

# 基于模型的企业多 Agent 系统开发方法

李东华, 沈文轩, 张媛媛

(鞍山科技大学计算机学院, 鞍山 114044)

**摘要:** 应用企业建模方法建立企业模型, 运用提出的 Agent 构造方法对模型进行 Agent 转化, 得到企业多 Agent 系统模型。这一开发方法分为两个阶段: 建模阶段和 Agentify 阶段。两阶段相承接构成一个完整的系统方法, 为实际应用提供了理论指导与方法依据。以企业管理诊断系统开发实例进行阐述, 证明了方法的有效性。

**关键词:** 企业模型; IDEF; 企业管理诊断系统; Agent; 多 Agent

## Model-based Development Method for Enterprise MAS

LI Donghua, SHEN Wenxuan, ZHANG Yuanyuan

(Computer School, Anshan Univ. of Sci. and Tech., Anshan 114044)

**【Abstract】** The method proposed can obtain an enterprise MAS model based on some models of enterprise which are built with the enterprise modeling method, by agentify them with agent elicitation rule. This development method comprises two joint steps: modeling of problem domain and agentify process, which are integrated systematic method, provides guide in theory and methods for application system development. As an illustration of the presented method, it describes the real case of enterprise management diagnostic system in detail.

**【Key words】** Enterprise model; IDEF; Enterprise management diagnostic system; Agent; Multi-agent

随着Agent技术与多Agent系统研究的深入, 它们在生产组织与信息管理、制造过程控制、机器人技术、电子商务等领域的应用不断扩大, 但真正展开的基于Agent的应用却不多, 文献[1,2]在综合研究后得出结论: 开发面向Agent的建模技术不属于Agent研究的核心技术, 也正因为缺乏面向Agent的软件工程学使得Agent技术在工业应用中得不到长足的进展; 面向Agent方法与基于Agent的系统建模需求之间存在差距, 现有的一些重要的建模技术也因为缺乏表达能力、开发生命周期的支持等因素存在无法满足工业应用的问题<sup>[3]</sup>。在企业的复杂环境下开发基于多Agent的系统必须对企业有全面、系统的理解, 在理解的基础上采用企业建模方法建立企业模型, 然后对模型依据一定的方法和原则使之Agentify后得到多Agent系统, 即本文提出的基于模型的企业多Agent系统开发方法, 能够弥补面向Agent系统建模方法与应用之间的差距。

### 1 基于建模理论的企业多 Agent 系统开发方法

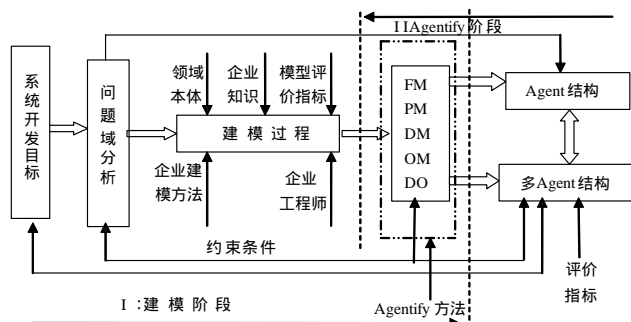


图1 基于模型的企业多Agent系统开发方法

在建模目标的驱动下应用企业建模方法从多角度对问题的领域知识进行理解, 结合系统对建模的要求进行抽象思考,

对问题域的理解越清晰, 模型描述会越准确。对问题域建模后, 使用 Agentify 方法, 对各种模型再次综合分析概括、抽象, 构造出系统的多 Agent 模型。图 1 是基于模型的企业多 Agent 系统开发方法的示意图, 分为两个阶段: 阶段 I 是对问题域建模, 使用各种企业建模方法对问题域建模; 阶段 II 是在阶段 I 得到的模型基础上使用 Agentify 方法构造具体的单体 Agent 及多 Agent 系统。

#### 1.1 问题域建模阶段

阶段I在明确系统开发目标后应用企业建模方法, 结合领域、企业知识对问题域进行建模分析, 得到综合反映企业整体面貌的一组模型。领域知识与领域本体对于获得对企业的一致性描述、建立企业模型是很重要的控制信息, 系统开发目标和模型评价体系是调整模型的重要参考<sup>[4]</sup>。建立这些模型是并行、递阶的过程, 因为各种模型相互参考, 之间有信息的传递, 具有关联性。

#### 1.2 Agentify 阶段

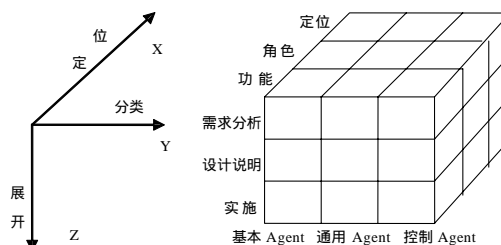


图2 Agentify方法模型

**基金项目:** 辽宁省教育厅科研基金资助项目(105L011)

**作者简介:** 李东华(1972 - ), 男, 工程师、硕士生, 主研方向: 企业建模理论, 多 Agent 技术; 沈文轩, 教授; 张媛媛, 硕士生

**收稿日期:** 2006-07-17

**E-mail:** lidongh328@sohu.com

开发多 Agent 系统关键在于构造具有独立功能的单体 Agent 和 Multi-Agents。阶段 II 是在阶段 I 得到的企业模型基础上, 概括提取 Agent 的过程。对各种模型依据一定的原则进行分析、抽取单体 Agent 和 MAS 的构造方法称为 Agentify 方法, 如图 2 所示, 采用功能分解方法从各模型中综合提取出各单体 Agent, 然后沿 X 轴方向从对应的功能、在系统中的角色、在系统中的地位 3 个角度进行分析; 沿 Y 轴方向是分类 Agent, 这种划分对于 Agent 功能界定、明确构造出的 Agent 在 MAS 系统中的地位是一种参考: 基本 Agent 是系统必需的功能 Agent, 位于系统底层或中间层; 通用 Agent 是系统中的主体综合功能单元 Agent, 处于底层或中间层; 特殊 Agent 是位于系统中、高层的用于系统支持和控制的辅助功能 Agent。具体 Agent 确定策略主要依据任务、资源情况, 在 Agentify 分析过程中依据功能分解、目标分解和粒度大小, 遵循 3 条原则: (1) 功能原则。提取 Agent 要考虑其粒度大小, 功能划分的太小需要定义的接口太多很难集成<sup>[5]</sup>; (2) 控制性原则。满足 MAS 协作的同时要使系统结构简单, 便于灵活控制, 但功能划分不能太大, 否则不能有效配置, 修改困难; (3) 数量原则。在满足原则 (1) 与原则 (2) 的同时以少为好, 便于控制系统, Agent 太多影响系统的运行效率。对提取到的 Agent 沿 Z 轴的 3 个阶段进行具体分析:

(1) 需求分析: 对提取的 Agent 进行功能分析, Agent 单体、群体是否能实现系统目标;

(2) 设计说明: 从设计角度分析、明确单体 Agent 如何实现其功能, 多 Agent 采用的结构模式等, 满足用户需求和系统约束, 通过 UML、业务协作流程图及 PAD (问题分析图) 等方法进行细化分析;

(3) 实施: 将设计说明的相关内容转化为具体的 Agent 及多 Agent 系统的代码设计。

在 Z 轴上需求分析是基础, 实施是目的, 设计说明是连接二者的桥梁, Agentify 过程需要逐步推进、反复迭代, 同样适用于子系统的分析。Agentify 的过程如图 3 所示, 表示 Agent 具体构造过程和各种模型的关系与作用。

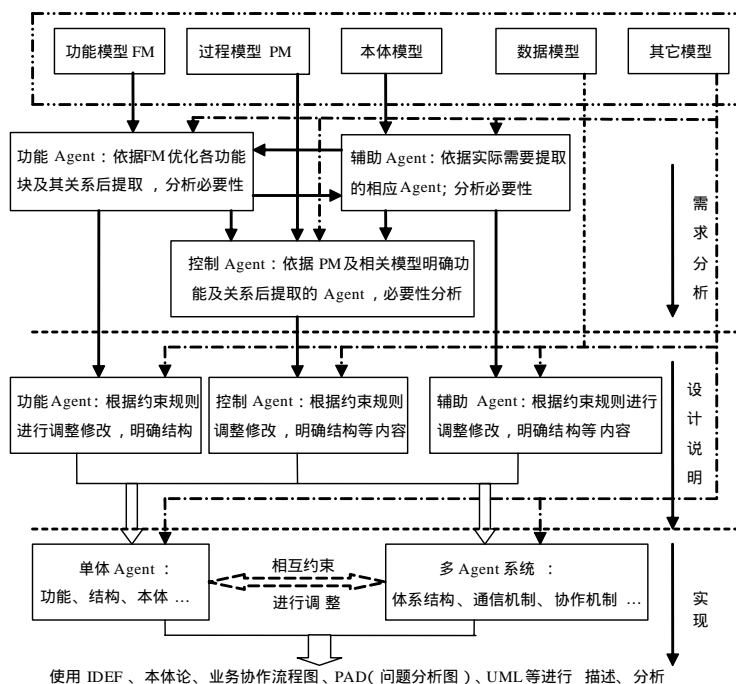


图 3 Agentify 过程 Agent 状态转化

### 1.3 单体 Agent 与多 Agent 分析

经过调整、修改后得到的单体 Agent 和 MAS 是对问题域的完整反映, 模型与 Agent、MAS 是映射关系。

#### (1) 多 Agent 的协作

可以采用业务协作流程图分析各 Agent 间的协作关系, 协作流程图融入时间、状态变化等因素, 表现出的内容、信息更清晰明了, 通过协作者之间的活动能体现各 Agent 间协调关系。Agent 间交互的低层网络通信协议选择 TCP/IP, Agent 间的交互主要有信息传递和服务请求; Agent 间通信的语言选用 KQML。

#### (2) 单体 Agent 的设计

单体 Agent 设计要有利于描述 Agent 的逻辑结构, Agent 自身采用何种结构完全取决于其具体功能, 因为对 Agent 的结构、功能、运作方式的分析都是 Agent 编程的基础。单体 Agent 的设计采用由模型分析到详细设计的方法。采用本体论对涉及的领域、问题域知识进行规范化和标准化, 能够使多 Agent 中不同视图模型之间在语义、语法上保持一致性, 可复用的本体还能够减少设计中的重复工作。使用 UML 的用例图可以进行交互性很强的功能性分析, 对于非功能性的性能、约束、可靠性等控制应用 IDEF0 进行分析, 同时可以结合伪代码和 PAD 进行细化分析。

### 1.4 开发方法的特点

这一开发方法有下述特点:

(1) 目标驱动与模型驱动相结合: 为了满足系统目标而在问题域上应用模型驱动的分析方法, 通过对建立的各种模型进行迭代、递阶来分析如何实现系统目标。

(2) 同时关注单体 Agent 及各 Agent 的协作关系: 在建设多 Agent 系统过程中, 通过功能活动和行为单元提取单体 Agent 保证 Agent 的功能和逻辑结构的准确; 通过全局 FM、PM 得到 MAS 的过程中保证 Agent 间的协作、协调关系。

(3) 是一个系统化的开发方法: 方法的不同阶段互相紧密联系、传承接续, 为开发多 Agent 系统提供了完整的分析、建设方法, 其中 Agent 构造方法具有需求分析、详细设计及实现阶段的全生命周期。

(4) 适合企业环境多 Agent 系统开发: 建立模型的理论和方法都是针对企业复杂情况的, 因此适合在企业环境下开发多 Agent 系统。

## 2 企业管理诊断系统应用实例

企业管理诊断系统是依据企业诊断学与企业管理学的理论和方法对企业运营进行全面科学诊断的信息化系统<sup>[6]</sup>。依据诊断任务要求, 通过调查、问卷等方式获得与任务相关的数据信息, 经过诊断系统的处理后变为相应的诊断结论输出。

对企业诊断系统进行过程建模得到如图 4 所示的 IDEF3 过程流程图 (PFN), PFN 用来获得、管理和显示以过程为中心的知识, 每个矩形代表一个行为单元 UOB;

对图 4 进行行为划分得到 6 个结构: 1 包括 UOB1、UOB2 和逻辑或交汇点 1, 代表与任务相关的功能; 2 表示诊断进程; 3 表示对诊断结论的约束检验; 4 表示对不符合要求的任务重新组织诊断; 5 由结构 2、3、4 构成, 表示具体诊断功能模块; 6 表示按照要求对诊断结论进行规范处理模块。

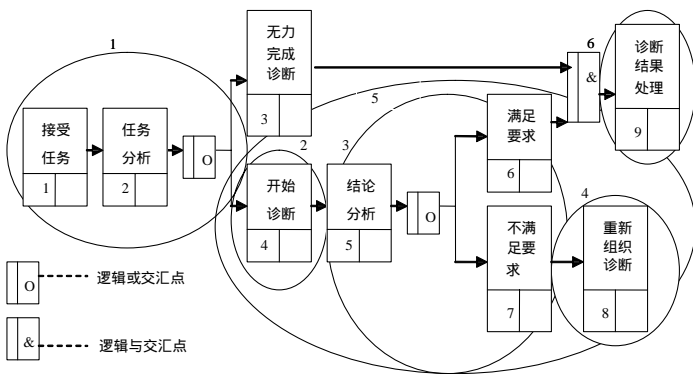
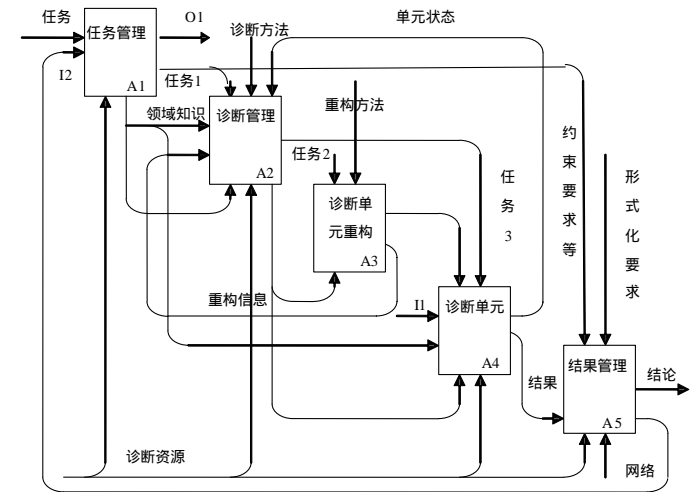


图4 企业管理诊断系统的 IDEF3 过程流图 (PFN)

图5是系统的A0级 IDEF0 功能图, IDEF0 模型中的功能活动 (FA) 与 IDEF3 过程模型中的 UOB 间存在联系, 即可以由 UOB 组合出 FA, 也可以由 FA 分解得到 UOB, 但它们之间不是一一映射, 一旦 UOB 确定下来就可以得到相应的 IDEF0 的 FA 了。



任务: 系统接受的诊断任务; 任务1: 系统可以接受的任务; 任务2: 需重构的任务; 任务3: 系统可以完成的任务  
I1: 诊断所需资料; I2: 结论不满足约束的信息; O1: 系统不能接受的任务; 诊断资源: 人、设备、规章等

图5 企业管理诊断系统的 A0 级 IDEF0 图

通过建立过程模型和功能模型对问题域有了清晰的认识, 综合图4、图5可以总结归纳出企业管理诊断系统包括6类主要功能模块: 功能(原子)单元模块, 诊断单元模块, 诊断管理模块, 诊断单元重构模块, 任务管理模块, 结果管理模块, 将它们 Agentify 得到各类单体 Agent 和 MAS: 对图5企业管理诊断系统 IDEF0 图分析可知, 每类功能模块各自具有一定的权限, 在自身功能机制和控制等局部数据的驱动下完成工作, 而且需要自主作出局部决策, 具有一定程度的自治性, 但是系统要得到全局结果要求各个功能模块间同步、协调; 因此在各功能模块处引入 Agent, Agent 是具有独立、自主行为能力, 能够满足其设计目标的计算系统, 独立处理事件的 Agent 能满足企业管理诊断系统的自主性和适应性要求; 另一方面多个 Agent 协调、协商、协作可以完成较高层的任务, 在各层设置 Agent 可以利用多 Agent 的协调性、反应性等特点更好地进行协调控制。从图5中的功能块构造出具体 Agent, 同时要设计一些 Agent 来满足系统的辅助功能:

(1)原子单元 Agent: 与原子单元对应, 包括系统必需的原子单元, 位于系统的底层。

(2)功能单元 Agent: 是功能单元的集合, 能够完成独立的子任务, 在系统中作为自主性实体存在, 处于系统的底层,

是系统的主体成分。各 Agent 相互协作完成更高级的任务。

(3)诊断单元 Agent: 与诊断单元模块对应, 是系统预先定义的功能 Agent, 它们并不封装组成元素, 只存放所属单元 ID、名称、本体论描述等信息, 被 Agent 单元管理 Agent 激活后组织各组成 Agent 进行诊断, 具有自主性和可扩充性。

(4)Agent 单元管理 Agent: 与诊断管理模块对应, 负责预定义诊断单元 Agent 和重构的诊断单元 Agent 的管理工作, 不封装组成元素, 只存放所属诊断单元 ID、名称、本体描述等信息, 具有注册、管理、注销诊断单元 Agent 的功能。负责激活相关联的诊断单元 Agent, 能够依据具体情况判断是否需要重构; 如果经过重构仍无法满足任务要求, 则把任务交给结果管理 Agent 处理。

(5)重构 Agent: 与诊断单元重构模块对应, 负责对诊断单元进行重构或构建新的诊断单元。预定义诊断单元 Agent 可提高诊断效率, 但由于不同的诊断任务涉及的功能可能重复或交叉, 若全部预定义, 系统将无法扩充, 不能满足诊断任务变化的需要, 因此采用预定义与重构相结合的方式, 只预定义出现频率较高的诊断单元; 重构是指改变预定义诊断单元 Agent 的组成来满足新要求, 可以扩充, 也可以完全重新组合成新的诊断单元。重构可以降低系统的刚性增加柔性, 能适应诊断任务多样化的要求, 减少诊断时的重复工作。重构生成的新诊断单元 Agent 的相关数据被传送给诊断单元管理 Agent 注册, 新的诊断单元 Agent 动态存在直至完成任务后被注销。

(6)任务 Agent: 与任务管理模块对应, 负责接收、分析任务信息, 并传送给诊断单元管理 Agent 组织诊断; 若得到需要重构的反馈则启动重构管理 Agent, 是与外部的接口 Agent。采用人-机接口方式, 降低系统复杂性的同时可以增加系统的效率和执行的可靠性。

(7)结果管理 Agent: 与结果管理模块功能对应, 依据约束性条件对诊断结果进行判断, 若不符合则反馈给任务管理 Agent 重新组织诊断; 根据各 Agent 诊断结果生成局部或全局诊断结论。

Agent 的组织结构采用图6的混合式多 Agent 体系结构, 图7是以任务 Agent 为例进行细化分析得到的 PAD 及伪代码。使用协作流程图表示各 Agent 之间的协作, Agent 需要协作时发出服务请求, 只有服务请求得到响应才表示在相关 Agent 间建立了联系, 必需的信息传递才会进行; 协作流程图能详细地表示 MAS 中信息、服务请求在各 Agent 间的传递、Agent 进行协作共同完成目标的过程。

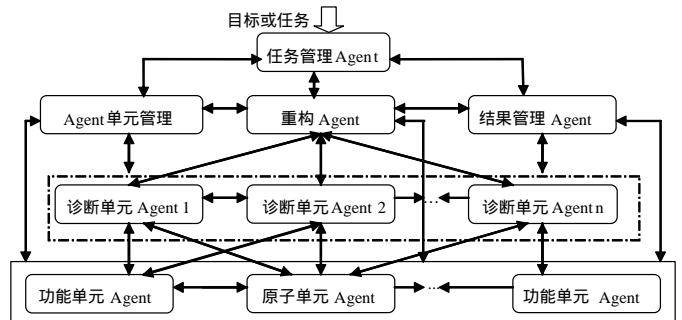
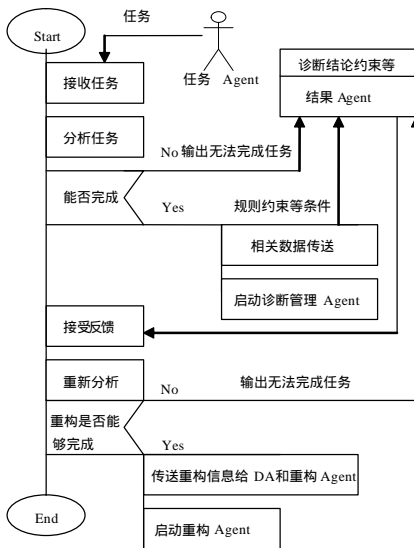


图6 基于多 Agent 的企业管理诊断系统总体结构



Agent Name : 任务 Agent (TA)  
 Input Parameters: 诊断任务及约束、规则等要求  
 Output Parameters : 任务子集、约束、规则等

```
Pseudo Code Body :
Begin
  //>>>接受并分析任务
  通信、信息处理 模块 ???
  //>>> 是否能够完成任务
  If( 能够完成 )then
    Begin
      //>>> 启动诊断管理 Agent 并向结果
      Agent 传送规则、约束等
      通信模块
      //>>> 接受回馈诊断信息
      //>>>重新对任务进行分析
      信息处理 模块 ???
      //>>> 通过重构是否能够满足要求
      If( 能够满足 )then
        Begin
          //>>> 传送重构信息给
          DA 和重构 Agent、启动重构 Agent
          通信模块 ???
        End
      Else
        //>>>传送信息给结果 Agent, 不接受
        任务
        通信模块 ???
      End
    End
  End
```

图 7 任务 Agent 的 PAD 及算法伪代码

### 3 总结

文中提出的建立企业 Agent 系统的方法实现了在对问题域全面理解的基础上构造 Agent 与 MAS 的目标, 将建模技术与 Agentify 方法紧密结合, 能够满足开发企业 Agent 系统的最新要求: 整体工作分析和功能个体独立运行。完善了基于

(上接第 260 页)

### 4 结论

本文所描述的“基于 IXP2400 的高性能网络认证/计费系统”既可以与 802.1x 协议设备兼容, 进行互联互通, 也具有一些有别于这些现行技术的特点, 包括: (1) 系统支持用户名、接入端口、IP 地址、MAC 地址等多重信息的绑定; (2) 系统不需改变网络原有拓扑, 节省网络投资; (3) 支持多种接入方式等。系统采用了 Intel IXA 计算架构及 IXP2400 处理器构建系统的硬件数据处理平台, 从而使得系统的性能得到了提高。

但是, 系统中还需要在其管理主机上部署多种软件, 存在着软件模块功能分散等缺陷。下一步的研究工作集中在将认证服务器和计费服务器移植到系统硬件平台中的 XScale 处理器上, 从而实现减小系统规模、降低系统造价、固化系统功能、增强系统的稳定性和灵活性的目的。

#### 参考文献

1 彭伟. 使用 802.1x 实现校园网认证[J]. 计算机应用, 2003,

Agent 的建模技术, 为企业建模和信息化的发展提供有效的方法保证和技术支持, 可以作为新一代企业建模方法学的构造依据, 对于应用系统开发能起到很好的指导和帮助; 并以企业管理诊断系统为例进行说明。现正在应用这种方法指导企业管理诊断系统的开发, 同时进行 Agentify 方法的改进工作, 包括 Agent 信息交互机制的研究, 使这一方法更完善、具有更广泛适用性。

#### 参考文献

- Janssens N, Steegmans E, Holvoet T. An Agent Design Method Promoting Separation Between Computation and Coordination [C]//Proc. of ACM Symposium on Applied Computing, Nicosia, Cyprus. 2004: 456-461.
- Sturm A, Shehory O. Towards Industrially Applicable Modeling Technique for Agent-based Systems[C]//Proc. of International Joint Conference on Autonomous Agents and Multi-agent Systems, Bologna, Italy. 2002-06.
- Sturm A, Dori D, Shehory O. Single-model Method for Specifying Multi-agent Systems[C]//Proc. of International Joint Conference on Autonomous Agents and Multi-agent Systems, Melbourne, Australia. 2003-06.
- 陈禹六. IDEF 建模分析设计方法[M]. 北京: 清华大学出版社, 1999-05.
- 盛步云. 基于多 Agent 的分布式工艺决策研究[D]. 武汉: 武汉理工大学, 2001-05.
- 夏伯忠. 企业诊断学[M]. 沈阳: 辽宁大学出版社, 1990.

23(3): 85-87.

- 蒋海锋, 苗放. 802.1x 标准和 Radius 协议的扩展应用研究[J]. 计算机应用, 2003, 23(11): 88-90.
- 谭章熹, 林闯, 任丰源, 等. 网络处理器的分析与研究[J]. 软件学报, 2003, 14(2): 225-265.
- 刘奖, 安常青, 何涛, 等. 基于网络处理器的高速网络计费管理系统[J]. 小型微型计算机系统, 2005, 26(9).
- Lakshmanamurthy S, Liu K, Pun Y, et al. Network Processor Performance Analysis Methodology[J]. Intel Technology Journal, 2002, 6(3).
- Zheng T S, Zhao Rongcai, Xie Kangmin. A Model for NP-based Application Design[C]//Proceedings of the 6<sup>th</sup> Internal Conference of Parallel and Distributed Computing, Application and Technologies. San Jose: IEEE Computer Society Press, 2005-12.

(上接第 278 页)

#### 参考文献

- Watts S H. The Team Software Process (TSP)[R]. Carnegie Mellon Software Engineering Institute, Technical Report: CMU/SEI-2000-TR-023, 2000.
- Noopur D, Julia M. The Team Software Process (TSP) in Practice: A Summary of Recent Results[R]. Carnegie Mellon Software Engineering Institute, Technical Report: CMU/SEI-2003-TR-014, 2003.

- 史美林. 计算机支持的协同工作——理论与应用[M]. 北京: 电子工业出版社, 2000.
- 王养廷. 面向 CMM 的 TSP 技术研究及支持环境实现[D]. 北京: 北京航空航天大学, 2002.
- 徐遵义. CSCT 协同工作的设计与实现[D]. 西安: 西安交通大学, 2002.
- 燕锦华, 刘宗田, 李强. 基于 TSP 的软件质量控制平台设计与实现[J]. 计算机工程与设计, 2006, 27(5).

