具有监控能力的 Agent 模型*)

赵新宇　林作铨
(北京大学信息科学系　北京 100871)

摘　要　在大信息量、动态环境下企业软件系统对业务流程的监控和自动管理的需求越来越大。本文把 Agent 技术引入业务流程的监控和管理，在基本的 BDI 模型基础上加入能力组件的概念，提出一种具有监控能力的 BDIC Agent 模型，以具体应用背景中的实例说明了该模型中监控能力的分类。该模型已在一个客户服务系统(CSM)中成功应用，说明所提出的 Agent 模型具有良好的可重用性和可定制性，适合对企业软件系统中的业务流程进行监控和管理。

关键词　Agent，BDI，监控，能力，业务流程

A Model of Agent with Monitoring Capabilities

ZHAO Xin-Yu　LIN Zuo-Quan
(Department of Information Science, Peking University, Beijing 100871)

Abstract　With the growing number of distributed information system, the problem of how to automatically supervise and control business processes in dynamic, data rich environments is becoming increasingly critical. We present a model of Agent with monitoring capabilities to liberate human beings from potentially monotonous monitoring tasks. Conventional BDI Agent model is extended to BDIC(Belief, Desire, Intention and Capability) model with monitoring capabilities. The monitoring capabilities are induced from a real life application background so that they are domain-independent. The implemented Agent model has been successfully adopted in a customer service management(CSM) system which proves that it is suitable for supervising the execution of the business processes in the enterprise information systems with its re-useable and customizable characters.

Keywords　Agent, BDI, Monitoring, Capability, Business processes

1　引言

在大信息量、动态环境下对业务流程的监控和管理是大型分布式应用软件系统一个非常重要的功能，许多应用领域中无法人工处理大量的来自系统内部和其他系统的信息。在企业软件系统运行过程中不可避免地会遇到很多例外情况，正确、及时地处理这些问题对于保证企业软件系统的良好运转具有十分重要的意义。随着业务流程数量和复杂程度的增加，软件系统对业务流程的自动监控和管理的需求越来越大，尤其是对业务流程运行的实时监控、预警、异常处理以及访问控制等方面：

• 虽然可以在系统设计时充分考虑各种情况，提供系统对特殊事件的处理能力，但现有系统很少能直接提供监控工具，从系统整体角度出发对系统运行过程中出现的特殊事件进行推理并做出适当的响应动作。

• 在动态环境下，软件系统需要持续不断地对发生的事件进行监控，传统软件系统的管理模式往往不能及时发现问题。持续监控和预警可以及时对各类异常和风险加以提示，避免业务流程运行失败，由此对企业造成损失。因此，软件系统中的预警功能十分必要。

• 复杂的业务流程在发生例外或运行失败的情况下，需要自动地针对不同的失败模式进行不同方式的处理和恢复操作。

• 传统的流程管理软件对于一个业务流程的实例，不能从系统整体角度出发，根据用户与该业务流程实例相关的信息，输出该业务流程实例整体的运行情况。不同类别的用户有通过管理和监控接口获取不同级别的业务流程视图的需求。

Agent 是可以为我们完成特定任务的具有行为能力的对象，在多 agent 系统中，agent 之间及 agent 与环境之间通过通信、协商和协作来共同完成任务。agent 具有的自主性、社会性、反应性和预动性[1]等性质十分适合解决上述问题。BDI 模型是 Rao 和 Georgeff[4,5]、Cohen 和 Levesque[3]以及 Bratman[2]等提出的一种重要的 agent 认知模型，并得到广泛的应用[4,6]。目前已经有一些工作探讨把多 agent 模型应用于业务流程的监控管理中[7~9]。

本文在一个企业的应用背景中归纳出独立于应用领域的监控能力分类，提出一个具有监控能力的 BDIC agent 模型，该模型是在 BDI 模型的基础上加入能力组件而得到，能力组件决定了 agent 的类别和监控能力。该 agent 模型适合对企业应用软件系统中的业务流程进行监控和管理。根据此模型设计开发的软件系统，可以根据用户的配置信息，监控和管理业务流程的运行，在一定程度上对发生的异常情况进行处理和恢复操作，并可以与其他系统中的 agent 合作，获得与业务流程实例的相关信息，为不同类别的系统用户提供获取同一业务流程不同级别信息视图的接口，将该业务流程的整体视

*)本文得到国家自然科学基金(编号：60373002，60496322)和“973”项目(2004CB318000)的资助。赵新宇　博士研究生，主要研究方向：基于 Agent 的管理软件、人工智能；林作铨　教授、博士生导师，主要研究方向：计算机软件，人工智能。

图返回给用户。该模型已在一个客户服务管理系统(CSM)中成功应用,说明本文提出的 agent 模型以及监控能力分类具有良好的可重用性和可定制性。

本文第 2 节给出 agent 的模型、性质以及应用背景;第 3 节结合应用背景中的实例给出 agent 具有的监控能力分类的详细说明;第 4 节说明该模型在一个客户服务管理系统中的实现和应用;第 5 节对相关工作进行比较;最后,我们简要地进行总结并指出进一步的工作。

2 Agent 模型

2.1 BDIC 模型

如今的企业计算环境具有开放和不可预测性,数量巨大的业务流程在企业应用中运行,它们之间进行着复杂的交互。很难想象一个系统在设计时可以完全了解运行时可能出现的各种异常情况和新的需求,有必要定义新的模型来适应这种应用的开发需要。文[10]提出一种可动态加载能力的多 agent 模型,对供应链进行建模。这里提出 BDIC(Belief-Desire-Intention-Capability)模型,用于对业务流程运行进行自动监控和管理,给出了 BDIC Agent 模型在系统开发不同阶段的开发方法。BDIC Agent 模型与文[10]中的 agent 模型的不同之处在于:加入了体现可定制性的用户配置和对业务流程监控能力的详细分类。

BDIC 模型是在 BDI 模型的基础上加入能力组件而得到的,能力组件决定了 agent 的类别和监控能力。

定义 1 具有监控能力的 Agent 定义如下:Agent $=(B, D, I, Tr, Se, Ef, Ev, P, Pr, Ca)$,其中(见图 1):

B:agent 的信念,是关于 agent 所监控和管理的系统中业务流程的信息;

D:agent 的愿望,是 agent 想达到的目标;

I:agent 的意图队列,根据当前信念选择队列中的规划进行执行;

Se:感知器,感知外界环境输入,更新信念,形成所要执行事件的愿望,加入 D 中;

Ef:效应器,agent 通过效应器执行规划;

Ev:评估器,对外来事件进行评估,决定是否执行相应的管理动作;

P:处理器,负责任务的分配和调度;

Ca:监控能力,agent 具有的、能执行的监控和管理的动作(将在第 3 节详细介绍);

Pr:配置(profile),是用户定制的需要监控和管理的业务流程信息。

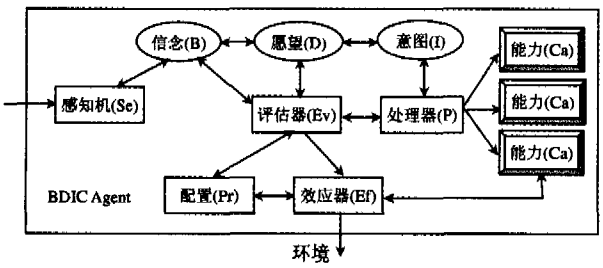

图 1 BDIC Agent 体系结构

BDIC 模型的 agent 通过感知机 Se 能感知外界环境输入,以标准格式的方法接收外来信息,同时更新信念 B。agent 的信念 B 包括本身状态信息(如名称、地址、能力组件信息等)以及其他 agent 状态信息(包括合作伙伴的信息、合作关系、通讯代价等);agent 从接收到的消息中过滤出有用的信息,根据当时精神状态(B、D、I)和用户定制的配置信息 Pr 形成新的愿望 D;然后由评估器 Ev 对愿望 D 进行评估,决定是否执行相应的管理动作以及执行愿望 D 的优先级;如果需要执行该管理动作,则根据能力组件获取监控行为 Ca,形成意图 I,然后由处理器 P 进行分配和调度,通过效应器 Ef 作用于环境,执行对业务流程监控和管理的动作。

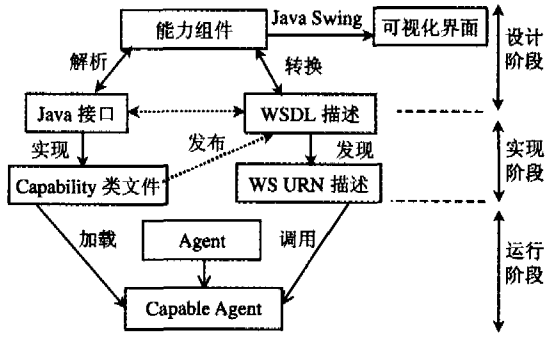

图 2 BDIC Agent 模型开发方法

BDIC Agent 模型中的能力组件具有一定程度的可扩展性和可重用性,在不同的开发阶段对系统提供支持(见图 2)。在设计阶段,通过分析企业应用系统中需要的监控和管理的业务流程的特征和要求,从已设计好的能力组件库中选择已定义好的监控能力组件,通过可视化界面进行选择、编辑。例如,不同类别的系统用户有通过管理和监控接口获取不同级别的业务流程视图的需求,因此可从监控能力组件库中选择访问控制能力组件赋予 agent,使之具有根据用户与该业务流程实例相关的信息,输出该业务流程实例整体的运行情况的监控和管理能力;在实现阶段,利用基于 Java 语言的 RDF 解析器[11]将监控能力组件(采用 RDFS 对能力组件进行定义,能力组件描述的细节可以参考文[10])转换成 Java 语言的接口(Interface)。在不同的应用领域,监控技术和实现算法也有所不同,所以不同系统的开发者根据实际监控和管理需要为能力组件实现不同的代码(完成不同的功能),实现的代码编译成不同的类文件;在系统运行阶段,agent 根据需要查找、匹配、动态加载适当的监控能力组件,成为具有该种监控能力的 agent。例如,复杂的业务流程运行时出现的不同异常情况,需要不同的处理方式;业务流程在运行失败的情况下,agent 针对不同的异常情况自动地加载异常处理能力组件,对发生异常的业务流程进行恢复操作。同时,BDIC Agent 模型的能力组件的定义可与 Web Service 描述语言 WSDL[12] 之间进行转换,使 BDIC Agent 可以向 Web 上发布自己的监控能力组件,发现 Web 上的适当的具有监控和管理功能的 Web Service,进一步加强 BDIC Agent 对业务流程的监控和管理能力。

agent 监控和管理的对象是业务流程,业务流程是为实现商业目标或策略而联系在一起的活动或过程的集合。一般地,一个业务流程包括执行需要满足的前提条件、执行后的效果以及执行过程中用到的中间变量和执行体,下面给出业务流程的定义。

定义 2 业务流程 BP $=\{$precondition, effect, body, variation$\}$,其中 precondition, effect, body, variation,分别是业务流程的前提、效果、执行体和业务流程执行所用到的中间变

量。

2.2 系统背景

本文从一个具体企业的应用背景中归纳出企业软件系统对业务流程的监控管理应该具有的行为分类，这些经过分类的行为具有一定的可重用性，适用于对业务流程的监控和管理。本文所用到的应用背景是在一个家电制造销售类企业客户关系管理/呼叫中心的客户服务管理系统(CSM)中处理客户来电请求的典型业务流程，如图3。

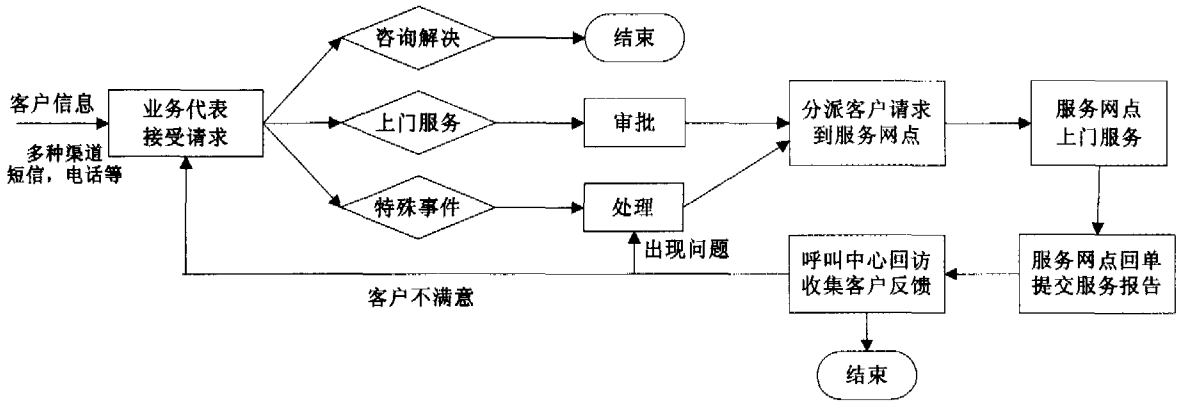

图3 呼叫中心处理客户来电的业务流程

CSM中的业务流程是从呼叫中心受理客户来电开始的，呼叫中心业务代表(座席)根据客户请求将来电信息分为咨询解决、上门服务和特殊事件三类，每类有不同的处理流程。咨询类来电信息只通过专家和客户交谈就可以解决；上门服务类和特殊事件类通过审批处理流程后由呼叫中心业务代表根据客户地域和服务网点业务合作伙伴的维修、安装能力进行派工处理；服务网点在规定事件内接到呼叫中心的派工信息后到客户处进行处理，处理完毕后通过系统进行回单操作；呼叫中心定期对服务网点完成的回单进行回访，确认服务网点的服务质量；若出现问题或客户不满意的现象，则该请求信息状态被重置，重新进行处理。由于软件系统的复杂性和动态性，常常导致业务流程不能按用户预料那样运行，需要持续不断对系统发生事件进行管理和监控，给予实时的响应。下文将以该业务流程为背景，具体阐述agent的监控能力分类。

3 监控能力分类

BDIC Agent模型中的监控能力应该具有一定程度的可重用性，即不仅适用于本文提到的应用领域和背景，还应该可以通过泛化使之广泛适用于企业软件系统对业务流程的监控管理。在实际应用中，针对不同的领域，监控技术和实现算法也有所不同，这里给出抽象化的描述，如图4所示。

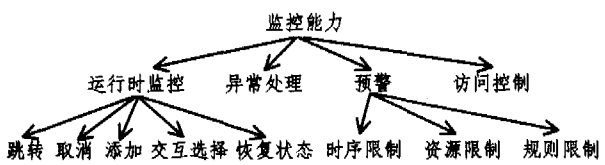

图4 BDIC Agent的监控能力分类

3.1 运行时监控

下面是一个agent在业务流程运行时对其进行管理和监控的解释器，目的是在抽象的层次上解释业务流程运行时agent的监控机制：

```
Monitoring-interpreter
Initialize-state();
repeat
   if all of currentBP. effects are true
                    //当前执行的流程的效果是否已实现
   then currentBP = getNextBP();
        continue;
     // 获得下一步所要继续执行的流程(动作1 跳转)
   end if
   if one of currentBP. precondition is false
     // 当前执行的流程的前提是否已得到满足
   then tempBPList = getBPList(); //获得一个流程列
   //表，列表中的流程的效果可以使当前流程的前提满足
         if tempBPList == null
       // 不存在使当前流程前提满足的流程(动作2取消)
     then reportToAdmin()// 向管理员发送提示信息
          currentBP = getNextBP()
          continue;
          // 获得下一步所要继续执行的流程
          else currentBP = getBestBP(tempBPList)
               continue;
               // 通过与用户交互选择，添加一个可以高质量完成目
               标的流程执行(动作3 添加)(动作4 交互选择)
          end if
   end if
   if currentBP. variationis false
   // 当前执行的流程所需的某个变量(该变量是某个已完成的流程
   的效果)不被满足
   then currentBP = getProperBP(currentBP. var);
        continue;
        // 查找到已完成的流程，恢复其初始状态，重新执行该已完
        成的流程(动作5 恢复状态)
   execute(currentBP. body)
end repeat
```

根据运行解释器，agent在业务流程运行时进行管理和监控主要有以下五种动作：

动作1 跳转。业务流程执行过程中，由于系统环境是动态、不断变化的，如果当前要执行的业务流程的目标已得到满足，就不必按正常的次序再次执行该业务流程。agent监控每个业务流程的目标，若当前所要执行的业务流程的目标已经满足，则agent改变业务流程执行走向，跳过当前业务流程，转向执行完该流程所要继续执行的业务流程。

例如，在呼叫中心的应用背景中，呼叫中心座席已受理客户的请求信息但还未派工时，客户再次来电报告故障现象解除，则此时派工业务流程的目标(即解除客户报告的故障现象)已实现，agent将业务流程导向回访业务流程(确认服务质量)。

动作2 取消。业务流程执行过程中，当前要执行的业务流程的前提条件有可能是不可实现的，如果按正常的次序执行，系统就会出现例外。agent监控每个业务流程的前提条件，若无法通过添加子流程，使当前要执行业务流程的前提条件成立，则agent取消当前执行的流程，并向管理员发送信息，报告取消执行业务流程事件，并同时跳转到执行完该取消

的流程后所要执行的业务流程。

例如,在呼叫中心的应用背景中,呼叫中心进行派工业务流程之后,服务网点发现派工数量超过维修处理能力的允许条件。Agent 监控到此类事件,向呼叫中心座席发送消息。呼叫中心座席接收到信息后将取消对该网点分配的服务任务,重新执行派工业务流程。

动作 3 添加。业务流程执行过程中,环境的改变常常使得正要执行的某个业务流程的前提条件不满足,不能按正常的流程进行。agent 监控每个流程的前提条件,若当前所要执行的流程的前提条件不被满足,agent 可以根据用户的配置信息添加一个新的流程。该流程的结果就是使当前流程的前提条件得到满足并能按正常的次序执行下一个业务流程。

例如,在呼叫中心的应用背景中,在呼叫中心进行派工业务流程之后,服务网点还没有到客户处进行处理时,客户再次打电话来要求更正上门服务的时间。此时,若不通知服务网点更改处理的前提条件(上门服务的时间),则当前执行的流程(服务网点上门服务)将不能正常执行。此时,agent 自动添加一个更改客户要求维修时间的流程,修改服务网点上门处理的时间,并使用预警功能(下节介绍)通知服务网点,使得服务网点上门处理的业务流程能正常进行。

动作 4 交互选择。业务流程执行过程中,常常会出现有多种可供选择的动作可以达到目标,需要用户的判断和决策才能使业务流程继续进行或者进一步提高完成目标的质量。BDIC Agent 根据用户的配置监控出现具有选择动作需要用户交互的流程,及时发送信息给用户,获取用户的知识和决策信息,以协助并高质量完成业务流程的目标。

例如,在呼叫中心的应用背景中,呼叫中心派工后,由于 CSM 系统组织结构存在地理上的分布性,服务网点可能未及时地接收派工信息并上门处理。agent 的预警能力(下面将要介绍)将对该类事件进行提示,呼叫中心座席人员得知该情况后可以通过其他方式(CSM 系统采用前后台分离方式,前台负责硬件接口的处理,因此通知方式有传真、短信、Email 等多种方式)通知服务网点。为了让服务网点尽早接收派工信息,agent 需要与呼叫中心座席人员交互,以获得最好的通知方式;agent 获得该知识后,更新自己的信念。下次若用户没有特别指出预警发送方式,agent 则根据自己的信念中的预警方式,对该服务网点进行预警通知。

动作 5 恢复状态。业务流程执行过程中,常常出现由于某种原因(已完成的目标重新不满足)需要恢复已完成的业务流程的状态,重新执行已完成的业务流程。agent 根据用户的配置监控已完成的业务流程的目标。若由于环境变化导致已完成的业务流程的目标不再被满足,agent 将控制流程执行方向,重新执行该已完成的业务流程。

例如,在呼叫中心的应用背景中,服务网点上门处理客户请求后,呼叫中心对客户进行回访,调查满意程度,agent 监视回访结果。若出现客户报告仍旧出现故障的情况,agent 将整个业务流程转向到来电受理流程,重新处理客户请求。

3.2 预警

在实际应用领域中,业务流程的运行常常受到一些限制,系统需要持续不断对发生的事件进行监控,其中部分事件需要实时的响应,传统的系统管理模式往往不能及时发现问题。持续监控和预警可以对各类异常和风险及时加以揭示,避免造成更大的损失。在实际应用中,针对不同的限制,预警技术和算法也有所不同,这里给出抽象化的描述。

定义 3 预警 Alert 定义如下:$Al=(Id,Re,Ty,Le,Ch,Co)$,这里,

- Id :预警的唯一标识符;
- Re :预警的接收者,通常是业务流程中为客户提供服务的企业合作伙伴;
- Ty :预警类别,包括破坏时序限制、资源限制和策略限制三类,将在下文详细阐述;
- Le :预警紧急程度
- Ch :预警发送渠道,包括传真、短信等(见系统实现部分);
- Co :预警信息内容。

其中 Id 标识了该预警信息;Le 表明了预警信息的紧急程度;Ch 表明预警信息的发送渠道,在系统实现中支持 agent 以传真、短信等方式发送;在业务流程过程中,如果破坏了系统的时序限制、资源限制或策略限制,agent 将发送一条预警信息 Al 给相关的接受者 Re。下面将详细阐述这三种类别的预警信息。

3.2.1 时序限制

业务流程通常要求开始时间和完成时间在一定时间范围内,时序限制包括标明时间范围的本体,如表 1 所示。

表 1 时间范围本体

类型(Temporal_ontology)	含义
Start-at	开始时间必须在
Start-lt(later than)	开始时间必须晚于
Start-et(earlier than)	开始时间必须早于
End-at	结束时间必须在
End-lt(later than)	结束时间必须晚于
End-et(earlier than)	结束时间必须早于

时序限制表示如下:

BPName temporal_ontology time [&temporal_ontology time…] action[;action…]

其中,BPName 表示业务流程名称,temporal_ontology 是时间范围本体,time 表示时间值,action 是该时序限制被违反时 agent 执行的动作。

在呼叫中心的应用背景中,系统向用户提供获取业务流程整体视图的接口,但由于生成业务流程的整体视图是个复杂的流程,需要在时序、系统资源和用户的级别上加以限制。因此,在 agent 的配置中具有如下的时序限制:

getBPView Start-et 8:00:00 & Start-lt 20:00:00 reportToAdmin ; refuseGetBPView

这条时序限制的含义是:用户向系统请求获取业务流程整体视图的开始时间必须早于上午 8 点、晚于晚上 8 点,这样系统生成业务流程整体视图的时间区间就可以与系统中其他业务流程运行的高峰期避开,系统不致于负荷过大,以保证系统运行安全。如果用户在上午 8 点到晚上 8 点之间请求获取业务流程的整体视图,agent 将向系统管理员报告这次用户请求并拒绝该次请求。

3.2.2 资源限制

业务流程的执行不能违反系统资源有限的限制。资源限制表示如下:

BPName resource relation threshold[&resource relation threshold…]action[;action…]

其中,resource 表示系统资源名称,relation 表示数量关

系，threshold 表示该资源的阈值，action 是该资源限制被违反时 agent 执行的动作。

在呼叫中心的应用背景中，agent 的配置里有如下的资源限制规则：

getBPView runtime < 86400s reportToAdmin; refuseGetBPView

这条资源限制的含义是：用户向系统请求业务流程整体视图执行时间不能超过阈值 86400 秒，这样可以保证及时中断执行时间过长的业务流程，保证有足够的系统资源运行其他业务流程。如果系统生成业务流程整体视图的时间超过阈值，agent 执行向系统管理员报告这次用户请求并拒绝该次请求的动作。

3.2.2 策略限制

业务流程通常会定义一些策略，这些策略在整个业务流程的执行过程中是不能违反的，否则将导致该流程的失败甚至整个系统的崩溃。agent 从用户配置中获得需要监控的业务策略，持续不断地监控业务流程的执行是否满足策略规则的限制，如果出现违反策略规则的事件，agent 向用户发出警告信息。

策略限制表示如下：

BPName condition[&condition…] action[;action…]

其中，BPName 表示业务流程名称，condition 是策略规则，action 是该策略限制被违反时 agent 执行的动作。

在呼叫中心的应用背景中，agent 的配置里有如下的策略限制：

getBPView userLevel > 3 reportToAdmin ; refuseGetBPView

该策略限制的含义是：向系统请求业务流程整体视图的用户，其级别必须高于三级，不是所有用户都可以向系统请求生成业务流程的整体视图，这就保证了系统运行的安全性。如果低于该级别的用户请求生成业务流程整体视图，agent 执行向系统管理员报告这次用户请求并拒绝该次请求的动作。

3.3 异常处理

业务流程在发生异常或运行失败的情况下，agent 根据用户的配置，会执行一个特殊的流程来取消发生异常的业务流程所产生的不良影响。用户对容易出现异常的业务流程定义它的恢复流程，进行异常处理。对正常流程 T 定义恢复流程 T*：如果 T 正常执行，则 T* 为空操作，否则 T* 实施的是 T 的异常处理操作。

例如呼叫中心座席接听客户电话（IP 电话）进行受理业务时，受通讯线路影响，可能会出现掉线、无反应等现象。若持续 3 分钟客户无通话内容，agent 视为通讯故障，做异常处理，执行受理业务流程 Accept 的恢复流程 Accept*，对已取得的客户的请求信息进行保存，待下次该客户再次打进电话时继续受理流程：

```
Accept* :
if reponse_time>3 minutes then store_state()
else null
```

异常处理和事务已经成为设计和实现高可靠性软件系统最重要的概念，人们为更好地解决工作流的事务特性和解决工作流的异常处理问题提出了嵌套事务模型[13]、分支/汇合事务模型[13]、SAGAS[15]、BTP[16]等许多高级事务模型，本文不做讨论。

3.4 访问控制

BDIC 模型的 agent 可以向用户提供完整的业务过程运行的细节信息，为不同类别的系统用户提供获取同一个业务流程的实例在不同层次上的信息视图的接口，系统用户可以通过 agent 提供的接口获得特定业务流程实例的整体视图。对于一个业务流程的实例，agent 通过与其他系统中的 agent 合作获得用户以及该业务流程实例的相关信息，根据用户级别、地域等与该业务流程实例相关的信息，输出该业务流程实例整体的运行情况。

在 CSM 系统中存在不同级别的用户，包括呼叫中心座席人员、总部管理人员、经营部用户和服务网点用户。他们向系统请求同一个业务流程实例运行情况的整体视图的时候，所获得的信息应该是不同的。agent 根据系统管理人员的配置获取业务流程实例的相关信息（主要是产品类别信息），通过 WebDaemon[1] 安全服务的认证授权协议（AAP）与 WebDaemon 系统中负责访问控制的 agent 通讯，获得请求者所在地域和具有的角色信息后，向不同级别的请求者提供该业务流程实例运行情况不同的整体视图。这种设计和实现方法在实际系统运行中取得了稳定可靠的结果（见第 4 节）。

4 系统实现

本文提出的 agent 模型及其监控能力分类已成功应用于一个企业的呼叫中心客户服务管理系统（CSM）中[2]。CSM 为企业产品售后服务管理提供运营支撑（见图 5），系统采用基于 J2EE 平台的 B/S 体系架构，其中 agent 通过 EJB（Enterprise Java Bean）[18] 实现。根据用户配置监控一个或者多个复杂的业务流程，执行监控能力分类、定义的监控动作。整个呼叫中心系统采用基于业务契约中间层的前后台连接方法[19]，该方法在传统的呼叫中心项目软件结构框架中添加一个新的层次——业务契约层。其实现的主要部分为一个前后台共同遵守的通信协议，将与业务流程相关的功能和与硬件相关的功能（短消息、传真等）连接起来。利用中间层，可以将预警信息发送给管理员；系统中的 agent 与 WebDaemon 集成提供访问控制的功能（见 3.4 节访问控制）。整个业务流程的状态信息可以通过基于 Web 的方式进行实时管理和查询，应用背景中的派工业务流程的监控管理实现界面如图 6 所示。

5 相关工作

业务流程的规划和生成备受重视的同时，关于业务流程运行时的管理和监控在文献上却较少被关注。文[7]旨在去掉传统上生成规划时认为世界是静态和确定的假设，着重研究了在动态环境下生成业务流程规划时的监控管理，提出智能监控的概念和分类，讨论了不同类别的智能监控在世界变化情况下对已生成的规划的修改和转化。本文提出的 agent 模型着重于业务流程运行时的动态监控和管理，并能根据用户配置，在变化更迅速、更丰富的情况下，对业务流程的运行进行监控管理，提出预警信息。

文[8]对 Robot 行为的实时监控进行了研究，但并没有对分类进行更深层次的分析和细化，而是着重提出了 Robot 系

[1] 身份认证及访问控制系统 WebDaemon[17] 采用基于角色的访问控制策略，已为多个应用系统提供安全服务。

[2] 集团有限公司呼叫中心，Http://www.tcl.com

统运行时获取的信息值(VOI)和警告值(VOA),并结合两个应用领域分析了两者之间的关系。本文提出的 agent 模型在它们的基础上重新提出监控能力的分类,更适合于软件业务流程运行时的监控和管理。

OVERSEER[9]是一个基于规划识别(Plan Recognition)技术的监控系统,即将一个系统的规划交给 OVERSEER,在不改变原有系统运行情况的基础上通过观察原有系统 agent 间的交互过程,推理整个规划的运行情况,提出了一个适合在低带宽条件下的一致性假设(Coherent hypothese)检查算法 YOYO*。

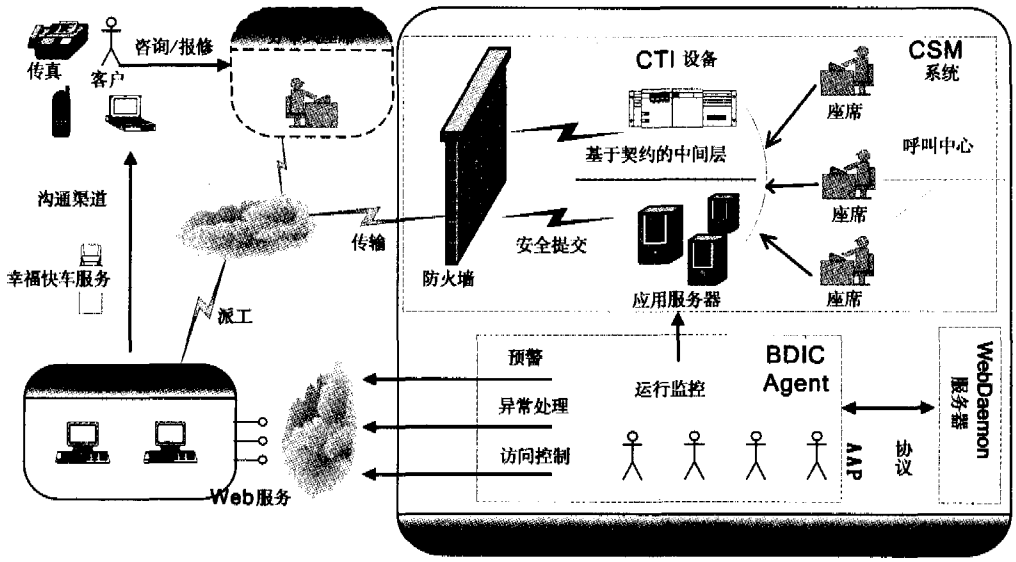

图 5 BDIC Agent 模型在企业应用中的配置

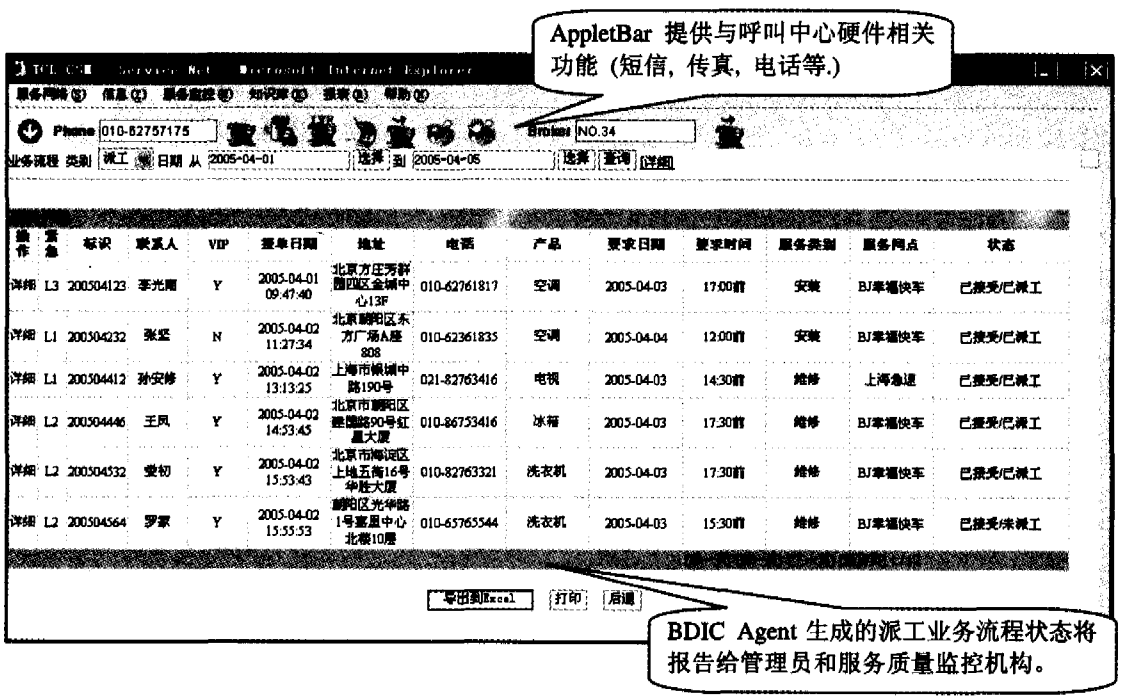

图 6 BDIC Agent 生成的派工业务流程详细信息

这些工作和系统已应用于分布式系统的监控和管理,它们或侧重于业务流程的定义和生成时的管理[9],或侧重于检测例外、发出警告[7,8],但在业务流程运行时对例外事件处理和业务流程的管理能力,以及根据用户需求获取业务流程不同视图等方面考虑不够,尚显不足。与以上工作相比,本文提出的 agent 模型扩展了基本的 BDI Agent 模型,加入监控能力和反映用户需求的配置信息,具有一定可重用性和可定制性,支持分布式系统在动态、大信息量的环境下对业务流程进行实时的管理和监控。

具有可重用的监控能力分类以及根据用户需要而定制的配置信息的 BDIC Agent 模型适用于电子商务平台、Web Service、Internet 平台相关应用[20]。

结论和进一步工作 本文在 BDI 模型的基础上,加入监控能力组件,提出一种具有监控能力的 BDIC Agent 模型,给出了 BDIC Agent 模型的体系结构和开发方法。监控能力组件包括业务流程运行时的监控、预警、异常处理和访问控制;通过客户服务管理系统(CSM)中的实例介绍了该 agent 模型以及监控能力分类的应用情况,并与一些基于 agent 技术的

业务流程管理和监控的研究成果进行了比较;说明该 agent 模型以及监控能力分类具有可重用性和可定制性使其更适合在大信息量、动态环境下对软件系统中的业务流程进行管理和监控。

以下几个问题是我们今后研究的重点:在更多领域中应用该模型和监控能力分类,开发更为详细的监控能力分类;进一步完善监控能力中对异常情况的处理和恢复操作,可与业务流程的事务方面的研究相结合;语义 Web 服务(Semantic Web Services)[21]本质上是一种业务流程,关于其运行监控的规范和形式化工作目前还没有全面展开[22],将本文提出的 agent 模型和监控能力引入语义 Web 服务的管理和监控也是一种尝试。

参 考 文 献

1 Woodridge M, Jennngs N R. Intelligent Agent: Theory and Practice, The Knowledge Engineering Review, 1995, 10(2): 115~152
2 Bratman M. Intentions, Plans, and Practical Reason. Cambridge: Harvard University Press, 1987
3 Cohen P R, Levesque H J. Intention is choice with commitment. Artificial Intelligence, 1990, 42(2~3): 213~261
4 Rao A S, Georgeff M P. BDI Agents: From Theory to Practice. In: Proceedings of the First International Conference on Multi-Agent Systems(ICMAS'95), 1995. 312~319
5 Rao A S, Georgeff M P. Modeling rational agents within a BDI-architecture. Allen J, Fikes R, Sandewall E editors. In: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning(KR'91), 1991. 473~484
6 Van Dyke Parunak H. Industrial and Practical Applications of Agent-based Systems. Weiss G, editor. Cambridge, Massachusetts, London, England: MIT Press, Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence, 1999. 377~424
7 Veloso M, Cox M. Rationale-based monitoring for planning in dynamic environments. In: Proc. of the 1998 International Conference on AI Planning Systems. Menlo Park: AAAI Press, 171~180
8 Wilkins D E, Lee T J, Berry P. Interactive Execution Monitoring of Agent Teams. Journal of Artificial Intelligence 2003, 18: 217~261
9 Kaminkaa G, Pynadath D, Tambe M. Monitoring deployed agent teams. In: Proc. of Autonomous Agents'01, Montreal, Canada, 2001. 308~315
10 赵新宇,林作铨. 基于 Agent 的供应链模型. 计算机科学,2004, 31(8):16~21
11 ARP: Another RDF Parser. Http://www.hpl.hp.com/personal/jjc/arp/
12 Web Services Description Language(WSDL). Http://www.w3.org/TR/wsdl
13 Moss J E B. Nested Transactions: An Approach to Reliable Distributed Computing: [PhD thesis] MIT, Dept of Electrical Engineering and Computer Science, 1981
14 Pu C, Kaiser G E, Hutchinson N C. Split-Transactions for Open-Ended Activities. Proceedings of the Fourteenth International Conference on Very Large Data Bases, Los Angeles, CA, 1988. 26~37
15 Garcia-Molina H, Salem K. SAGAS. International Conference on Management of Data and Symposium on Principles of Database Systems. San Francisco, California, United States: ACM Press, 1987
16 Ceponkus A, et al. Business Transaction Protocol. 2002, OASIS Committee Specification. 2002
17 桂艳峰,林作铨. WebDaemon:一个 Web 安全访问控制系统. 计算机研究与发展,2003,40(8):1186~1194
18 Java 2 Enterprise Edition home page. Http://java.sun.com/j2ee
19 Wu Cen, Lin Zuoquan, Zhao Xinyu, et al. Contract-based Interlayer: A Two-way Approach to Integrate Call Center With J2EE Framework, International Symposium on Distributed Computing and Applications to Business. Engineering and Science, DCABES, 2004
20 Zhao Xinyu, Wu Cen, Zhang Runjie, et al. A Multi-Agent System for E-Business Processes Monitoring in a Web-Based Environment. Chen Jian(Ed). The Fourth International Conference on Electronic Business - Shaping Business Strategy in a Networked World(ICEB 2004), Academic Publishers/World Publishing Corporation, 2004. 470~475
21 McIlraith S A, Son T C, Zeng H. Semantic Web Services. IEEE Intelligent Systems, 2001, 16(2): 46~53
22 DARPA Agent Markup Language. DAML-S/OWL-S 1.1 Draft Release, 2004. http://www.daml.org/services/owl-s/1.1//

(上接第 6 页)

34 Peng J, Jay Kuo C-C. Geometry-guided progressive lossless 3D mesh coding with octree(OT) decomposition. ACM Transactions on Graphics, 2005, 24(3): 609~616
35 Zhang Jinghua, Owen C B. Octree-based Animated Geometry Compression. In: Data Compression Conference, Snowbird, 2004. 508~520
36 Saupe D, Kuska J -P. Compression of isosurfaces. In: Proc. of IEEE Vision, Modelling and Visualization, Stuttgart, 2001. 333~340
37 Mroz L, Hauser H. Space-Efficient Boundary Representation of Volumetric Objects. In: Proc. of the Joint Eurographics-IEEE TCVG Symposium on Visualization, Ascona, 2001. 193~202
38 Yang S N, Wu T S. Compressing isosurfaces generated with marching cubes. The Visual Computer, 2002, 18(1): 54~67
39 刘迎,蔡康颖,王文成,吴恩华. 基于 Marching Cubes 重组的外存模型渐进压缩. 计算机学报,2004,27(11):1457~1463
40 Bertram M, Duchaineau M A, Hamann B, et al. Generalized B-spline subdivision -surface wavelets for geometry compression. IEEE Transactions on Visualization and Computer Graphics, 2004, 10(3): 326~338
41 Sweldens W. The lifting scheme: a custom-design construction of biorthogonal wavelets. Applied and Computational Harmonic Analysis, 1996, 3: 186~200
42 Szymczak J, Rossignac D. Piecewise Regular Meshes: Construction and Compression. Graphical Models, 2002, 64(3-4): 183~198
43 Attene M, Falcidieno B, Spagnuolo M, et al. SwingWrapper: Retiling Triangle Meshes for Better EdgeBreaker Compression. ACM Transactions on Graphics, 2003, 22(4): 982~996
44 Surazhsky V, Gotsman C. Explicit Surface Remeshing. In: Proc. of Eurographics Symposium on Geometry Processing, Aachen, 2003. 17~28
45 Lee A W F, Sweldens W, Schröder P, et al. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In: Computer Graphics Proc. Annual Conference Series, ACM SIGGRAPH, Orlando, 1998. 95~104
46 Cignoni P, Rocchini C, Scopigno R. Metro: Measuring error on simplified surfaces. Computer Graphics Forum, 1998, 17(2): 167~174
47 Pajarola R, Rossignac J. Compressed Progressive Meshes. IEEE Transactions on Visualization and Computer Graphics, 2000. 6(1): 79~93
48 Khodakovsky A, Guskov I. Compression of Normal Meshes. In: Proceedings of Geometric Modeling for Scientific Visualization, Springer Verlag, 2004. 189~206
49 Guskov I, Vidimce K, Sweldens W, et al. Normal Meshes. In: Computer Graphics Proc. Annual Conf. Series, ACM SIGGRAPH, New Orleans, 2000. 95~102
50 Garland M, Heckbert P S. Surface Simplification Using Quadric Error Metrics. In: Computer Graphics Proc. Annual Conf. Series, ACM SIGGRAPH, Los Angeles, 1997. 209~216