第34卷 第23期 计 算 机 工 程 2008年12月
Vol.34 No.23 Computer Engineering December 2008
·软件技术与数据库· 文章编号：1000—3428(2008)23—0038—03 文献标识码：A 中图分类号：TP311

面向目标的迁移工作流研究

郑 楠，韩芳溪，曾广周
(山东大学计算机科学与技术学院，济南 250061)

摘 要：传统的迁移工作流系统为迁移实例编写面向过程的工作流说明并驱动其工作，这种方法限制了迁移实例的灵活性。该文提出面向目标的迁移工作流的概念，定义相应的概念模型，给出基于BDI体系的迁移实例结构，使得迁移实例可以产生令工作流状态向目标转移的信念、愿望、意图和行为。相比传统的工作流方法，该概念模型和迁移机制能够灵活地处理工作流环境中的变化和根据工作流状态进行智能决策，具有目标驱动的特性。

关键词：工作流管理；迁移工作流；迁移实例；BDI体系

Study of Target-oriented Migrating Workflow

ZHENG Nan, HAN Fang-xi, ZENG Guang-zhou
(School of Computer Science & Technology, Shandong University, Jinan 250061)

【Abstract】Traditional migrating workflow systems make procedural specifications drive their instances to work, which greatly weakens the agility of instances. To solve this problem, a conception of target-oriented migrationg workflow and its conceptual model are proposed in this paper. It proposes a mechanism based on the BDI-architecture. The migrating instances can generate beliefs, desires, intentions and actions to lead the workflow state to the ultimate purpose. Compared with the traditional workflow methods, the model is flexible and intelligent. It is target-driven and capable of dealing with changes and making decisions according to the workflow state.

【Key words】workflow management; migrating workflow; migrating instance; BDI architecture

1 概述

近年来研究者把移动agent技术引入到工作流管理系统的研究[1]中。以移动agent为范型构造的迁移工作流实例(简称迁移实例，下同)可以在不同站点之间迁移，并应用当地的服务执行任务。这类研究一般事先编写面向过程的工作流说明并驱动迁移实例工作。这种方法智能性较差，不利于处理例外事件，对于工作流环境的变化缺乏很好的适应性。

针对上述问题，本文提出了面向目标的迁移工作流(TOMWF)的概念，并参照文献[2]的方法，给出了一个概念模型，进而基于BDI体系给出一种面向目标的迁移实例构造方法。BDI体系由Rao和Georgeff等人[3-5]提出并完善，是一个成熟的并广泛被采用的agent结构。它通过信念(B)、愿望(D)、意图(I)3个主要的心智状态来表示agent的推理模型和推理策略。该模型直观，符合人的逻辑思维，能够适应变化的环境，在许多实时系统和PRS系统中得到成功应用[4]。通过引入BDI体系，能够提高工作流系统的智能性以及在变化的环境下的适应性。

本文的主要目的是结合传统迁移工作流和BDI体系理论的相关知识，提出一种面向目标迁移工作流研究的思路，以期对这一方向的进一步研究提供帮助和启发。

2 面向目标的问题描述

面向目标的迁移工作流方法是指给定工作流的初始状态、目标状态和工作流服务的上下文环境，建立一种目标驱动的迁移工作流机制。所谓目标驱动，即能够使迁移实例根据对工作流目标和环境的感知，建立对环境的信念，产生改变工作流状态的愿望和意图，并转化为行为，如图1所示。

与以往指定迁移实例按照工作流说明来完成业务过程相比，面向目标的迁移工作流只需要告知迁移实例需要达到的目标，迁移实例根据自身的当前状态与目标状态的差距，自行决定执行的细节。

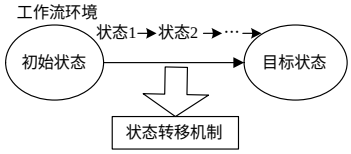

图1 面向目标的工作流方法

下面举一个采购的例子，假设某公司分配给采购员M的目标是ABCD 4种零件。每种零件的产地有多个，且价格因产地而不同。M的初始状态和目标状态可以分别表示为0000(全部没买)和1111(全部买到)。M对于采购路线和采购方案的选择将导致不同的状态进化路线。面向目标的思想即公司不限定采购零件的路线，只是要求M必须达到1111的最终状态，并尽量减少路费和采购成本。这决定了M可以综合考虑状态进化的路线，选择优化的方案。当情况发生变化时(例如某产地不再生产A，或者提高了B的价格)，M还可以及时调整自己的采购策略。

3 面向目标迁移工作流的概念模型

面向目标迁移工作流(TOMWF)的定义以文献[2]的概念模型为基础，但在TOMWF中，工作流根据需要划分成多个目标，每个目标由一个迁移实例完成。

基金项目：国家自然科学基金资助项目(60573169)
作者简介：郑 楠(1984－)，男，硕士研究生，主研方向：网络与分布式技术；韩芳溪，副教授；曾广周，教授、博士生导师
收稿日期：2008-03-08 **E-mail：**prosnan@gmail.com

定义 1　TOMWF 是一个四元组(*Wid, MI, WP, Engine*)，其中，

Wid 是迁移工作流标识；

$MI=\{mi_1,mi_2,\cdots,mi_n\}$是迁移实例(简称*mi*，下同)的集合，每个$mi \in MI$对应一个独立的目标；

$WP=\{wp_1,wp_2,\cdots,wp_m\}$是工作位置集合；

Engine 是工作流引擎。

一般来说，一个 *mi* 的目标又可以分解为若干个按一定逻辑关系组合的子目标。因此，*mi* 的执行状态，可以用子目标的完成情况来表示。

定义 2　设一个*mi*的目标为*T*，*T*可以分解为*k*个子目标。令谓词$A_i(x)$表示在环境输入为*x*的情况下目标*T*的第*i*个子目标的完成情况。定义$S_t=\{A_1(x_t), A_2(x_t),\cdots,A_k(x_t)\}$(x_t表示*t*时刻的环境)是*mi*在*t*时刻的状态。显然，如果S_t中所有子句取值为 1，即所有$A_i(x_t)$ (i=1,2,⋯,k)为真，则可知目标*T*已经完成。

mi 根据功能划分为决策部件和执行部件 2 部分。*mi* 的决策部件 *d-agent* 由工作流引擎或指定的工作位置创建，并停留在创建位置，用于感知工作流环境，建立信念，并根据自身状态与目标状态的差异产生改变状态的愿望和意图。*mi* 的执行部件 *e-agent* 在 *d-agent* 的指导下进行迁移，并在工作位置上应用工作流服务；促使 *mi* 的状态向着目标状态不断转移。

定义 3　迁移实例 $mi \in MI$ 是一个六元组(*miid, ToL, MP, Sg, Sc, TM*)，其中，

miid 为可认证的迁移实例标识；

ToL 是 *mi* 的生命周期；

$MP \subseteq WP$ 为允许 *mi* 迁移的工作位置集合；

Sg 和 *Sc* 分别是 *mi* 的目标状态和当前状态；

TM={*d-agent, e-agent*}是目标驱动的状态转移机制，其中，*d-agent* 是 *mi* 的决策部件。*e-agent* 是 *mi* 的执行部件。

定义 4　*d-agent* 是一个八元组(*d-aid, Bel, Des, Int, brf, options, filter, DC*)，其中，

d-aid 是 *d-agent* 的标识；

*Bel, Des, Int*是*d-agent*为*mi*构建的信念、愿望和意图，使用基于时间树的可能世界结构[3]来表达；

Brf, options, filter 分别是定义在 *Bel, Des, Int* 上的信念修正函数、选择生成函数和过滤函数。各函数功能将在第 4 节描述。

DC 是的 *d-agent* 的工作机，包括信息的输入输出、服务发现选择、迁移决策、通信和协作、安全保护等。

定义 5 *e-agent* 是一个七元组(*e-aid, S-current, S-next, MP', p, EQ, EC*)，其中，

e-aid 为 *e-agent* 的标识；

S-current 代表 *mi* 的当前状态，即 *Sc*；

S-next 是由 *d-agent* 决定实现的下一步状态；

$MP' \subseteq MP$ 为允许 *e-agent* 迁移的工作位置集合；

p 为 *e-agent* 当前所处的工作位置，$p \in MP'$；

EQ 是 *e-agent* 的事件队列，接收来自 *d-agent* 的决策事件和工作位置的局部事件；

EC 为 *e-agent* 的工作机，负责工作流服务的应用和执行、迁移控制、信息收集和反馈、安全保护等。

工作位置定义与文献[2]类似，它为 *mi* 提供服务和资源。当工作位置无法满足 *mi* 工作需要的时候，*mi* 迁移到下一个合适的工作位置继续未完成的工作。

4　BDI 结构和迁移决策

在本文研究中，*mi* 通过决策部件 *d-agent*，基于 BDI 构建出对于工作流环境的信念、愿望和意图，选择从当前状态到目标状态的转换的优化方案，同时，环境的变化也可以通过在 BDI 上的更新得到响应并影响 *mi* 的行为。

参考Rao和Georgeff的BDI理论[3]以及石纯一等人[4]对此理论的完善和拓展，*d-agent*使用基于时间树的可能世界结构来表示*mi*在某一个时刻所处的情景(即包含工作流环境，*mi* 状态等多种信息的时间点)及拥有的选择。

可能世界对应一个时间树结构，其中，每个时间点所发散的路径代表了*mi*在当前情景下所具有的选择。如图 2 所示，*mi*在时刻t_0，其状态为S_{t0}，工作位置为wp_1，*mi*面临多个选择：可以执行wp_1所提供的多个服务中的一个，或者迁移到其他工作位置wp_2, wp_3中的一个。这些信息，是*d-agent*对环境的认知，即信念。信念修正函数*brf*收集环境信息构建*d-agent*的信念*Bel*，还接收环境变化的事件和来自执行部件的反馈，更新信念，以便*d-agent*的决策能够适应环境的变化。

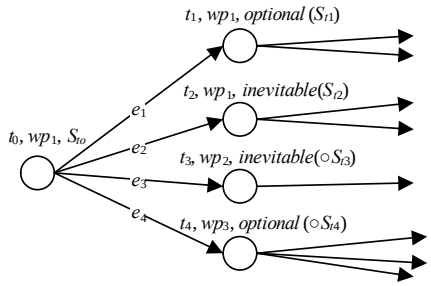

图 2　可能世界结构

在图 2 中，事件：e_1, e_2 分别为执行wp_1上的 2 种不同服务；e_3, e_4分别为迁移到工作位置wp_2, wp_3。模态操作符：*optional*表示有可能；*inevitable*表示必然。时态操作符：○- *next* 表示下一步。

与OASIS系统中对于飞行路线建模[4]类似，*d-agent*计算*Bel*结构中最终的状态与*mi*的目标状态*Sg*相一致的路径，剪除不符合条件的路径以及和当前意图不一致的信念，形成愿望世界*Des*，这个过程对应选择生成函数*options*。

最后，d-agent 考虑愿望世界中路径的性能。这包括迁移或者执行服务等行为对于总体目标的影响(即行为前后状态之间的差异)，以及该行为所消耗的资源和时间等代价。剪除耗费巨大的路径，则形成意图世界 *Int*，这个过程对应过滤函数 *filter*。过滤函数同时还要完成几个任务，包括放弃不能实现的意图，保存还没有完成的意图和接受新的意图。

假设 *d-agent* 接收的环境信息和来自执行部件的反馈信息为 *Info*, *select* 函数可以将当前意图映射为决策事件 *decision*，则上述决策过程可以描述为函数 *decide*:

```
function decide (i : Info) : decision
begin
    Bel    := brf (Bel, i)
    Des    := options (Bel, Int)
    Int    := filter (Bel, Des, Int)
    return select (Int)
end function decide
```

5　系统框架和工作位置

本研究沿用文献[2]中的迁移工作流系统框架，它是由一个执行工作流引擎的管理机和若干个停靠站服务器组成的网络。其中，停靠站服务器是 *mi* 的工作位置，它接受 *mi* 的查

询和迁移请求，并在 *mi* 到达后为其提供工作流服务和资源服务。与文献[2]不同，由于 *d-agent* 的位置固定，因此工作位置可以主动与 *mi* 通信。通信内容包括工作位置的服务能力以及对应的耗费和代价等信息。当服务能力发生变化时，产生环境事件发送给 *d-agent*，以便后者更新 BDI 结构。

此外，工作位置还可以提供目标解析的功能，即：工作位置获得 *e-agent* 的 *S-current* 和 *S-next* 并进行解析，比较两个状态间的差异，得到实现状态转移所需要完成的子目标集合。分析实现这些目标的逻辑关系，提供所需要的服务以及对应的资源和工序。

6 工作流实例构造

按照定义 3，*mi* 具有位置固定的决策部件和在工作位置间迁移的执行部件。它们之间的分离协作关系如图 3 所示。这种结构既保证了目标驱动的特性，同时又减小了迁移规模。位置固定的决策部件也为迁移实例之间以及迁移实例与工作位置间的通信提供了方便。

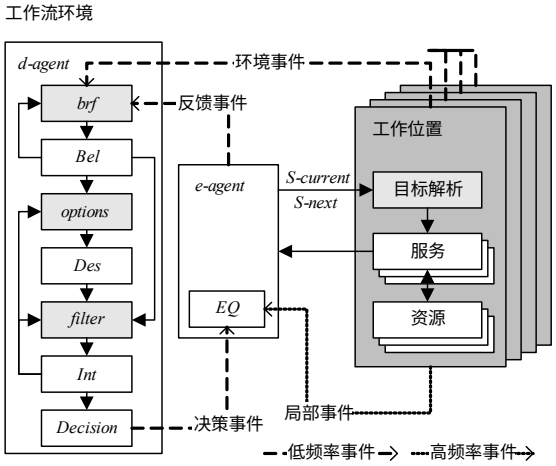

图 3　迁移实例的分离协作模型

基于第 5 节系统框架和图 3 的迁移实例结构，*mi* 的生命周期过程大致如下:

(1)工作流引擎为一个工作流目标创建迁移实例 *mi* 和其决策部件 *d-agent* 以及执行部件 *e-agent*。

(2)*mi* 成功创建后，创建者向所有工作位置发送消息并公布 *mi* 的服务和资源需求。接收到消息的工作位置记录下其 *d-agent* 的地址，并将自身的服务能力告知 *d-agent*。此后工作位置仅在 *mi* 发出查询或迁入请求以及自身服务能力变化时与 *mi* 进行通信。

(3)对于任一个 *mi*：

1)*d-agent* 收集工作位置的信息，构建 BDI 结构，作出有利于向目标状态转移的决策，以事件的形式发送到 *e-agent* 的事件队列，指导 *e-agent* 的行为。

2)*e-agent* 根据决策事件更新 *S-current* 和 *S-next*，工作位置对此进行目标解析，提供对应的资源和服务。*e-agent* 的事件队列也用来接收工作位置上的局部事件，以便对执行的细节进行控制和管理。当服务执行完成(成功或者失败)后，*e-agent* 向 *d-agent* 进行反馈，等待下个决策事件。

3)当 *d-agent* 发现需要迁移时，选择最合适的目的位置，发出迁移请求。目的位置收到请求后作出应答。当 *d-agent* 接收到允许应答时，向 *e-agent* 发送命令，*e-agent* 挂起正在进行的任务，进行迁移。新位置在 *e-agent* 到来时对其注册，激活挂起的任务。原工作位置将 *e-agent* 注销，释放服务资源。

4)当 *mi* 状态达到目标状态时表示迁移实例完成了所有任务。

(4)*mi* 达到目标状态后，创建者销毁或者回收 *d-agent* 以及 *e-agent*，*mi* 生命周期结束。创建者通知所有工作位置释放为 *mi* 作准备而占用的资源。

7 结束语

本文在传统的迁移工作流的模型定义的基础上，结合 BDI 相关理论，给出了一个基于 BDI 体系的面向目标迁移工作流实例构造方法。这种构造方法能够给迁移工作流系统带来许多新的特性，比如目标驱动的智能决策和对于变化的灵活响应等。但是由于面向目标的迁移工作流还是一个很新的概念，需要更多的研究和完善。

参考文献

[1] Cichocki A, Rusinkiewicz M. Migrating Workflows[C]//Proc. of Workflow Management Systems and Interoperability. Berlin, Germany: [s. n.], 1998: 339-355.

[2] 曾广周, 党　研. 基于移动计算范型的迁移工作流研究[J]. 计算机学报, 2003, 26(10): 1343-1349.

[3] Rao A S, Georgeff M P. Modeling Rational Agents within a BDI-architecture[C]//Proc. of Principles of Knowledge Representation and Reasoning. San Mateo, France: [s. n.], 1991: 473-484.

[4] Rao A S, Georgeff M P. BDI Agents: From Theory to Practice[C]//Proc. of the 1st Int'1 Conf. on Multi-agent Systems. San Francisco, France: [s. n.], 1995: 312-319.

[5] 程显毅, 石纯一. 避免逻辑全知的 BDI 语义[J]. 软件学报, 2002, 13(5): 966-971.

（上接第 30 页）

参考文献

[1] Taura K, Chien A. A Heuristic Algorithm for Mapping Communicating Tasks on Heterogeneous Resources[C]//Proc. of the 9th Heterogeneous Computing Workshop. Washington D. C., USA: IEEE Computer Society, 2000-05: 102-115.

[2] 乔伟光, 曾国荪. 一种基于分簇复制的 DAG 任务图调度算法[J]. 计算机工程, 2006, 32(17): 126-128.

[3] 蒋韵联, 孙广中, 许胤龙. 并行异构系统中的一种高效任务调度算法[J]. 计算机工程, 2007, 33(11): 39-41.

[4] 刘振英, 方滨兴, 张　毅. TSA-OT: 一个调度 Out-Tree 任务图的算法[J]. 计算机学报, 2001, 24(4): 390-394.

[5] Kwok Y K, Ahmad I. Dynamic Critical-path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors[J]. IEEE Transactions on Parallel and Distributed Systems, 1996, 7(5): 506-521.

[6] Darbha S, Agrawal D P. Optimal Scheduling Algorithm for Distributed-memory Machines[J]. IEEE Transactions on Parallel and Distributed Systems, 1998, 9(1): 87-95.