们采用了基于公共对象请求代理体系结构(CORBA)的分布式 Agent 通信机制和“软总线+软构件”的体系结构。其中,软总线是以分布式对象技术为基础的,它不仅能够支持应用集成框架的建立,而且还能保证该软总线上的软构件之间的协调工作。在 CORBA 中,CORBA/ORB 相当于一根软总线,而软构件可以在这根软总线上即插即用。

### 3 框架结构

对于本系统来说,框架结构就是未完成的 Agent 结构或者是未完成的由多个 Agent 组成的分布式系统的结构。这样,本系统不仅要提供一组构造 Agent 的基本单元,而且还

要提供将其基本单元组合成目标系统的手段。

由于面向对象技术所具有的对象封装特性和可扩展特性,采用基于面向对象技术的建模方案。按照全面讨论的 Agent 的概念模型,本系统把 Agent 的定义分解为事件感知对象的定义和事件处理对象的定义,它们分别对应 Agent 的事件感知行为的定义和 Agent 的事件处理行为的定义。事件感知对象包含一些事件感知方法,它们的功能是用来探测是否有事件发生;而事件处理对象包含一些事件处理方法,它们的功能是处理由事件感知对象探测到的所发生的事件。在本系统中,当事件感知对象探测到有事件发生后,它会启动相应的事件处理对象的事件处理方法来处理该事件。

由于在本系统中 Agent 的事件发生与事件处理相对分离,所以我们构建了如下的多层次框架结构,如下图 1 所示:

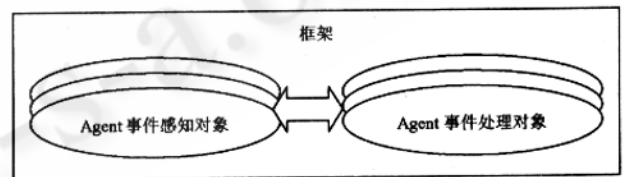


图 1

第一层次的 Agent 框架除了主控流程外,其余部分(即事件感知对象和事件处理对象)均由开发人员设计实现;第二层次的 Agent 框架和第一层次的 Agent 框架相比,进一步承诺了(部分)事件感知对象,但是事件处理对象仍然由开发人员设计完成;第三层次的 Agent 框架较之第二层次的 Agent 框架,更进一步承诺了(部分)事件处理对象。

根据这个多层次框架,我们可以针对不同的医疗设备进一步发展下去,一般情况下,Agent 的层次越高,其面向的医疗硬件设备越具体,应用开发者的开发工作越简单。

我们可以根据不同的医疗设备,建立多层框架结构:当事件感知对象探测到某个事件发生时,它会启动相应的事件处理对象来处理这个事件,同样,这个事件处理对象可以采

<sup>①</sup> 此项目为 863 基金项目,项目号是:2003AA1Z2200

用同样的 Agent 框架,即发起一个事件,当事件感知对象探测该事件后,再调用相应的事件处理对象来处理,周而复始,这样就可以建立起多层的框架结构。此处,针对不同的医疗设备所确定的框架结构的最终层数是不同的。

通过以上描述可知,本系统的框架结构自身具有可扩展性,既可在其中增加新的结构,也可提高已有结构的层次和专用性,该框架也支持面向应用的开发,即应用开发者只需在适当的框架中加入具体的应用成份即可完成开发任务。

#### 4 Agent 通信

Agent 间的协作建立在相互间的通信基础上,也就是 Agent 必须使用相互可理解的通信语言 KQML (Knowledge Query and Manipulation Language, 知识查询及操作语言) 来进行通信。本系统把 CORBA 和 KQML 相结合,构造了 Agent 通信框架,如下图 2 所示:

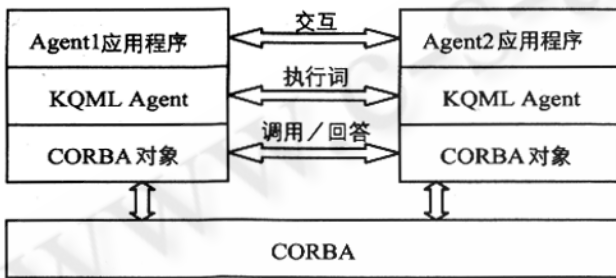


图 2

通过将 CORBA 和 KQML 相结合,我们把整个 Agent 通信框架分为两个层次的集成:

第一层次的集成是从 Agent 层到 KQML 层,此处,用 KQML 文本形式表达协作 Agent 之间的交互,这需要一个 KQML 语句解析/生成器。解析器将另一个 Agent 发来的消息截获,并将 KQML 转化为实际信息;生成器将 Agent 的信息需求转化为 KQML 文本。

第二层次的集成就是从 KQML 层到 CORBA 层,在 CORBA 层中需要 IDL 语言定义所有 Agent 可交换的消息。当 Agent 的信息请求转化为 KQML 后,由 CORBA 层中的相关程序对 KQML Agent 进行编码,产生 IDL 描述的服务请求;当 Agent 的处理结果返回时,由 CORBA 层中的相关程序将处理结果进行解码,把 IDL 转化为 KQML。

#### 5 通信流程

事件感知 Agent 对象包含一些感知方法。事件感知方法的一般流程为:

- (1) 首先探测是否有事件发生;
- (2) 如果有事件发生,它会启动相应的事件处理 Agent 对象来处理该事件;

(3) 转(1)。

这里,事件感知 Agent 对象的事件探测方法一旦被调用,便进入周而复始的事件探测活动中。因为 Agent 事件发生与事件处理相对分离以及框架结构的分层特点,在事件感知对象中,定义了事件处理对象注册的方法,其功能是将处理某事件的对象注册到该事件的感知对象中。

下面举例说明整个平台如何运作,假设事件感知 Agent 对象已经探测到一个事件已经发生,那么,它将按照下面的步骤处理这个事件(如下图 3 所示):

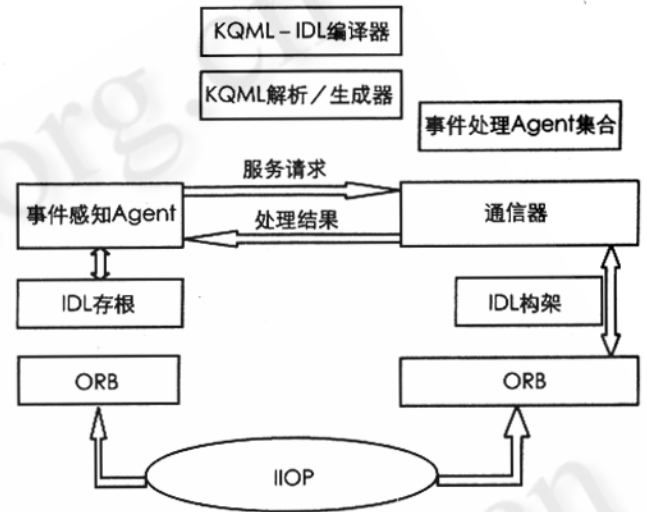


图 3

① 事件感知 Agent 对象根据探测到的所发生的事件的相关信息,提出启动相应事件处理 Agent 对象的处理方法的信息请求;

② KQML 解析/生成器将该信息请求转化为 KQML,然后,IDL 存根程序对 KQML 进行编码,产生 IDL 描述的服务请求;

③ 事件感知 Agent 对象将服务请求发送给通信器;

④ 通信器对接收到的信息进行过滤,并将请求转发到提供服务的相应事件处理 Agent 对象;

⑤ IDL 构架对 DL 描述的信息服务请求进行解码,恢复原来的 KQML,然后,KQML 解析/生成器对 KQML 再解析,恢复原来的 Agent 信息请求,提供服务的事件处理 Agent 对象对该信息请求进行处理;

⑥ IDL 构架对处理结果或者异常信息进行编码,生成 IDL 描述的处理结果;

⑦ 提供服务的事件处理 Agent 对象将处理结果返回给通信器,通信器将结果返回给提出服务请求的感知事件 Agent 对象;

⑧ IDL 存根将处理结果进行解码,转化为 KQML,然

(下转第 57 页)

后, KQML 生成/解析器对 KQML 进行解析, 将 KQML 转化为实际信息。

在⑤中, 提供服务的事件处理 Agent 对象在处理事件感知 Agent 对象提出的服务请求时, 它可能要进一步发起一个事件, 这样某个事件感知 Agent 对象会又一次重复①-④步骤, 这样不断循环, 直到处理完该服务请求为止。

## 6 结束语

分布式对象技术已成为建立应用系统集成框架和标准构件的核心技术。数字化医疗设备软件开发平台遵循 CORBA 标准, 为软件开发人员提供了一个面向医疗硬件设备的软件开发及系统集成平台。

本平台是在网络基础设施上建立的支持集成化的分布式医疗系统开发平台, 通过本平台, 面向不同医疗设备的医

疗系统可以在分步异构的环境上移植、重用和互操作, 同时协同工作, 适应企业医疗设备软件开发的应用需要。

### 参考文献

- 1 Lewie T G. Where is client/server software headed [J]. Computer, 1995, 28(4): 49-55.
- 2 Tim Finin. Don McKay, Rich Fritzson et al. KQML: an information and knowledge exchange protocol [C]. In: Kazuhiro Fuchi, Toshio Yokoi, eds. Knowledge Building and Knowledge Sharing. Ohmsha and IOS Press, 1994.
- 3 Object Management Group. The Common Object Request Broker. Architecture and specification [S]. Revision 2.2 February 1998.