高技术通讯 2010 年 第 20 卷 第 4 期:379 ~ 385

doi:10.3772/j.issn.1002-0470.2010.04.009

基于主体的软件的故障诊断系统 eHealer①

张冬蕾② 韩 旭 史忠植

(* 中国科学院计算技术研究所智能信息处理重点实验室 北京 100190)

(** 中国科学院研究生院 北京 100049)

摘 要 提出了一种通过监控多主体系统中各主体之间的消息通信定位故障主体的故障诊断方法。对于在多主体平台上运行的应用软件系统,首先对其功能进行建模,进而对其行为进行建模,然后在软件实际运行过程中捕获其中多个主体之间的通信消息及各类事件,通过设计的诊断算法与软件行为模型进行匹配和比较,通过推理发现运行过程中出现故障的主体及其某个异常行为,从而实现对软件系统的故障进行定位和诊断。在此基础上,实现了一个原型系统 eHealer。实验表明,该方法能够准确定位多主体软件系统中 agent 级别的故障,与已有方法相比,具有实时性强、定位准确、普适性好的优点,为进一步选择故障恢复策略提供了有效的信息。

关键词 故障诊断(MAS),多主体系统,系统功能模型,主体行为模型,比较诊断算法

0 引 言

故障诊断是人工智能和系统仿真中的一个重要研究方向。运行中的软件一旦发生故障而又无法及时发现,不仅软件功能无法正常完成,还可能造成巨大的损失。亚马逊网站 2008 年 6 月出现的故障,造成服务中断数小时,据统计平均每分钟损失达 3.1 万美元。随着软件日趋呈现网络化、复杂化和大型化,软件的自我诊断和修复能力已经成为越来越迫切的需求。

本文在已有工作的基础上[1],研究了 MAS 中的故障诊断问题,将基于模型的诊断技术应用于多主体平台上的应用软件主体故障诊断过程,给出了一种利用多主体之间消息通信来进行故障定位和诊断的新方法,并构建了诊断原型系统 eHealer 的应用模型及其体系结构,最后讨论了系统的优缺点及扩展问题。

1 现有故障诊断方法和多主体系统故障分析

已有故障诊断的方法主要有基于规则推理、基于模型、基于人工神经网络和基于案例的方法。其中基于模型的诊断(model based diagnosis, MBD)[2-4]近年来已发展为故障诊断研究中的一个十分活跃的研究分支。因为基于模型的诊断系统不依赖诊断专家的经验知识或已有案例,只需通过为诊断对象的各构件的特征建模即可表征诊断对象的深度知识并实施诊断,从而使得数学逻辑模型与诊断推理过程相互独立,降低了诊断方法对诊断对象的依赖性。de Kleer 及 Williams 所提出的通用诊断引擎(general diagnostic engine, GDE)[2,3]是一种很有效的基于模型的诊断方法,其原理是:通过测量获取待诊断对象的观测形态,然后基于系统所有组件的正常状态模型获得待诊断对象预测形态。根据这两种形态的差异,计算出最小冲突集。由最小冲突集计算出最小诊断,即关于故障元件的假设。早期的研究主要针对集中式、静态系统,而随着诊断对象范围的扩大,出现了针对分布式和动态系统的诊断方法,研究各个子系统局部诊断之间的协调与合成,以及如何高效地匹配系统行为模型与观测序列[5-8]。

软件实体是构成软件系统的基本元素,其发展经历了语句、过程、模块、抽象数据类型、对象、构件、服务等多个侧面与层次。在当前开放、动态和多变的 Internet 环境下,软件实体的主体化已经成为其发展的主要趋势,即软件实体应具有内容的自包含性、

① 863 计划(2007AA01Z132),973 计划(2003CB317004,2007CB311004),国家自然科学基金(90604017,60435010,60775035)和国家科技支撑计划(2006BAC08B06)资助项目。

② 女,1979 年生,博士生;研究方向:主体技术,网构软件等;联系人,E-mail: zhangdl@ics.ict.ac.cn

(收稿日期:2009-01-05)

结构的独立性与实体的适应性[9]。分布式的多主体系统(multi agent system, MAS)[10]正是满足了这一需求,成为构建应用软件的不可或缺的方法之一。

从故障诊断的角度看,MAS 的故障主要体现在两个方面。一个方面是 agent 本身的故障,这是一种局部故障,可以通过对 agent 本身的诊断来发现并恢复;另一方面是 agent 之间的故障,可以看作全局故障,其特点是:agent 本身运行状况良好,未出现任何异常情况,但在整个 MAS 运行过程中由于多个 agent 之间相互作用、相互协同,也有可能出现错误或异常,比如由于网络原因造成主体间通信消息丢失等。这种故障对于 MAS 来说也是致命的。因此,MAS 的自诊断功能就显得至关重要,这也是目前多主体系统亟待解决的关键问题之一。

2 多主体平台 MAGE

多主体平台 MAGE 是我们建立的一个面向主体的软件开发、集成和运行环境[11],主要基于智能主体和多主体技术,为用户提供一种面向主体的软件开发和系统集成模式,包括面向主体的需求分析、系统设计、主体生成以及系统实现等多个阶段。MAGE 提供了面向主体的软件开发模式,以主体为最小粒度,封装和自动化实现了主体的一般性质,应用程序可以通过添加特殊的主体行为方便地实现自己的特定功能。它还提供了多种软件重用模式,可以方便地重用以不同语言编写的主体或非主体软件,进行各种应用集成。

MAGE 主体平台主要包括 4 个模块:主体管理系统 AMS、目录服务主体 DF、一般主体以及消息传输系统 MTS,此外,还有两个辅助的模块为主体系统开发提供方便,即主体库和功能构件库。相应地,MAGE 主体平台具有以下 6 个功能:

- 主体系统管理:主要对主体平台中的主体进行基本的管理功能,包括生命周期管理;
- 主体目录服务管理:负责对主体提供的服务进行目录管理;
- 主体消息传输:主要负责平台之间和主体之间的消息传输管理,即通信管理;
- 主体库管理:负责由各种主体类型组成的主体库的管理,包括添加、生成、删除等;
- 主体功能构件管理:主要针对不同功能的构件的管理,可以组装出不同类型的主体;
- 主体软件集成:主要负责对各种不同类型的软件进行主体封装,集成为多主体系统。

MAGE 主体平台的功能特点包括分布式计算平台,多种软件重用及多种主体生成方案。

MAGE 系统内部采用内部消息传输协议(IMTP)和消息分发相结合的消息传输方式,主体之间通信传输的内容是 FIPA-ACL 格式的消息。它完全屏蔽了底层实现细节,发送主体只要设置好消息中各个槽的值,然后发送即可,而接收主体也只需要简单地从消息队列中取出相应消息。MAGE 还提供了多种不同的消息接收模式,提供了消息模板匹配供主体使用。

综上所述,在 MAGE 这一多主体平台上开发的应用程序有着不同于一般软件的特点,比如分布性、主体性、消息通信机制等。这些特性使得对于该类程序的诊断,有一些特殊的要求。我们的诊断方法正是针对这样一类特殊的基于多主体系统的应用程序来设计实现的,既能满足这些要求,也利用了这些特性来提高诊断效率。而且由于 MAGE 能为各种信息处理系统、管理系统、决策系统、智能控制系统、实时系统等提供很好的构建框架和应用模式,在电子商务、电子政务、集成制造、工业控制、智能决策、智能交通、军事指挥等众多领域中具有广泛的应用前景,因此我们的诊断方法在基于多主体系统的应用软件范围内有着很好的通用性。只要是按照多主体架构来构建的应用程序,均可采用此方法进行故障诊断。

3 基于主体行为模型的诊断方法

传统的基于模型的诊断方法,其主要思想是:根据组成系统的构件之间的关系建立待诊断系统的模型(如系统的内部结构、构件的功能、行为等),这种模型通常用一阶逻辑语句(如谓词符号、框架、约束和规则等)来描述,根据系统的逻辑模型以及系统的输入,能够通过逻辑推理或演算推导出系统在正常情况下的预期行为,如果观测到的系统实际行为与系统预期行为有差异,则说明系统存在故障,利用逻辑推理确定引发故障的构件集合。

而在基于 MAS 的应用程序中,主体是其中的基本要素,在搭建具体应用时,是按照面向主体的设计思想划分系统中不同的角色以及它们之间的交互关系,然后生成相应的主体,封装各自的具体行为,提供所需服务。因此,我们对于应用程序的模型的构造,重点放在对于主体的行为建模上。Reiter 等人提出的经典方法只能处理静态模型,不适用于描述多主体构成的动态系统,而我们的方法,其本质是基于

主体状态变迁,从而适合描述动态系统。

3.1 构建系统功能模型

方法的第一步是,采用我们设计的主体统一建模(AUMP)语言,构建多主体应用系统的功能模型。AUMP 是一种基于用例思想的面向主体的可视化建模语言,它包含了系统边界、系统中各类主体、主体的目标(对应系统要实现的功能)以及主体的行为(对应如何实现目标的任务,即如何完成用例)等元素(用特定符号表示),然后将这些元素通过有特殊含义的连线连接起来,表示相互间的作用和关系。仿真实验中构建的功能模型实例如图 1 所示,其中矩形代表系统的边界,小人代表应用系统中的主体,椭圆代表主体的目标,三角则代表实现目标的任务。功能模型使得用户在对主体行为进行建模之前,首先从宏观上把握主体间的关系及系统总体结构,避免陷入主体局部的细节,而无法对系统整体功能有一个全面正确的认识,该步骤尤其对于复杂系统,是一个必不可少的环节。

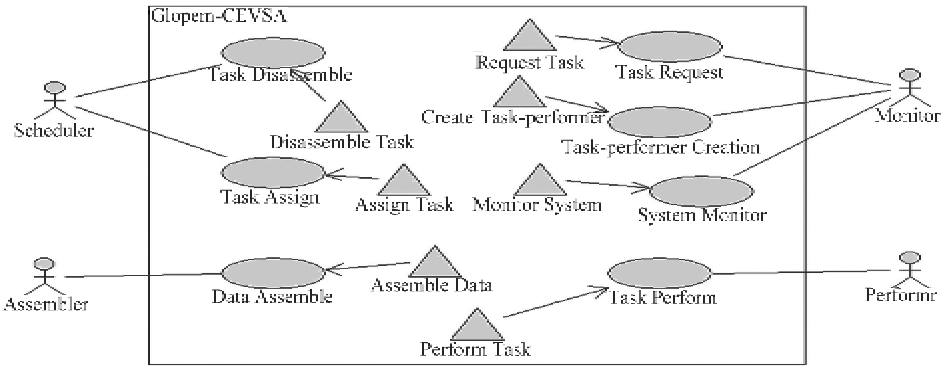

图 1 多主体系统(MAS)功能模型

3.2 构建主体行为模型

在系统功能模型的基础上,我们可以进一步构建主体的行为模型,其基本思想是:把整个待诊断系统按照 agent 进行划分,不需要详细考虑各个 agent 的结构,只考虑 agent 的行为及其相互作用。如果用图的节点表示各 agent 所能产生的状态,用状态间的弧表示 agent 的行为,即激发 agent 状态变迁的事件,或者变迁过程及结果产生的可供观察和记录的事件,则对这些事件及信息的描述就构成了 agent 的行为模型,通过分析,我们认为 agent 的行为可以用 3 类事件来表征:

- 外部事件(exogenous event),即软件使用者的某些操作,如点击按钮;
- 通信事件(communication event),即 agent 之间的通信消息;
- 观察事件(observable event),即使用者可以观察到的系统状态的改变,也表明了 agent 的某种状态产生了变化,如某个计算结果在文本框中的显示。

其中,外部事件和通信事件是触发 agent 状态变迁的事件,通信事件和观测事件是 agent 状态变迁产生的事件。在离线构建好各 agent 的行为模型后,系统实际运行时,由监控程序捕获上述类型的信息,即对于每个 agent 都会得到一个可以观察到的事件序列,将这些信息记录下来,保存在各 agent 的日志文件里。诊断时将日志文件与行为模型进行比较,如果两者存在不一致,说明该 agent 发生了故障,然后通过获取系统信息进一步推理确定故障类型。

需要说明的是,agent 行为模型包括了对其正常行为和故障行为的描述。我们初步只考虑了相对简单的情况,将故障行为分为系统级故障(即全局故障,包括资源依赖故障和运行故障,如网络连接、服务器、存储设备故障等)和 agent 级故障(即局部故障,包括崩溃性故障和遗漏性故障,如发送故障、接收故障,响应故障等)。

图 2 和图 3 是对 agent 的行为建模示例,其中图 2 是在我们开发的可视化建模工具 FSMEditor 中利用有限状态自动机(finite state machine, FSM)构建的 agent 行为模型,图 3 是由 FSMEditor 自动转化的对该行为模型的动态描述逻辑(dynamic description logic, DDL)描述。用 FSM 构建行为模型的优点是直观易读,易于理解和分析主体行为,当系统结构或主体行为发生变化时易于对模型进行修正。底层转化为 DDL 描述便于算法的表示,比较和推理。

通过对行为模型的构建,各 agent 都拥有了 DDL 语言描述的动作集,下一步,通过我们设计的基于一致性诊断算法,就可以在动作集表示的行为模型基础之上进行故障诊断。在此,我们假定软件系统中,所有与通信、观测事件相关的消息,以及大部分与外部事件相关的消息,都是可获取的,并且获取的消息是可信的。

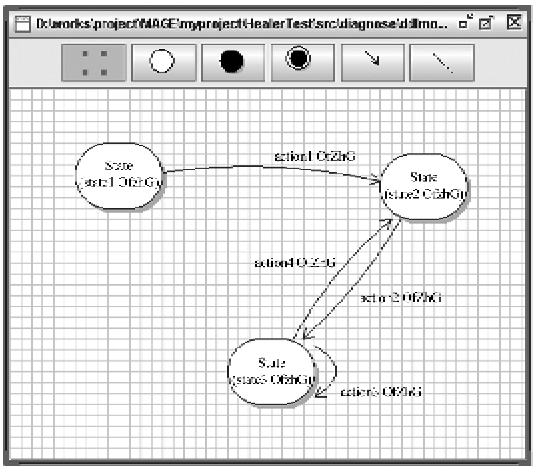

图 2 Agent FSM 模型

```
Agent(ZhG), Agent(QuSH)
State(state1OfZhG), State(state2OfZhG), State(state3OfZhG)
ObserEvent(calButtonAction), ExogeEvent(completeComput)
ComEvent(INFORM-REF), ObserEvent(resultsetText)
ComEvent(REQUEST), ComEvent(INFORM)
Action(action1OfZhG), Action(action2OfZhG)
Action(action3OfZhG), Action(action4OfZhG)
action1OfZhG = <{currentState(ZhG, state1OfZhG),
                 hasObserEvent(ZhG, calButtonAction),
                 currentState(ZhG, state2OfZhG),
                 NEG(currentState(ZhG, state1OfZhG))} >
action2OfZhG = <{currentState(ZhG, state2OfZhG),
                 hasExogeEvent(ZhG, completeComput)},
                 currentState(ZhG, state3OfZhG),
                 NEG(currentState(ZhG, state2OfZhG)),
                 comEventTo(INFORM-REF, QuSH),
                 hasObserEvent(ZhG, resultsetText)} >
action3OfZhG = <{currentState(ZhG, state3OfZhG),
                 comEventFrom(REQUEST, QuSH)},
                 {comEventTo(INFORM, QuSH)} >
action4OfZhG = <{currentState(ZhG, state3OfZhG),
                 hasObserEvent(ZhG, state2OfZhG)},
                 currentState(ZhG, state2OfZhG),
                 NEG(currentState(ZhG, state3OfZhG))} >
```

图 3 Agent 模型 DDL 描述

```
FautDiag (AgentAct, State, EventSeq, RecStack,
          NewState, RemEventSeq, NewRecStack, FaultType, FaultInfo)

Inputs: AgentAct:待诊断 Agent 的动作集,其中包含该 Agent 的 ID 信息;
        State: EventSeq 消息序列出现前 Agent 所处的状态;
        EventSeq: Agent 消息事件公式偏序集;
        RecStack: FaultDiag 执行前诊断过程恢复信息堆栈;
Outputs: NewState:EventSeq 消息序列出现后 Agent 所处的状态;
         RemEventSeq: FaultDiag 执行后剩余的消息公式偏序集
         RewRecStack: FaultDiag 执行后诊断过程恢复信息堆栈;
         FaultType: 故障类型,0-无故障,1-应产生事件尚未出现,2-外部事件引发的故障;
         FaultInfo: 故障信息,如故障原因
Begin
    //EventSeq 与动作集匹配完成,且当前不处于故障状态
    If (RemEventSeq = = null && NewState 为非故障状态){
        FaultType = 0;
        FaultInfo = "no fault";
    }
    //RemEventSeq 不够匹配下一动作,存在变迁产生事件未出现
    else if (FindAct(AgentAct, NewState, RemEventSeq) = = 2){
        FaultType = 1;
        FaultInfo = "messages produced by latest action not show up"
    }
//RemEventSeq 非空,且匹配下一动作失败
    else if (RemEventSeq 1 = null =
            && FindAct(AgentAct, NewState, RemEventSeq) = = 0){
        Recover(NewRecStack, State, EventSeq);
        //递归调用 FaultDiag
        FaultDiag(AgentAct, State, EventSeq, RecStack,
                NewState, RemEventSeq, NewRecStack, FaultType, FaultInfo);
    }
    //RemEventSeq 可继续匹配下一个或多个动作
    else{
        If (FindAct(AgentAct, NewState, RemEventSeq) = = 1)
            Push(GetActsRes(AgentAct, NewState, RemEventSeq), NewRecStack);
        Recover(NewRecStack, State, EventSeq);
        RecStack = Pop(NewRecStack);
        //递归调用 FaultDiag
        FaultDiag(AgentAct, State, EventSeq, RecStack,
                NewState, RemEventSeq, NewRecStack, FaultType, FaultInfo);
    }
End
```

图 4 基于主体行为模型的诊断算法

算法首先设置每个 agent 的初始状态 InitialState,然后获取每个 agent 的输入、输出、通信(包括发送和接收)消息序列,将获取到的消息偏序集转换为由 ExogeEvent, ObserEvent, ReceiveEvent 和 SendEvent 公式组成的 DDL 公式偏序集 EventSeq,具体诊断过程见图 4。

算法中匹配的定义为:按照时间非递减顺序在该 agent 的消息公式偏序集中选出一个触发事件(ExogeEvent 或 ReceiveEvent)公式,加上当前状态公式,如果使得该 agent 动作集中某动作的前提公式集满足,则按时间非递减顺序检索消息公式偏序集,若该动作结果公式集也满足,则表示匹配;如果该 agent 动作集中不存在前提公式集满足的动作,则找出动作集中所有以该 agent 当前状态公式和 ExogeEvent 公式为前提的动作,按时间非递减顺序检索消息公式偏序集,若该动作结果公式集满足,则表示匹配。

4 故障诊断系统 eHealer 及其仿真实例

在 FaultDiag 算法基础上,我们构造了基于 MAGE 平台的故障诊断系统 eHealer。该系统主要有监视器模块、执行器模块两大功能模块,其系统结构如图 5 所示。监视器模块,即对 MAGE 平台上运行的应用程序中各 agent 的行为进行监控,记录系统实际运行时 agent 的行为事件,包括正常行为和故障行为,并在系统发生故障时对其进行诊断;执行器模块,根据诊断结果,由决策器选择恢复策略,对故障 agent 进行修复,从而使系统恢复正常运行。

之所以将 eHealer 作为一个单独的监控系统,独立于应用程序而存在,是因为由于多主体系统(MAS)中每个 agent 都是相对独立和自主的,不存在能够控制各个 agent 的第三方,如果不设定一个单独的监控系统,那么当 agent 出现故障时,只能由某个应用程序 agent 来诊断,而且在诊断的过程中必然需要其他 agent 的协作。也就是说,agent 不但是对软件实体的抽象,同时还要具有故障诊断的功能。这就对应用程序的结构耦合性提出了更高的要求。而且,在分布式 MAS 环境下如果要求 agent 具有故障诊断功能,也存在较大的困难,一是每个 agent 拥有的知识一般是不完全的,二是在大规模 MAS 中,系统的组织结构及 agent 的协作方式都会严重影响计算效率,构造高效的组织结构和有效的协作方式并非一件容易之事。

另一方面,在图 5 中可以看到,eHealer 与 MAGE 平台也存在一定程度的交互,这是因为在 FaultDiag 算法中,应用程序的行为描述主要针对其正常行为,而对其故障行为仅限于较简单的情况,所以算法中的故障诊断主要采用基于模型的方法。而更为具体的故障信息(如具体故障类型)需要结合 MAGE 平台提供的系统消息以及故障库进行进一步推理和诊断,因此还需要采用基于知识的方法,在 eHealer 的系统结构中,加入和 MAGE 平台之间的交互部分以及故障库推理查询部分,从而获取更为完全的系统状态的知识。

图 6 是实际运行中的 eHealer 仿真平台,方框表示了系统中的主体,对于每个主体,都以不同颜色的

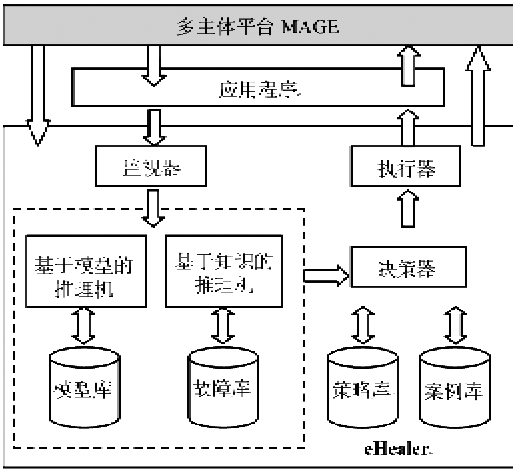

图 5 eHealer 系统结构

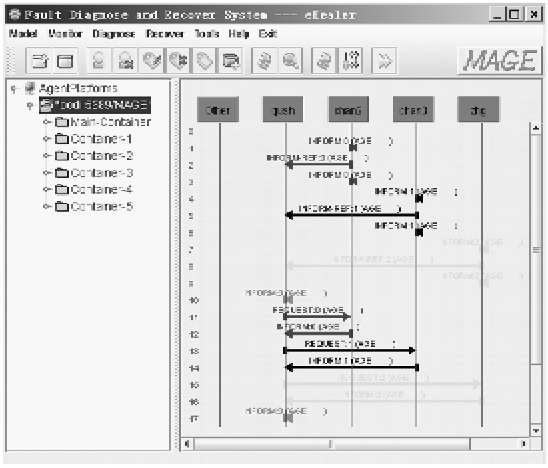

图 6 eHealer 仿真界面

箭头记录了其外部事件、通信事件和观察事件。该仿真平台在与电力研究院合作开展的《配电网故障恢复决策系统》项目中,使用MAGE平台构建的分布式电力智能决策系统上进行了实验,结果表明eHealer可诊断出80%的agent级别的故障并确定故障类型。

目前我们的工作主要集中在监视器模块部分。下一步会更多地考虑执行器模块部分,通过机器学习等手段来加强恢复策略的选择功能。

5 相关工作比较

与以往的诊断系统相比,eHealer具有如下的优点:

(1) 可以与应用程序同步运行,实时监控,对应用程序透明;

(2) 能够对系统从宏观上加以控制,可以诊断大型随时间变化的动态系统;

(3) 有很强的通用性和独立性,方法不针对具体系统,没有对具体应用程序所涉及领域的依赖性,将系统的推理内核与系统模型分开,对一个对象适用的诊断方法同样可以用在另一个诊断对象上,只要构造出新的应用程序的模型,就可以实施诊断,具有普适性和良好的可推广性;

(4) 主体行为模型可以从应用程序开始设计或开发时就开始构造,减少了获取环节,也避免了从领域专家获取知识的困难;

(5) 不需要关于应用程序的具体知识,简化了知识获取过程,缩短了诊断系统研制周期,提高了诊断效率,也避免了对诊断对象知识的不完备造成的漏诊误诊;

(6) 只要模型抽象的正确,就能诊断所有可能的故障,不但能处理新故障,还能给出令人信服的解释。

但是,eHealer目前也存在自身的缺陷,它对应用系统有一些特定要求,比如,应用系统需要是运行在MAGE平台上的MAS,需要应用系统有相对健全的消息机制,agent间交互全部通过消息发送和接收来完成,且对系统的行为有相对完整的日志记录,对MAGE平台本身的健壮性有较高要求,需要对被诊断系统有一个宏观的认识,建立准确的系统行为模型,对于复杂系统相对困难,对系统的描述粒度较粗,等等。这些在下一步工作中有待改进。

6 结 论

故障诊断技术经过几十年的发展,已经开发和研究出比较成熟的方法和理论,如专家系统、神经网络、模式识别、模糊推理等。然而复杂的故障往往需要一种或多种诊断方法协同工作才能得出结论。因此我们一方面设计了特定于多主体平台MAGE上的应用系统诊断算法FaultDiag,另一方面在此基础上研究了基于两层架构的综合故障诊断系统eHealer。第一层以基于模型诊断算法FaultDiag为核心,由诊断对象模型组成,分别描述了诊断对象正常工作时的正常行为模型和故障时的故障行为模型,它们被用于基于模型的诊断。第二层描述了一般的诊断知识,由根据诊断对象的领域知识和专家诊断经验归纳出的诊断规则组成,用于基于知识的诊断。实验表明,eHealer系统可以准确定位故障agent及故障类型。

下一步,在功能层次上,一方面要使eHealer具备更强大的诊断能力,比如诊断复杂故障,有效发现并及时阻止故障传播,实现分层分步骤诊断,以及分布式诊断,等等;另一方面要扩展eHealer的功能,建立好的恢复策略。在应用层次上,还要将eHealer推广到对网构软件的诊断,因为主体具有自治性、交互性、协作性、可通信性、自适应性等特点,是网构软件基本单元的最好实体。一个网构软件系统可以抽象为一个MAS,其中agent就是软件实体的抽象。因此,基于agent的诊断技术应当也是对网构软件系统进行诊断的最为理想和有效的方法。更进一步,还可以推广到对Web上的服务的诊断,一个服务可以看作一个agent,服务间的信息通信可以看作消息,因此可将诊断方法移植到Web服务的故障诊断中,使eHealer系统得到更广泛的应用。

参考文献

[1] 韩旭,史忠植,林芬. 基于模型的诊断研究进展. 高技术通讯,2009, 19(5):543-550

[2] de Kleer J, Williams B C. Diagnosing multiple faults. *Artificial Intelligence*, 1987, 32: 97-130

[3] de Kleer J, Williams B C. Diagnosis with behavioral models. In: Proceedings of the 11th International Joint Conference on Artificial Intelligence, Detroit, USA, 1989. 1324-1330

[4] Reiter R. A theory of diagnosis from first principles. Artificial Intelligence, 1987, 32: 57-95

[5] Pencolé Y, Cordier M-O. A formal framework for the

decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 2005, 164(1-2): 121-170

[6] Cordier M-O, Grastien A. Exploiting independence in a decentralised and incremental approach of diagnosis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India: AAAI Press, 2007. 292-297

[7] Kalech M, Kaminka G A, Meisels A, et al. Diagnosis of multi-robot coordination failures using distributed CSP algorithms. In: Proceedings of the 21st National Conference on Artificial Intelligence, Boston, Massachusetts, USA: AAAI Press, 2006. 970-975

[8] Grastien A, Anbulagan, Rintanen J, et al. Diagnosis of discrete-event systems using satisfiability algorithms. In: Proceedings of the 22nd National Conference on Artificial Intelligence, Vancouver, British Columbia, Canada: AAAI Press, 2007. 305-310

[9] 吕建,马晓星,陶先平等. 网构软件的研究与进展.中国科学 E 辑(信息科学),2006,36(10):1037-1080

[10] 史忠植. 高级人工智能. 第 2 版. 北京:科学出版社, 2006

[11] Shi Z Z, Zhang H J, Cheng Y, et al. MAGE: an agent-oriented software engineering environment. In: Proceedings of the 3rd IEEE International Conference on Cognitive Informatics, Victoria, Canada, 2004. 250-257

eHealer: a fault diagnosis system for agent-based software

Zhang Donglei[*,**], Han Xu[*,**], Shi Zhongzhi[*]

([*] The Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] Graduate University of Chinese Academy of Sciences, Beijing 100049)

Abstract

The paper proposes a method that can diagnose faulty agents in multi-agent systems (MAS) by monitoring communication messages among agents. For the application software running on a MAS platform, the method is descnbed as below: firstly, establishing its function model, which helps to construct its behavior model, secondly, catching the communication messages among agents and other events during its practical running process, thirdly, comparing the message sequence with the behavior model by the consistency-based algorithm and reasoning to find the faulty agent and its faulty behavior. On this basis, eHealer, prototype system for sault diagnosis, is realized. The experiment shows that this method can locate those faults on agent level in MAS software. It has real-time response, accurate location and strong adaptation, which provide effective information for selection of recovery strategies.

Key words: fault diagnosis, multi-agent systems (MAS), system function model, agent behavior model, comparison based algorithm