文章编号: 1000-6788(2014)09-2446-11 中图分类号: TP391 文献标志码: A

面向主体的协同产品开发过程动态仿真

李英姿 [1], 张 硕 [1], 张晓冬 [1], 王爱平 [2]

(1. 北京科技大学 东凌经济管理学院, 北京 100083; 2. 北京现代汽车有限公司, 北京 101300)

摘 要 为了研究设计人员在协同产品开发过程中的行为以及过程管理中的典型问题, 首先基于复杂适应系统 (complex adaptive system, CAS) 和多主体系统 (multi-agent system, MAS) 理论建立了面向主体的协同产品开发过程动态仿真集成模型, 该集成模型主要包括 4 大部分: 设计主体, 管理主体, 环境对象 (任务、资源和产品信息) 以及动态仿真系统的功能模型. 其次, 介绍了动态仿真系统的开发方法, 并对仿真流程、类的封装及各主要函数的调用过程进行了详细的描述; 最后, 对某摩托车发动机开发项目进行了仿真实验研究. 仿真结果表明, 面向主体的动态仿真能为协同产品过程管理提供科学和量化的决策依据, 同时, 为该过程中人员的行为研究提供了一种新的思路.

关键词 主体; 协同产品开发; 过程管理; 动态仿真

Agent-oriented dynamic simulation in collaborative production development process

LI Ying-zi[1], ZHANG Shuo[1], ZHANG Xiao-dong[1], WANG Ai-ping[2]

(1. Donglinks School of Economics and Management, University of Science and Technology Beijing, Beijing 100083, China; 2. Beijing Hyundai Motor Company, Beijing 101300, China)

Abstract In order to study the behaviors of designers in collaborative production development process and the typical problems of process management, an integrated model of agent-oriented dynamic simulation based on CAS and MAS theory was set up firstly. The integrated model included 4 important parts: Designer agents, arbitration agent, environment objectives (task, resource and product information) and function model of the dynamic simulation system. Secondly, simulation system development method was introduced, and simulation process, class packaging, and functions calling process were described in detail. Lastly, the proposed method was applied on some motorcycle engine development project. The simulation results show that agent-oriented simulation can provide scientific and quantized decision basis, simultaneously, provides a new idea for behavior study of designers in the process.

Keywords agent; collaborative production development; process management; dynamic simulation

1 引言

协同产品开发的复杂性, 动态性和分布性等特点使其过程中的各类管理问题更加突出. 学者们利用仿真方法对该过程进行预测、评价、管理及优化, 为提高协同产品开发过程管理效率提供了新的途径. 其中, 利用 Petri 网和 DSM 对产品开发过程进行仿真是最典型的方法. Petri 网是完全从过程的角度出发为复杂系统的描述与分析而设计的一种有效模型工具, 适合描述并发、冲突、同步等重要行为现象, 并具有形式化步骤与数学图论相支持的理论严密性 [1]; 而 DSM 能够表达任务间的信息耦合与迭代关系, 并能有效地支持过程重构 [2], 从而使得基于 DSM 的产品设计过程模型具有良好的可操作性和稳健性. 许多学者在该领域已经取得了大量值得借鉴的研究 [3−5], 例如: 李伯虎等人分别基于 DSM 和 Petri Net 对产品开发过程建模进行了深入研究, 相关研究成果描述了产品开发子项目内部任务的串行迭代关系, 并通过仿真分析由于迭代带来的周期影响 [6]; 另外, 他们运用基于着色 Petri 网和基于扩展的高级关系 Petri 网建立了产品开发过程的调度模

收稿日期: 2012-10-18
资助项目: 国家自然科学基金重点项目 (71231001); 国家自然科学基金 (71171019); 中国博士后科学基金特别资助项目 (2013T60064)
作者简介: 李英姿 (1985–), 女, 山东新泰人, 博士, 讲师, 研究方向: 协同产品过程管理及其仿真, E-mail: liyz@ustb.edu.cn.

型, 对不确定条件下的产品开发项目进行动态排序 [7]; 杨青等人分别考虑返工风险和活动重叠, 应用 DSM 对项目进行仿真优化, 使得项目的持续时间及返工时间显著降低 [8−9]; 李洪波等人构建了考虑活动随机重叠与迭代的 DSM, 并建立了复杂产品开发过程的仿真模型, 实现对项目费用、工期及风险的预测和评估 [10]. 然而, 以上研究所基于的仿真模型均是以任务为中心, 通过任务触发规则驱动仿真进行. 协同产品开发是一项典型的知识工作, 设计人员是该过程中的主体, 其行为对整个过程管理有着至关重要的作用. 因此, 面向主体的过程仿真更能反映出设计人员的主动性、自治性和协同性. 近年来, 面向主体的动态仿真方法也在各领域得到了广泛应用, 包括公共管理 [11]、机械制造 [12]、供应链管理 [13]、知识管理 [14] 等. 针对产品开发过程, Garcia 等人讨论了在创新和新产品开发过程中主体方法的应用, 主要包括创新扩散、组织策略以及知识/信息流动等三个方面 [15]; 刘晋飞等人针对产品协同设计问题, 提出了基于多 agent 的设计策略 [16]; 蒋伟进等人重点从 agent 心智变化角度描述动态协作任务求解模型实现的六个阶段 [17]. 然而, 以上较少涉及协同产品开发过程中设计人员的行为研究.

因此, 本文在复杂适应系统 (complex adaptive system, CAS) 理论和多主体系统 (multi-agent system, MAS) 理论的指导下 [18−19], 在前期研究的基础上 [20−21], 建立了面向主体的协同产品开发过程动态仿真集成模型, 该模型集成了设计主体、管理主体、环境对象等要素以及动态仿真系统的功能模型. 仿真实验结果表明, 本文所提出的仿真方法能够对协同产品开发过程管理提供科学和量化的决策依据以及优化方案, 为研究过程人的行为提供了新的方法.

2　面向主体的协同产品开发过程动态仿真集成模型

在协同产品开发过程中, 设计团队是由来自不同部门, 不同专业技术背景的多设计人员共同组成, 针对某个设计目标, 设计人员在设计资源、产品信息等外部环境的约束下, 不断改变设计任务状态, 直至完成. 在此过程中设计人员之间以及人员与环境之间不断地相互作用, 相互适应. 通过这种交互与适应, 设计人员也因此不断学习, 积累经验, 并根据学到的经验改变自身的知识结构和行为机制, 推动整个系统动态地向目标推进. 结合 CAS 理论, 可将设计人员作为智能主体, 称为设计主体, 而设计任务、设计资源等可组成设计主体活动的环境, 从而, 可将协同产品开发系统定义为一个由设计主体、设计任务、产品信息、设计资源等对象组成的复杂自适应系统, 以多主体建模思想作为建模原理为基础, 建立协同产品开发过程动态仿真集成模型. 该模型包括 4 大部分: 设计主体模型、管理主体模型、环境因素模型以及动态仿真系统. 其中, 环境因素包括任务、资源及产品信息, 如图 1 所示. 该模型描述了设计主体、仲裁主体在协同产品开发过程中的主要知识和行为, 并描述了主体进行任务选择、资源选择、伙伴选择、资源冲突消解、伙伴选择冲突消解的行为. 同时, 产品信息的分类和状态也反映在该模型中. 因此, 协同产品开发过程可以描述为: 设计主体在设计资源和产品信息的约束下不断改变任务的状态, 直至任务完成的过程. 仿真模型充分体现了产品开发过程以人为中心的特点, 以及设计主体在过程中的主动性、自治性和协作性特征, 从而能更好地支持设计主体各类行为的分析.

由图 1 可知, 面向主体的协同产品开发过程动态仿真系统包括四个层次: 数据处理层、运行层、模型层和用户层. 用户通过验证后可在可视化平台上建立仿真模型, 定义主体和各类环境对象的属性, 设定主体、任务和信息等的约束关系, 运行层将实现仿真模型的运行, 最后进入数据处理层, 通过统计分析、图形化输出, 以及假设检验实现仿真结果的处理; 数据处理结果反馈至用户层, 用户参考这些分析结果, 调整、优化当前的协同产品开发过程, 修改仿真模型及其参数进行新一轮的仿真, 直至得出优化方案. 该仿真系统采用主体驱动的动态仿真原理, 以固定增量推进的离散系统仿真为基础, 实现了对协同产品开发过程中设计主体、任务、资源以及信息的状态变化及交互作用的动态模拟, 并支持各类行为的过程分析.

在协同产品开发过程中, 主体的行为大致可分为三个阶段: 自主选择阶段、二者协商阶段、仲裁选择阶段. 在自主选择阶段, 设计主体根据一定的选择规则对任务、资源和协作伙伴进行选择, 选择规则的设定除了包括任务重要度、任务紧急度、任务复杂度、资源功能匹配度、能力匹配度等客观因素, 还包括设计主体自身对任务、资源、伙伴的主观偏好度; 二者协商主要针对冲突只涉及两个主体的情况, 两个主体可根据所承担任务的优先级进行协商; 当冲突涉及三个及以上的主体, 各设计主体将冲突信息提交到仲裁主体处, 仲裁主体将根据各设计主体、任务、资源的状态调用策略库中的策略进行优化配置. 设计主体对自身任务箱中的任务选择与调度可参见文献 [20]. 二者协商方法和仲裁策略分别如表 1、2 所示.

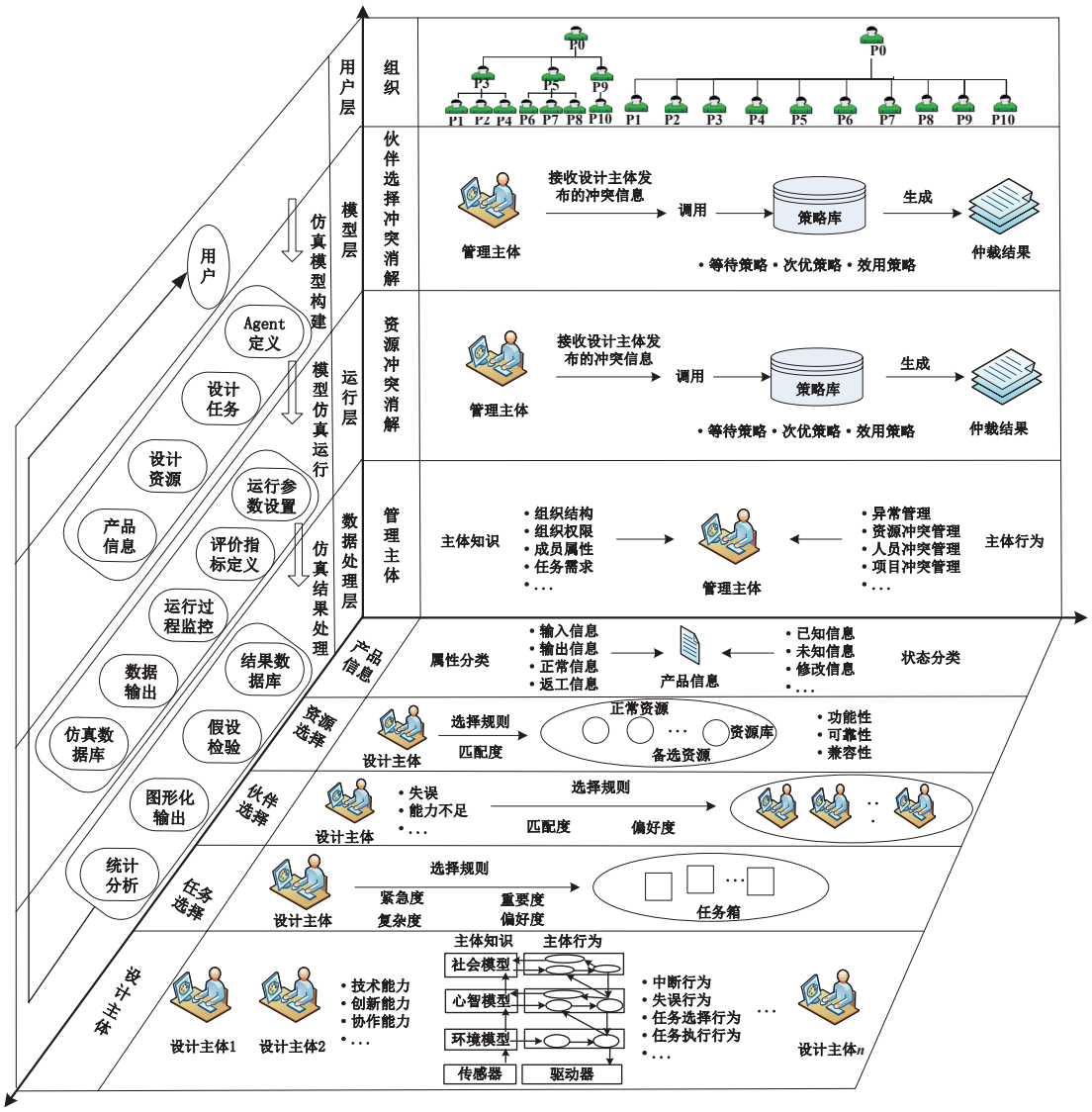

图 1 面向主体的协同产品开发过程动态仿真集成模型

表 1 二者协商的方法

资源冲突二者协商 (承担低优先级任务的设计主体向承担高优先级任务的设计主体发出协商请求)		协作伙伴选择冲突二者协商 (承担低优先级任务的设计主体向承担高优先级任务的设计主体发出协商请求)
建立协商函数: $f_{ii'}(t)=\frac{FT_i(t)}{RT_{i'}(t)}$ $FT_i(t)=\max(LT_{ik}(t),LR_{ijk}(t))$ $FRT_{i'}(t)=\max(LT_{i'k}(t),LR_{i'jk}(t))$ 计算损失率: $LT_i(t)=\frac{\sum_{j=1}^{TM} req_j(t)}{\sum_{j=1}^{TM} req_j(t)+NT_j}$ $LR_{ijk}(t)=\frac{M_{ijk}-M_{ijk'}}{M_{ijk}}$ $f_{ii'}(t)>1$: 设计主体 i 占用正常资源 $f_{ii'}(t)<1$: 设计主体 i' 占用正常资源	(Step1) (Step2) (Step3) (Step4)	按照规则进行协商: 高匹配度、高优先级 以及主观偏好等 1. 双方达成一致 2. 达到终止条件 (协商次数 ≤ 3), 协商未果, 提交给仲裁主体 协商结束

3 动态仿真系统的开发

面向设计主体的产品开发过程仿真流程的原理为: 通过模拟各设计主体的自主行为以及与产品开发团队中其他主体进行协作等交互行为, 从而不断改变更新设计任务、产品信息和设计资源的状态, 最终实现完成整个产品开发过程的目标.

表 2　仲裁主体调用的策略

资源冲突消解策略		协作伙伴选择冲突消解策略
(承担高优先级任务的设计主体将优先获得正常资源)	(Step1)	(承担高优先级任务的设计主体将优先获得最优协作伙伴)
其他没有获得正常资源的设计主体选择等待.	等待策略 (Step2)	其他没有获得最优协作伙伴的设计主体选择等待.
其他没有获得正常资源的设计主体选择备选资源.	次优策略 (Step2)	其他没有获得最优协作伙伴的设计主体选择次优协作伙伴.
其他没有获得正常资源的设计主体通过效用函数值选择等待或者使用备选资源.	效用策略 (Step2)	其他没有获得最优协作伙伴的设计主体通过效用函数值选择等待或者选择次优协作伙伴.
$E_i(t)=\frac{LT_i(t)}{LR_{ijk'}}, LR_{ijk}=\frac{M_{ijk}-M_{ijk'}}{M_{ijk}}$		$f_i(t)=\frac{LP_i(t)}{LT_i(t)}, LP_i(t)=\frac{M_{ii'j}-M_{ii''j}}{M_{ii'j}}$
$LT_i(t)=\frac{\sum_{j=1}^{TM} req_j(t)}{\sum_{j=1}^{TM} req_j(t)+NT_j}$		$LT_i(t)=\frac{\sum_{j=1}^{TN} req_j(t)}{\sum_{j=1}^{TN} req_j(t)+NT_j}$

其中, $E_i(t)$ 和 $f_i(t)$ 分别为 t 时刻资源选择和伙伴选择的效用函数; $f_{ii'}(t)$ 为 t 时刻资源冲突的协商函数; LR_{ijk} 和 $LP_i(t)$ 分别为 t 时刻使用备选资源和选择次优人员的匹配度损失率; M 为各类匹配度; $req_j(t)$ 为 t 时刻 j 任务的剩余执行时间, NT_j 为 j 任务的标准执行时间; $LT_i(t)$ 为时间损失率, TM 和 TN 分别为所承担任务优先级小于竞争同一资源和同一协作伙伴的其他设计主体承担任务优先级的任务数.

3.1　仿真流程

面向主体的协同产品开发过程仿真流程的单次仿真步骤如图 2 所示, 具体如下:

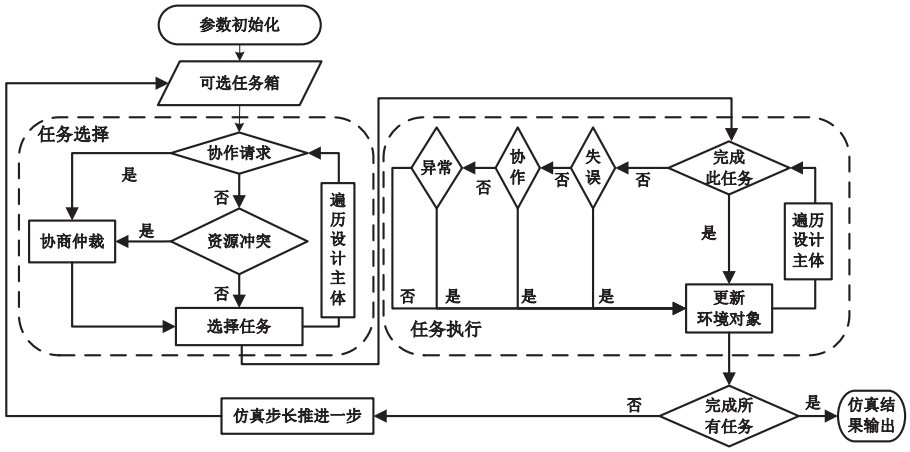

图 2　面向主体的协同产品开发过程仿真流程

①初始仿真步长为 0, 初始化设置各模型变量;

②每个步长开始阶段, 设计主体更新各自任务箱, 及总任务箱, 任务箱储存满足当前步长执行条件的任务序列;

③进入任务选择阶段, 遍历设计主体, 按任务优先级由大到小的顺序选择当前步长要执行的任务 (规定每个步长, 各设计主体最多执行一个任务), ④ ∼ ⑥为详细的选择步骤;

④判断任务是否为协作任务. 如果是, 进入协商仲裁阶段, 为临时任务选择协作伙伴; 如果否, 进行下一步骤;

⑤判断任务是否存在资源冲突. 如果是, 则进入协商仲裁阶段, 解决存在的资源冲突; 如为否, 进入下一步骤;

⑥选择当前步长要执行的任务, 并推至下一主体, 转步骤③, 直到遍历完所有设计主体, 然后再进入任务执行阶段;

⑦进入任务执行阶段, 遍历设计主体, 当前步长选择到任务的设计主体执行设计任务, ⑧和⑨为任务执行阶段详细步骤;

⑧判断到此步长, 任务是否完成. 如果是, 进入下一步; 如果否, 则判断任务执行过程中是否出现失误、协作、异常等情况, 判断后进入下一步;

⑨根据步骤⑧对任务状况的不同判断, 相应地更新环境对象的变量和状态;

⑩判断所有的任务是否全部完成. 如果是, 仿真结束, 输出仿真结果; 如果否, 转②步;

3.2 类的封装

设计主体、管理主体以及环境对象的模型通过类的封装实现. 其中, 设计主体封装为 Person 类; 环境对象分别封装在 Task(任务)、Resource(资源)、Item(信息) 三个类中; 另外, 仲裁主体是一类特殊的主体, 被封装在 Arbi_Person 类中, 并继承了 Person 类. Person 类将主体的属性和仿真过程中的参数全部封装, Task、Resource 和 Item 则把环境对象属性和参数全部封装. 部分重要的函数如表 3 所示.

表 3 设计主体类与仲裁主体类中的部分函数及功能说明

类的名称	函数及功能说明			
	属性类函数	说明	行为类函数	说明
Person (设计主体)	CString status()	状态 (忙碌/空闲)	void Delete_CoPrsn Unchosed_Cotask()	需要协作的设计主体将临时任务删除出自己可选任务箱
			void Sel_Task_ UsingSpareRe()	选择任务, 使用备选资源
			int Negotia_Sel()	二者协商, 选择可执行任务
	int same[]	同级人员数组	void Sel_Task_Indpdt()	自主选择可执行任务
	int TBox[]	当前步长选择的任务序号	void Sort_TBox()	设计主体的任务箱排序
			CString execute()	设计主体的任务执行函数
			bool If_ReConflict()	判断是否存在资源冲突
	int now_task_todo	当前步长可选任务箱数组	bool If_multi_people()	判断资源冲突是否设计为多个设计主体
	...	...	...	...
Arbi_Person (仲裁主体)	int t_TodoBox[]	总任务箱	void TCollaBox_Sort()	对协作临时任务箱的任务排序
			int Sel_CollabPrsn()	为协作临时任务选择协作伙伴
	int m_PrsnBox[]	协作临时任务的可选设计人员任务箱	int PBox_Sort()	按设计主体匹配度, 对协作任务可选主体数组排序
			int TBox_Sort()	对总任务箱的任务序列排序
			void Prio_Caculate()	计算总任务箱中各任务的优先级
	int m_CollaBox[]	协作临时任务箱	void Sel_TaskTodo()	仲裁主体为设计主体分配任务, 并删除与分配好任务的设计主体相关任务
	...	...	...	...

另外, Task、Resource 和 Item 三类中主要封装的为属性函数, 包括任务状态、类型、任务的紧急度、任务重要度、任务所属项目编号、资源的总量、资源的种类以及产品信息的状态等. 在此不再赘述.

3.3 函数调用

(1) 设计主体任务排序过程的行为函数调用

完成仿真步长 rt 开始阶段的初始化函数调用后, 设计主体进入任务的排序过程, 如图 3 所示. 设计主体遍历其负责的任务箱 p[i].task[], 调用 BOOL 函数 p[i].tasktodo(), 筛选满足执行条件的任务; 如果函数返回 true, 则把任务对应的编号 T[i].id 分别存放到总任务箱 Arbi_P.t_TodoBox[] 和主体任务箱 p[i].TBox[], 如果 T[i].type=“collab temp”, 也将其添加到协作任务箱 m_CollaBox[]; 筛选完毕后, 计算总任务箱 t_TodoBox[] 各任务的获益值 T[i].priority 的大小, 此时调用函数 Prio_Caculate(); 之后, 调用函数 Arbi_P.TBox_Sort()、Arbi_P.TCollaBox_Sort() 和 p[i].Sort_TBox, 分别对三个任务箱的任务序列按照计算的 priority 由大到小排序.

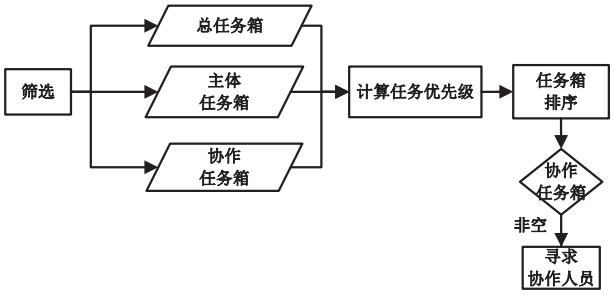

图 3 任务排序流程

完成任务箱排序后, 判断协作任务箱是否为空, 如果非空, 为协作任务箱中的协作临时任务寻找协作伙伴. 仲裁主体 Arbi_P 帮助设计主体 p[i] 产生的临时协作任务寻找协作伙伴. 首先调用 Arbi_P.PBox_Sort (), 计算 p[i] 与 p[i] 同级的设计主体的匹配度 matchility, 按照 matchility 的大小顺序排列 p[i] 的同级设计主体,

作为协作伙伴的候选者; 排序完毕后, 调用 Arbi_P.Sel_CollabPrsn(), 选择协作伙伴. 选择协作伙伴的原则为: 1) 优先选择 matchility 大的设计主体 p[m_pCollab] 作为协作伙伴; 2) 优先选择状态 free, 即当前空闲的候选主体 p[m_pCollab].

(2) 设计主体选择任务过程的行为函数调用

任务排序完成, 并为临时协作任务寻求协作伙伴后, 各设计主体 p[i] 进入任务选择过程. 如图 4 所示, 任务选择流程主要细分为 4 个阶段: 自主选择阶段、二者协商阶段、仲裁选择阶段和任务补充选择阶段. 设计主体在任务选择过程中要调用图中所示的主体行为函数数据库中的大量行为函数, 从而选择当前步长 rt 的执行任务.

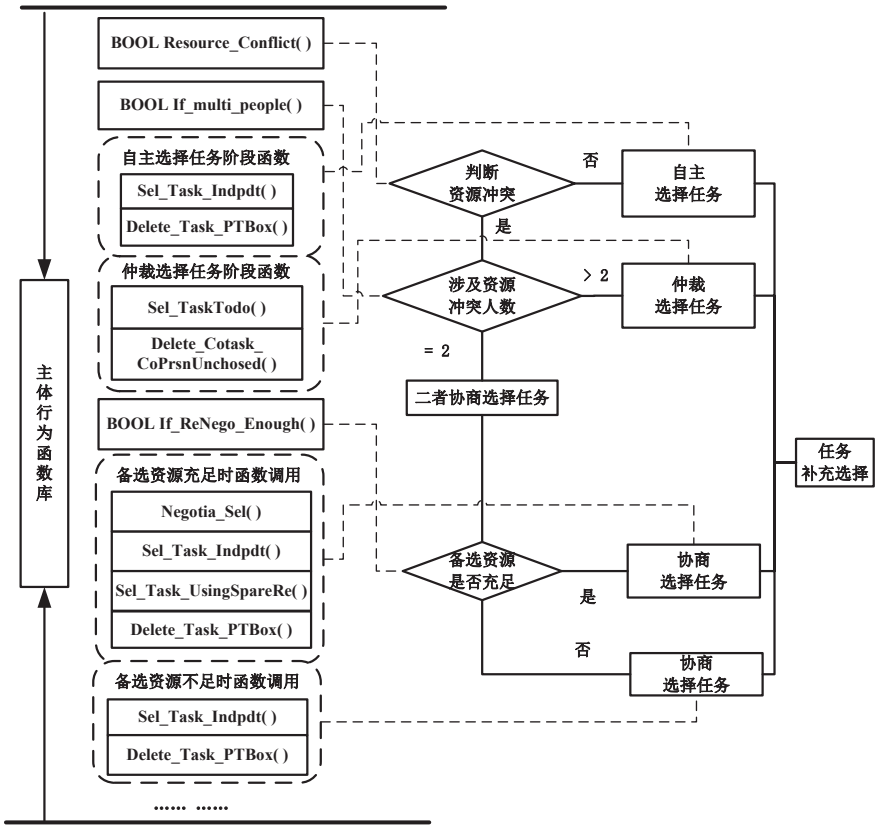

图 4　任务选择流程

①自主选择阶段

设计主体 p[i] 首先判断设计任务 p[i].TBox[0] 与其它设计任务间是否存在资源冲突, 此时调用 BOOL 函数 p[i].Resource_Conflict(); 如果返回 true, 表明 p[i].TBox[0] 没有资源冲突, 设计主体 p[i] 自主选择任务, 调用函数 p[i].Sel_Task_Indpdt(), 得到其当前步长需要执行的任务 p[i].now_tasktodo; 选择完毕后, p[i] 将选择结果提交给仲裁主体 Arbi_P,Arbi_P 调用函数 Arbi_P.Delete_Task_PTBox(), 把 p[i].TBox[] 任务箱中的任务序列从总任务序列 Arbi_P.t_TodoBox[] 删除出去, 便于后续任务的选择.

②二者协商阶段

未涉及资源冲突的设计主体选择完任务后, 进入二者协商阶段. p[i] 首先调用 BOOL 函数 p[i].If_multi_people(), 判断资源冲突是否涉及到多设计主体. 如果函数返回 false, 表明资源冲突只限于两个设计主体, 此时两个设计主体 p[i]、p[j] 协商解决资源冲突; p[i]、p[j] 查看涉及冲突的资源 re 是否有足够的备选资源, 调用函数 p[i].If_ReNego_Enough(). 如果函数返回 true, 表明备选资源充足. 二者调用函数 p[i].Negotia_Sel (), 协商选择 p[i]、p[j] 如何分配现有资源和备选资源. 假设协商结果为 p[i] 使用现有资源, p[j] 使用剩余后的资源和备选资源, 则 p[i] 调用函数 Sel_Task_Indpdt() 选择要执行的任务 p[i].now_tasktodo; p[j] 调用函数 Sel_Task _UsingSpareRe() 在使用剩余资源的前提下选择任务 p[j].now_tasktodo; 二者协商完毕后, 仲裁主体 Arbi_P 调用函数 Delete_Task_PTBox(), 将二者的任务箱 TBox[] 从 Arbi_P.t_TodoBox[] 删除出去. 如果函数 p[i].If_ReNego_Enough() 返回 false, 表明备选资源不足. 比较 T[p[i].TBox[0]].priority 和 T[p[j].TBox[0]].

priority 的大小, priority 值大的设计主体使用此资源, 调用函数 p[i/j].Sel_Task_Indpdt(), 选择执行任务 p[i/j].now_tasktodo; priority 值小的设计主体在后续阶段进行选择; 协商完毕后, 仲裁主体调用函数 Arbi_P.Delete_Task_PTBox() 把 p[i/j].TBox[] 任务序列从 Arbi_P.t_TodoBox[] 删除出去.

③仲裁选择任务阶段

如果函数 p[i].If_multi_people() 返回 true, 则表明设计资源冲突的人员超过了 2 个. 这种情况下, 协商已难以解决资源冲突, 因此设计主体 p[i] 将资源冲突信息提交给仲裁主体 Arbi_P, 任务选择进入仲裁选择阶段. 仲裁主体 Arbi_P 调用函数 Arbi_P.Sel_TaskTodo(), 将涉及多主体资源冲突的任务通过总任务箱 Arbi_P.t_TodoBox[] 进行横向的任务比较, 最后决策出由哪个设计主体优先使用资源, 并选择执行任务 p[].now_tasktodo, 从而解决复杂的资源冲突问题. 经过前两个步骤的自主选择和二者协商选择后, 此时的总任务箱 Arbi_P.t_TodoBox[] 的可执行任务序列均为涉及到资源冲突的设计主体的任务序列 p[].TBox[] 的集合.

④仲裁选择任务阶段完毕后, 任务选择进入补充选择阶段. 在前三步的选择阶段均未能选择执行任务的设计主体 p[i], 已经没有足够的资源执行当前步长 rt 的任务. 因此 p[i] 检查是否还有足够的备选资源, 尽管使用备选资源 spare_resource 可能导致工作效率降低. 调用函数 p[i].If_Spare_Resource(), 如果返回 true, 表明剩余资源还充足, 可以使用; 调用函数 p[i].Sel_Task_UsingSpareRe, p[i] 在利用剩余资源的前提下选择当前步长 rt 的执行任务 p[i].now_tasktodo.

4 个步骤的的任务选择过程完成后, 仿真步长 rt 的任务选择阶段结束, 选择了执行任务的设计主体 p[i] 准备进入任务的执行阶段.

(3) 设计主体执行任务过程的行为函数调用任务选择阶段结束后, 选择了任务的设计主体 p[i] 进入执行任务阶段. 任务执行的流程如图 5 所示.

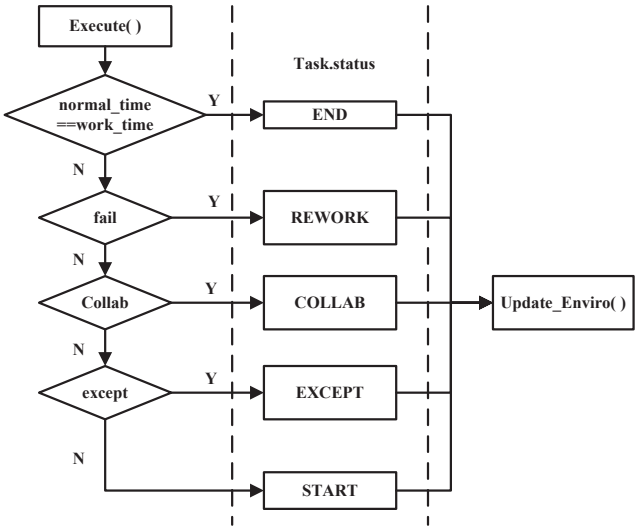

图 5 任务执行流程图

设计主体 p[i] 执行任务的整个过程是在函数 p[i].execute() 完成的. 进入函数 p[i].execute() 后, p[i] 首先判断任务的额定工作时间是否等于实际工作时间, 即 if(normal_time==work_time), 如果相等, 则 p[i] 在步长 rt 时刻, 完成此任务 p[i].now_tasktodo, 任务 T[p[i].now_tasktodo] 的状态设为 “END”. 如果 normal_time 与 work_time 不等, 则任务未完工. 引入随机函数 random(), 得出数值 m_value; 将 m_value 与参数初始化时设定的参数 t[task_id].fail_prob(任务失误率)、t[task_id].collab_prob(任务协作率)、t[task_id].except_prob(任务异常率) 进行比较 (task_id 为任务 p[i].now_tasktodo 的编号). 当 m_value 满足 t[task_id].fail_prob 时, 任务出现失误, 需要返工, 任务状态 t[task_id].status 设为 “REWORK”; 当 m_value 满足 t[task_id].except_prob 时, 此时设计主体 p[i] 需要协作才能继续完成此任务, 任务状态 t[task_id].status 设为 “COLLAB”, 并产生协作临时任务 t[collab_temp], 调用函数 p[i].Tcollab_tempSet(), 设置临时任务 t[collab_temp] 的相关参数; 当 m_value 满足 t[task_id].except_prob 时, 任务出现异常, 任务状态 t[task_id].status 设为 “EXCEPT”, 并产生异常临时任务 t[except_temp], 调用函数 p[i].Texcept_tempSet(), 设置临时任务 t[except_temp] 的相关参数; 如果 m_value 均不满足上述的参数, 任务正常执行, 任务状态 t[task_id].status 继续为 “START”. 设计主体 p[i] 对任务状态 t[task_id].status 设置完毕后, 根据不同的状态, 调用函数 p[i].Update_Enviro(), 更新相关的环境对象属性, 以进行下一步长 rt++ 的仿真运行. 环境对象属性更新完成后, 设计主体 p[i] 的任务执行阶段结束. 仿真系统则按上述的流程进行下一步长 rt++ 的仿真运行, 直到完成要求的仿真次数 a.

4 仿真实验

4.1 实验背景

本文以某发动机产品的开发项目为例, 该产品开发项目的任务流程如图 6 所示, 项目的重要参数如表 4 至表 6 所示.

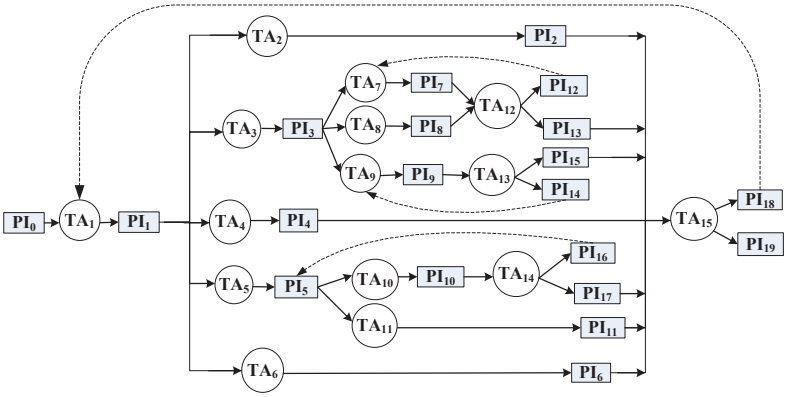

图 6　某摩托车发动机开发流程图

表 4　人员 - 任务 - 资源分配方案及任务额定执行时间

任务	设计主体	t_i	资源 种类	资源 数量	任务	设计主体	t_i	资源 种类	资源 数量	任务	设计主体	t_i	资源 种类	资源 数量
TA_1	P_1	4	R_4 R_5	3 1	TA_6	P_5	30	R_3 R_5	2 2	TA_{11}	P_1	5	R_2 R_6	2 2
TA_2	P_3	5	R_2 R_4	2 2	TA_7	P_7	25	R_1 R_2 R_3 R_5	1 2 2 2	TA_{12}	P_6	1	R_6	4
TA_3	P_8	5	R_2 R_3 R_6	2 2 2	TA_8	P_8	7	R_4 R_5 R_6	2 1 2	TA_{13}	P_9	1	R_6	2
TA_4	P_2	6	R_1 R_2 R_4	3 1 1	TA_9	P_4	5	R_3 R_4 R_6	2 1 1	TA_{14}	P_2	1	R_1 R_4	2 3
TA_5	P_6	30	R_1 R_5 R_6	1 2 1	TA_{10}	P_9	15	R_1 R_3 R_4 R_5	1 1 2 1	TA_{15}	P_4	4	R_1 R_6	2 4

表 5　资源的类型与数量

正常资源 (数量)	$R_1(5)$	$R_2(4)$	$R_3(5)$	$R_4(5)$	$R_5(8)$	$R_6(3)$
备选资源 (数量)	$R_7(2)$	$R_8(3)$	$R_9(2)$	-	-	$R_{10}(2)$

"-" 表示原正常资源没有备选资源

表 6　仿真参数设置

产品信息的初始状态 $PI()$	$PI_0 = Known; PI_j = Unknown(j = 1, 2, \cdots, 19); TYPE(PI_j) = Rework(j = 13, 15, 17, 19);$ $TYPE(PI_j) = Initial(j = 1, 2, \cdots, 12, 14, 16, 18, 20);$
学习效应因子 $decrease$	$redo\_decrease = 0.8, except\_decrease = 0.8$
协作率: α	$\alpha = 2\%$
协作率: β	$\beta = 2\%$
失误率: γ	$\gamma = 2\%$
返工率: λ	$\lambda(TA_1) = 2\%; \lambda(TA_5) = 2\%; \lambda(TA_7) = 2\%; \lambda(TA_{10}) = 2\%; \lambda(TA_{11}) = 2\%$
协作伙伴响应	全局响应

4.2　实验设计

为了为该项目选择适合的仲裁策略来消解过程中的资源冲突和协作伙伴冲突, 本文共设计 9 组实验, 如表 7 所示.

表 7 仲裁策略选择的实验设计

实验名称	协作伙伴选择冲突消解策略	资源冲突消解策略
实验 A	次优策略	等待策略
实验 B		次优策略
实验 C		效用策略
实验 D	等待策略	等待策略
实验 E		次优策略
实验 F		效用策略
实验 G	效用策略	等待策略
实验 H		次优策略
实验 I		效用策略

4.3 实验结果及分析

经过 100 次仿真之后, 9 组数据的统计分析及项目周期均值如表 8 所示, 资源冲突消解策略为备选策略时某次仿真主体行为如图 7 所示.

表 8 统计结果分析一览表

(I) 组别	(J) 组别	均值差 $(I-J)$	项目周期均值 $(a=100)$(天)	标准误差	显著性
A	B	9.800*	66.87	1.35042	.000
	C	11.650*		1.35042	.000
B	A	–9.800*	57.07	1.35042	.000
	C	1.850		1.35042	.172
C	A	–11.650*	55.22	1.35042	.000
	B	–1.850		1.35042	.172
D	E	7.520*	71.72	1.59488	.000
	F	8.500*		1.59488	.000
E	D	–7.520*	66.87	1.59488	.000
	F	0.980		1.59488	.539
F	D	–8.500*	63.22	1.59488	.000
	E	11.650*		1.59488	.539
G	H	11.880*	66.86	1.27628	.000
	I	12.330*		1.27628	.000
H	G	–11.880*	54.98	1.27628	.000
	I	0.450		1.27628	.725
I	G	–12.330*	54.53	1.27628	.000
	H	–0.450		1.27628	.725
D	A	4.850*	71.72	1.27286	.000
	G	4.860*		1.27286	.000
A	D	–4.850*	66.87	1.27286	.000
	G	0.010		1.27286	.994
G	D	–4.860*	66.86	1.27286	.000
	A	–0.010		1.27286	.994
E	B	7.130*	64.2	1.59529	.000
	H	9.220*		1.59529	.000
B	E	–7.130*	57.07	1.59529	.000
	H	2.090		1.59529	.191
H	E	–9.220*	54.98	1.59529	.000
	B	–2.090		1.59529	.191
F	C	8.000*	63.22	1.35317	.000
	I	8.690*		1.35317	.000
C	F	–8.000*	55.22	1.35317	.000
	I	0.690		1.35317	.610
I	F	–8.690*	54.53	1.35317	.000
	C	–0.690		1.35317	.610

* 表示有显著差异.

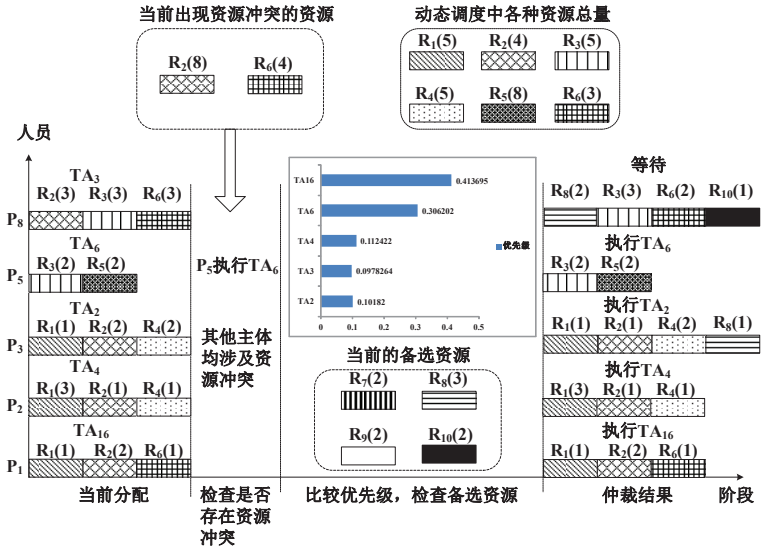

图 7　某次仿真中主体行为 (备选策略)

由图 7 可知: 当前, 设计主体 P_1, P_2, P_3 和 P_8 在资源 R_2 上发生占用冲突, 而设计主体 P_1 和 P_8 在资源 R_6 上发生占用冲突, 资源冲突涉及的设计主体大于 2, 仲裁主体调用相应的策略进行资源冲突消解. 仲裁主体将首先比较各个任务的优先级, 并查看备选资源情况, 将采用备选策略进行仲裁. 协作临时任务 TA_{16} 在当前步长具有最高的优先级, 具有资源的优先占用权, 所以仲裁主体将 R_2 与 R_6 均优先分配给设计主体 P_1; 另外, TA_4 的优先级较高, 可占用 R_2, 此时资源 R_2 与 R_6 的数量不足以支撑任务 TA_2 和 TA_3, 备选资源将被启用, 设计主体 P_3 使用备选资源 R_8 执行 TA_2, 而由于 R_2 的备选资源 R_8 数量不足, TA_3 将无法执行, 设计主体 P_8 在下一仿真步长将等待.

由表 8 可知, 当协作伙伴选择冲突消解策略一定时, 采用完全等待策略作为资源冲突消解策略时的项目周期最长, 而采用完全替代策略和替代效用策略时的项目周期无显著差异, 这表明, 该产品开发项目中, 由使用备选资源造成的时间损失较低, 也就是匹配度损失率较低, 远小于等待正常资源释放的时间. 当资源冲突消解策略一定时, 采用等待策略作为协作伙伴选择冲突消解策略时的项目周期最长, 而采用次优策略和效用策略时的项目周期无显著差异, 这表明, 该产品开发项目中, 由使用次优人员作为协作伙伴造成的时间损失较低, 也就是匹配度损失率较低, 远小于等待首选协作伙伴的时间. 针对该产品开发项目, 在动态任务调度过程中, 采用次优策略或者备选策略作为协作伙伴选择冲突消解策略以及同时采用完全备选策略和替代效用策略作为资源冲突消解策略能达到缩短项目周期的目的. 另外, 为了减少动态任务调度中计算过程, 在实际的协同产品开发过程中, 若出现类似情况, 则直接采用次优策略和完全备选策略作为过程管理中的仲裁策略.

5　结束语

本文在前期研究的基础上, 基于 CAS 理论和 MAS 理论建立了面向主体的协同产品开发过程动态仿真集成模型, 该模型以主体为核心, 打破了传统的以任务为中心的建模方式. 动态仿真系统实现了对协同产品开发过程中设计主体、仲裁主体、任务、资源以及信息的状态变化及交互作用的动态模拟, 并支持各类主体行为的过程分析. 仿真结果表明, 面向主体的动态仿真为协同产品开发过程的预测、评价和优化提供一条有效的途径, 并为研究过程中人员各类行为提供了新的思路.

参考文献

[1] 袁崇义. Petri 网原理与应用 [M]. 北京: 电子工业出版社, 2005.
Yuan Chongyi. Theory and application of Petri net[M]. Beijing: Electronic Industry Press, 2005.

[2] Steven D, Eppinger S D. Organizing the tasks in complex design projects[C]// NSF Design and Manufacturing System Conference, USA, 1992.

[3] Lin C S, Tang X Q, Duan G H. Analysis of Petri net model and task planning heuristic algorithms for product reconfiguration[J]. High Technology Letters, 2007, 13(3): 254–260.

[4] Danilovic M, Browning T R. Managing complex product development projects with design structure matrices

and domain mapping matrices[J]. International Journal of Project Management, 2007, 25: 300–314.

[5] 钱晓明, 唐敦兵, 楼佩煌. 基于 DSM 的产品并行开发过程仿真 [J]. 南京航空航天大学学报, 2007, 39(1): 88–93.
Qian Xiaoming, Tang Dunbing, Lou Peihuang. Simulation of product concurrent development process based on DSM[J]. Journal of Nanjing University of Aeronautics & Astronautics, 2007, 39(1): 88–93.

[6] 施国强, 李伯虎, 柴旭东. 基于设计结构矩阵的复杂产品开发项目规划模型 [J]. 计算机集成制造系统, 2007, 13(11): 2105–2109.
Shi Guoqiang, Li Bohu, Chai Xudong. DSM-based modeling of project scheduling for complex product development[J]. Computer Integrated Manufacturing Systems, 2007, 13(11): 2105–2109.

[7] 施国强, 李伯虎, 柴旭东. 基于着色 Petri 网的复杂产品开发多项目调度建模研究 [J]. 系统仿真学报, 2007(17): 3869–3872.
Shi Guoqiang, Li Bohu, Chai Xudong. Modeling of multi-project scheduling for complex product development based on CPN[J]. Journal of System Simulation, 2007(17): 3869–3872.

[8] 杨青, 吕杰峰. 基于 DSM 返工风险评价矩阵的项目优化与仿真 [J]. 系统工程理论与实践, 2010, 30(9): 1665–1671.
Yang Qing, Lü Jiefeng. Project optimization and simulation based on DSM rework risk evaluation matrix [J]. Systems Engineering — Theory & Practice, 2010, 30(9): 1665–1671.

[9] 杨青, 黄建美. 基于活动重叠的 DSM 项目时间计算及排序优化 [J]. 系统工程理论与实践, 2011, 31(3): 496–504.
Yang Qing, Huang Jianmei. Project time calculation and optimization based on DSM activities overlapping[J]. Systems Engineering — Theory & Practice, 2011, 31(3): 496–504.

[10] 李洪波, 徐哲. 考虑活动随机重叠和资源冲突的复杂产品开发流程仿真建模 [J]. 系统工程与电子技术, 2012, 34(7): 1412–1418.
Li Hongbo, Xu Zhe. Simulation modeling of complex product development process considering stochastic activities overlap and resource conflict[J]. Journal of Systems Engineering and Electronics, 2012, 34(7): 1412–1418.

[11] 陈海涛, 白凤, 房明民. 基于多 Agent 的城市应急管理通信机制研究 [J]. 情报科学, 2010, 28(12): 1884–1888.
Chen Haitao, Bai Feng, Fang Mingmin. Communication mechanism research of city emergency management based on multi-agent systems[J]. Information Science, 2010, 28(12): 1884–1888.

[12] 孙洁香, 潘福成, 尹作重. 基于多 Agent 的排产仿真系统研究 [J]. 制造业自动化, 2008, 30(22): 37–42.
Sun Jiexiang, Pan Fucheng, Yin Zuozhong. Research on task scheduling simulation system for multi-agent-based[J]. Manufacturing Automation, 2008, 30(22): 37–42.

[13] 陈成, 薛恒新, 张庆民. 基于本体与多 Agent 的可靠供应链网络设计模型 [J]. 计算机集成制造系统, 2011, 17(1): 142–150.
Chen Cheng, Xue Hengxin, Zhang Qingmin. Reliable supply chain network design model based on ontology and multi-Agent[J]. Computer Integrated Manufacturing Systems, 2011, 17(1): 142–150.

[14] 沈洁, 罗建利. 基于多 Agent 系统的分布式知识管理研究 [J]. 系统工程理论与实践, 2006, 26(1): 42–47.
Shen Jie, Luo Jianli. Researches on distributed knowledge management based on multi-agent system[J]. Systems Engineering — Theory & Practice, 2006, 26(1): 42–47.

[15] Garcia R. Uses of agent-based modeling in innovation/new product development research[J]. Journal of Product Innovation Management, 2010, 22(5): 380–398.

[16] 刘晋飞, 陈明, 姚远, 等. 基于多 Agent 的产品模块化协同设计策略 [J]. 计算机集成制造系统, 2011, 17(3): 560–570.
Liu Jinfei, Chen Ming, Yao Yuan, et al. Collaborative design strategy of product modularity based on multi-Agent[J]. Computer Integrated Manufacturing Systems, 2011, 17(3): 560–570.

[17] 蒋伟进, 张莲梅, 史德嘉. 复杂自适应系统的 MAS 动态协作任务求解时序逻辑模型 [J]. 系统工程理论与实践, 2012, 32(6): 1305–1313.
Jiang Weijin, Zhang Lianmei, Shi Dejia. The sequential logic model to solve the multi-agent dynamic cooperative tasks of complex self-adaptive system[J]. Systems Engineering — Theory & Practice, 2012, 32(6): 1305–1313.

[18] 约翰 ·H· 霍兰. 隐秩序适应性造就复杂性 [M]. 周晓牧, 韩晖, 译. 上海: 上海科技教育出版社, 2000.
Holland J H. Implicit order adaptive makes complexity[M]. Shanghai: Shanghai Science and Technology Education Press, 2000.

[19] 史忠植. 智能主体及其应用 [M]. 北京: 科学出版社, 2000.
Shi Zhongzhi. Intelligent agent and its application[M]. Beijing: Science Press, 2000.

[20] Zhang X D, Zhang S, Li Y Z, et al. Task scheduling behaviour in agent-based product development process simulation[J]. International Journal of Computer Integrated Manufacturing, 2012, 25(10): 914–923.

[21] Zhang X D, Luo L, Yang Y, et al. A simulation approach for evaluation and improvement of organizational planning in collaborative product development projects[J]. International Journal of Production Research, 2009, 13(47): 3471–3501.