第 12 篇　并 行 设 计

主　编　严隽琪
编写人　严隽琪
谢友柏
马登哲
唐　泉

第1章 绪 论

1 并行工程基本概念

并行工程是对产品设计和相关过程(包括制造过程和支持过程)进行集成,开展并行设计的一种系统化方法。换言之,并行工程是在产品设计阶段侧重于同时考虑产品全生命周期(从概念形成到产品回收或报废处理)中的各种主要性能指标,从而避免在产品研制后期出现不必要的返工与重复性工作。因此,并行工程有时表达为并行产品设计,有时表达为并行产品开发,在本章中认为两者是同一个含义。

并行工程曾有许多不同的名称:包括周期工程,生命周期工程,并行产品和过程设计,集成和协同设计,设计合成,生产率工程和系统工程等。无论叫什么,其基本含义都是把产品设计师和制造工程师结合起来用以改进制造过程和产品。它要求将不同的专业人员(包括设计、工艺、制造、销售、市场、维修等)组成开发小组,亦叫并行开发小组,以协同工作方式进行开放的和交互式的通信联系,以便缩短从概念设计至开始生产的生产准备时间,消除各种不必要的返工,使产品一次开发成功,获得优良的性能价格比。

将并行工程同传统的设计过程作一比较,见图12.1-1。传统的设计过程可以概括为"顺序工程"或"点到点"的设计。一个设计方案形成并通过后,送给另一个小组(或负责人)修改。每个责任人依次修改设计方案以满足自己的设计目标,产生另一个"点"。这种设计过程的缺点是,每个责任人都代表着特定的专业背景和企业中的某个局部责任,从自己的观点出发,而忽视了其决定对其他责任人的影响,有可能一个责任人作出的修改却不能为另一个责任人所接受,那么就得有初始的责任人重新修改设计方案,如此因相互冲突造成不必要的重复设计,设计效率低下,顺序的生产模式,还使得产品生命周期的各个环节(产品设计、生产、分配和维修)相对独立,其结果是工程决策缺乏远见并导致产品成本提高、质量下降。而且,由于信息在过程中是以顺序方式传递的,因此,要是在下游环节的工作中发现上游环节的工作需要改善,而这种滞后的信息反馈往往造成大的损失。

并行工程以并行的、交互的、协同的工作方式进行产品及其相关过程的设计。各环节在过程的开始阶段就尽可能地协同工作,信息流是双向的、比较即时地进行反馈。每个制造系统实质上都包括了众多的资源和设计活动,在产品开发过程中如能同时规划这些活动,将是企业实现并行运作的关键,这会使企业节约财富,提高产品质量。

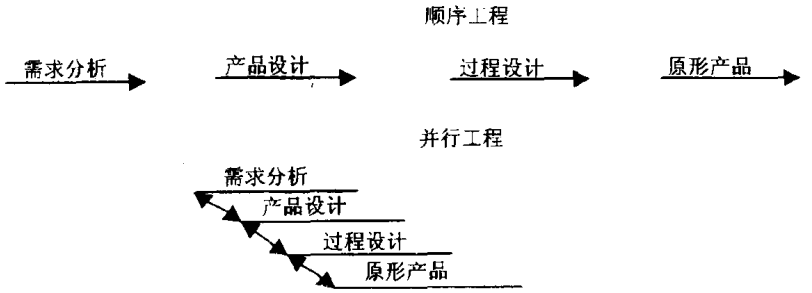


图12.1-1 顺序工程与并行工程的比较

并行工程的特点是:致力于满足用户需求,确信质量乃是改进过程的结果,并认为设计、生产和辅助过程的改进乃是整个企业永恒的责任。

一般说来,并行工程可包括四方面的特性:

①并行性:产品和设计过程,在同一个时间框架内进行。②约束性:过程约束性被认为是产品设计的一部分,从而保证设计出的零件易于制作、搬运和装配,并便于采用简单成本核算的工艺、工装和物料贮运方法。③协调性:产品和过程密切协调以获得有效成本、质量和交货期的最佳匹配。④一致性:产品和过程的重大决策要征求全组人员的意见并取得一致。由于上述"四性"的英文均以C开头,所以也称为并行工程的4C性。

采用并行工程这种工作模式,从产品设计的早期开始,由于注意到设计的规范性和制造的简单化,因此"全局成本意识"增强了。与以往的顺序生产方式相比,并行工程方式也许在产品开发的早期阶段投入了较高的资金,但整个产品开发和生产成本却会降得更低。不

仅如此，还有其他的优点，如产品质量和可靠性提高了。图 12.1-2 表示了产品的全局成本意识。图中下方的折线表示超过 80%的开发成本是处于过程下游的环节——生产阶段和运行支持阶段消耗掉的；而图中上方的折线却表示处于过程上游的环节——概念设计、初步设计和详细设计对产品最终价值的影响程度达到 90%。如进一步分析，可以看到，概念设计阶段的费用仅占总成本的 1%左右，却使产品价值的 70%成定局。因此，图中用虚曲线表示了修改产品设计的可能性从上游到下游是急剧减少的。因为到了下游阶段，任何细微的修改都需要付出大的代价。并行工程促使早期的产品概念设计和工程设计尽可能周到地考虑后续过程的要求，即可制造性、可检验性、可装配性、可维护性、可回收性等等。从而达到产品上市及时、质量好、成本低、易维护的目的。

图 12.1-3 以定量方式表示并行工程方式加速过程的优越性。图中的数字仅仅是一种假设，但它表示出在串行工程方式中，各个环节之间往往留有间歇的时间，有项目负责人们进行调整或修改设计，他们非固定地用这些间歇时间进行冲突的调解，交换想法以及向最新版本同一。而在并行工程方式中，间歇时间得到消除，生产者依靠计算机系统和产品数据管理软件（在下节将详细论述）保持项目的正常运行。项目管理的许多功能，包括项目调度、冲突的调解、文档工作、通信、赋权、信息流等均可在计算机上进行。当然这是一张理想化的示意图，因为工程师 A、B、C、D 的工作不可能是同时开始，又同时结束的，但至少他们可以阶梯型开始工作（如图 12.1-1 所示）。

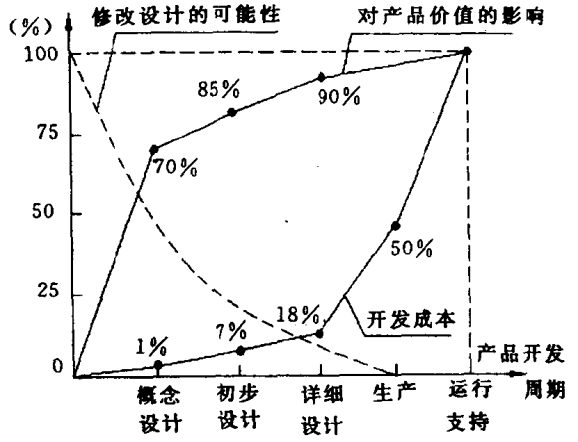
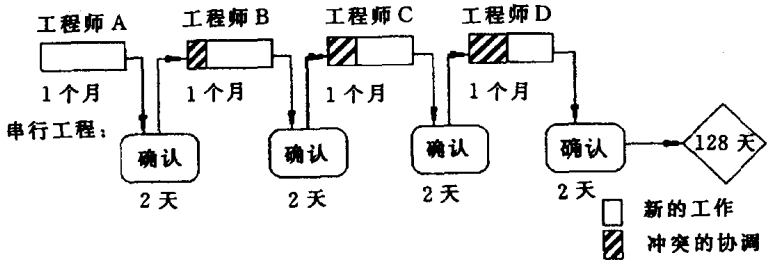


图 12.1-2　全局成本意识



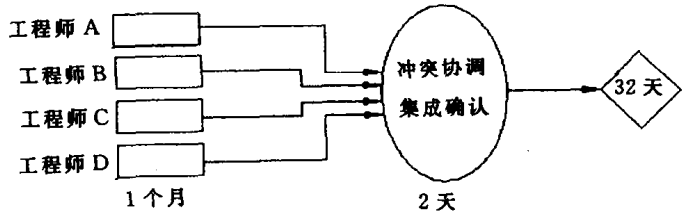


图 12.1-3　并行工程加速设计过程

2　并行工程方法学

并行工程以信息集成为基础，逐步向产品开发过程集成发展。它继承了以往制造技术中的很多精髓，同时又具有独特的理论和方法学，其要点为：

1）并行工程的效益来自组织模式改革、用户需求定义、设计环境建设和产品开发新技术应用等四个要素的综和平衡。

根据具体的产品对象以及市场的影响，上述四个要素及其相关关系都在不断地发生变化。只有达到优化意义上的平衡，并行工程才能发挥最大的效益。

2）从信息集成到产品开发过程集成需要逐级演进。

从信息集成发展到并行过程，经历了五个阶段：互操作的计算环境；互操作的工具和任务；自动化的数据管理；自动化过程管理；自动决策支持。从信息集成到产品开发过程的自动集成体现了重视知识、协作和人的有效参与，并且是从单纯的信息集成向人——管理

——技术三者的集成发展。

3) 组织模式的变革经历了三个阶段。

第一阶段是按功能部门划分的传统模式，如市场部、工程部、制造部、销售服务部等。第二阶段是按专业划分小组，如机械组、电子组、动力组等。波音777飞机的开发被认为是并行工程方式的成功范例，波音777的并行产品开发正是采用了这种模式。第三阶段是按产品机构划分的集成产品开发小组，对飞机而言，如机翼小组、机身小组、推进器小组等。在成功地开发了777后，波音公司1994年起开始实行集成产品开发小组，亦称多功能小组的方式。

3 并行工程的若干关键技术

并行工程要解决产品开发过程的描述(建模)、优化(重组)、控制(自动化)等方面的问题，而这几个方面都必须满足集成的需要。

(1) 开发过程的描述

美国国防部委托的高技术咨询小组1988年在研究并行工程框架的报告中提出将产品开发过程看作为一个协调的单一活动过程。这需要从概念上提高认识和从组织上得到保证，而技术上的描述，尚属研究的热点，首先要求从认识过程的抽象层次提出新的描述方法，其次要求在技术上的实现有所突破。传统上，我们用“工程图”作为定义产品的语言，因为它是面向工程技术人员的，所以被称为工程师的语言。在产品设计开发过程中，“工程图”为单一理解产品定义奠定了基础。但是，计算机不能理解“工程图”，无法实现自动化，因此需要机器可理解、处理的“语言”来定义产品，即产品模型。关于产品建模技术，将在第5章详述。

从信息处理或知识处理的角度看产品开发过程，实质上是单一的产品模型构造及模型的实例化过程。产品开发过程的自动化，包括产品的定义与实现定义过程的自动化。这里，产品定义的技术实现过程同样也需要机器可以理解、处理的语言来描述。随着设计工作的展开，设计者拥有一些实现产品定义过程的模型是很重要的。这些模型使设计者有可能运用已知的经验法则尽早获得可制造性等信息、评估与功能目标有关的设计方案的质量、估算产品的性能和费用，并能预测改变方案所产生的影响。这样的模型大致有三类：①工艺过程模型，对于评估各种产品及其工艺过程改变所产生的影响，最好是有制造工艺过程的模型，如金属切削、成形、注模、铸造、焊接等。理论上，这些模型应该能使用户很容易地以交互方式更改产品几何形状或过程参数，借助各种工程、几何、统计和科学分析方法，还能准确预测过程运行情况，把过程运行情况(如速度、温度和准确度)与产品测量值(如公差、物料完整性、强度和表面光洁度)直接联系起来。同时，再生产出产品前就可以估算产品成本、性能及可靠性。如果这些模型是建立在这种产品或类似产品的实际生产系统的真实测量值的基础之上，那么利用模型就可以准确评估在企业的实际生产厂内产品和工艺变化所产生的影响。②装配模型，目前有许多系统可用于评估装配设计的“质量”。三维CAD商品化软件常提供装配干涉检查的功能，对两个零件装配在一起时可能发生的对应性能的错误状态进行校验。另一些系统提供各种层次的装配成本预测和一种指导工具，这种工具基于“为装配而设计(DFA——Design For Assembly)”的启发式经验规则，能指挥设计者选择更容易装配的设计方案。还有的装配模型是基于具体位置和轨迹的几何模型，位置和轨迹都由人或机器人控制，以便实现所期望的装配。就并行工程的原理而言，重要的是这些模型是与制造工艺模型联系在一起的。③制造系统模型，评估制造系统的生产能力对于并行工程也很重要。生产能力即一定数量的加工设备和生产配备人员的投入，生产出一定数量的产品的能力。这些制造系统的模型既可以是分析模型，也可以是仿真模型。目前有一些适用于多设备、夹具、人员、刀具和物料贮运系统的简单系统的分析模型。但是，由于制造系统的复杂性，经常需要采用分析模型中的启发式规则或建立适用于更复杂情况的仿真模型。

(2)产品开发过程的优化

过程优化，必须满足产品上市快、质量好、成本低和易回收这一企业和社会期望的基本目标。过程的集成还不是最终的目的。通过优化以取得产品设计与设计过程满意的性能结果才是目的。优化内容主要涉及产品设计性能与产品可生产性的综合平衡。对过程优化的研究集中于设计方法学的研究、设计过程的质量控制以及过程优化决策的自动化。

在设计方法学的研究中，面向制造的设计(Design For Manufacturing，简称DFM)、面向装配的设计(Design For Assembly，简称DFA)等DFX技术，针对产品生命周期中的不同阶段对候选设计进行评估，是目前并行工程研究的一个十分活跃的领域。图12.1-4表达了设计性能在过程控制域中的映射问题。过程的“质量功能配置”(Quality Function Deployment，简称QFD)正是从过程的总体上反映了这一概念。

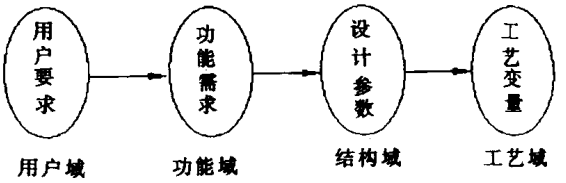


图12.1-4 产品开发过程领域映射图

质量功能配置是一种系统化的产品质量规划方法。它采用质量屋(House of Quality，简称 HOQ)的形式，通过定义“做什么”和“如何做”将顾客需求(Voice of Customer，简称 VOC)逐步展开、分层转换成为产品工程特性、零件特性、工艺计划和生产控制等，形成从用户需求到产品上市的连续转换和“实施流”以确保用户需求就是开发目标，加强了产品开发各阶段中项目组成员对用户需求及为满足用户需求所采取的措施的理解，从而提高了产品设计质量。它的特点是将注意力集中于规划和问题的预防上，而不是仅仅集中于问题的解决上，是实现产品质量保证的重要手段和工具。图 12.1-5 表示了质量屋的一般形式，它清晰明了地表达了各要素之间的相互关系。质量屋由五个组成、A 区是用户需求及权重矩阵，收集各种用户需求要素，并比较其重要程度(相对权重的总和为 1)，逐一填入该矩阵；B 区是工程特性矩阵，分析用户需求，确定影响用户需求的功能特性要素，填入该矩阵；C 区是用户需求与工程特性关系矩阵，逐一审核各功能要素与用户需求的关联程度，确定出在单一需求下各功能要素的相对重要性，填入对应的方格中；F 区是工程特性相关矩阵，用与检查工程特性各项之间的关联和冲突，如果某一项对应参数的增加将导致另一项对应参数的增加，则在工程特性相关矩阵中填入正相关符号(＋)，反之填入负相关符号(－)，据此可检查出各项之间是否有冲突，以及是否可以进行优化；D 区是评估结果矩阵，表示功能特征项的重要性程度，根据需求要素的权重和功能特征要素对其影响程度的比率确定；E 区是市场竞争力评估矩阵，收集主要竞争对手有关产品的信息，分别评估客户对各种需求的满意程度。该项表示了竞争对手满足各项用户需求的实力，以确定市场营销的策略。

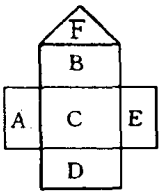


图 12.1-5　质量屋的一般形式

质量功能配置通常分为四个基本阶段，即产品规划、结构和零件设计、工艺规划、作业控制等，四个阶段的质量屋的结构形式类似。但产品规划质量屋中的工程特性及其权重可以作为结构设计阶段的需求输入，由此确定第二阶段为满足这些需求而采取的措施，构造出第二个质量屋。这样逐个将有关需求映射到产品生命周期各个阶段的有关技术和措施，形成一个瀑布式逐级分解过程，并能逐级回溯和审核。图 12.1-6 表示了产品生命周期 QFD 系列及产品需求间的映射过程。

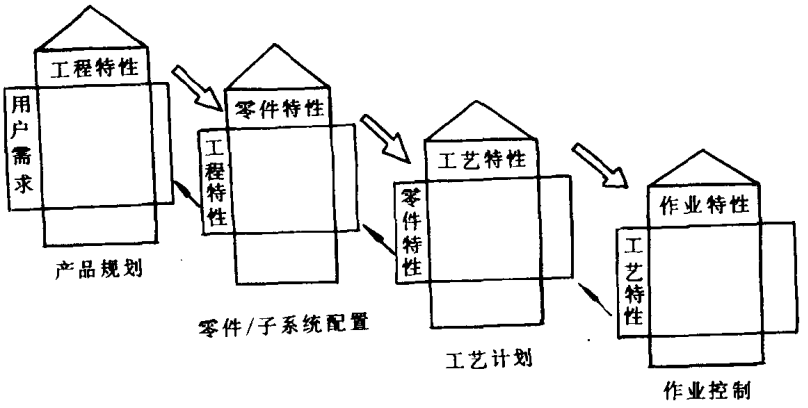


图 12.1-6　产品生命周期 QFD 系列及产品需求间的映射过程

这儿介绍的仅仅是质量功能配置最基本的原理及实施方法。实际应用中，由于许多信息的模糊性，要使信息处理科学化，还有待大量的研究工作。目前国内外学者的研究重点包括 QFD 模型，QFD 中定性信息的模糊处理，QFD 自动化实现技术。

(3) 过程的控制

过程控制包括了设计过程质量控制及过程优化决策的自动化。所谓质量控制，就是要求制成的产品与最初产品设计定义相吻合，尽量降低由于产品设计过程与后续过程间互相冲突而导致的产品定义修改等等，寻求能使早期产品设计达到尽可能完善的一种实现机制。这一点需要在早期产品设计过程中，应用前文提及的设计方法学，通过与后续相关过程共享产品模型信息，并作适当的决策才能达到。目前关于设计过程优化的决策，主要还只能依赖于专家的领域知识、经验和科学计算相结合。因此，产品的并行设计，也是人机交互和人工智能相结合的设计系统，由于多知识源之间互相关联、冲突、约束的关系，在决策过程中，协调冲突约

束的机制将成为系统运行的管理核心。过程控制应传递决策指令，提供调整系统运行状态的机制，评估系统现行运行状态。对控制的支持依赖于控制结构和网络通信，可包括：任务管理、工具存取和共享管理、多媒体信息交换、远程表示能力等。为此，发展上文所述的过程"质量配置"(QFD)的研究，发展计算机对协同工作支持的系统结构成为实现过程控制的基础任务的两个方面。如目前正处于发展中的"多代理人"(Multiagent)结构，实质上是模仿多个自主决策中心间互相协同工作概念的一种系统结构，正形成当前的研究热点。

4 并行工程的使能工具

有许多应用工具可视为并行工程使能工具，在实施并行工程中，如能正确选择和有效地应用这些工具，必将使企业获得降低成本、缩短开发周期、减少重复性工作和设计修改量的益处。并行工程的使能工具可大致分为四类：

(1) 全面质量管理

这类工具用于收集用户需求，将这些需求转变成具体的时间值、成本值与性能值，并监控整个系统建立的过程，以便最大限度地满足用户的需求。如设计标准化，使设计人员可以重复使用现有设计方案，同样的零件无须再进行重新设计，即使现有的设计不完全符合某种产品的设计要求，设计标准也将有助于设计人员确定相似的设计方案，从而大大减少设计工作量。又如被认为是质量方法学上一大贡献的塔古其(Taguchi)方法，把产品和工艺过程设计分为三个阶段，即系统设计、参数设计和公差设计。系统设计是找寻新概念、新思路、新方法及最佳适用技术的阶段，向用户提供新的或经过改进的产品设计；参数设计可以选择参数的最优级；而公差设计是制定制造公差，力求以最低费用改善质量。塔古其方法需要对于如何选择统计方法进行实验和分析实验结果的问题进一步进行研究。这类工具还包括质量功能配置(QFD)、成本质量分析、统计过程控制等。

(2) 计算机集成制造工具

这类工具用于辅助开发和评估各种有关复杂产品及其工艺设计方案，并支持项目小组交流这些方案，如产品数据管理系统可以作为并行产品设计的集成平台，可以从数据共享、开发过程管理、协同工作、与应用工具集成等方面作为集成平台支持并行设计；产品数据交换标准提供了可互操作的产品模型。又如网络及数据通信，使各个领域(设计、分析、制造和测试等)通过先进的通信设备可快速地传递数据，使许多用户可以即刻享用同样的信息，并可以基于网络进行合作设计。又如价值工程，这是集成企业内数据资源、工程资源和制造资源的极好工具，它追求以最低成本开发出满足功能要求的产品和服务。这类工具还包括各种计算机辅助系统，如 CAD、CAE、CAPP、CAM、仿真、分析评估等等使能工具。

(3) 准时生产能力改进工具

这类工具用于寻求改进产品及其工艺过程的最佳方法，从而改善加工能力及缩短生产周期。这类工具包括了面向装配的设计工具、面向制造的设计工具及其他 DFX 工具。

图 12.1-7 表示了上面所述三类使能工具在并行工程各阶段的使用情况。通常，大多数使能工具可在多个阶段中使用。对于一个具体项目而言，确定各阶段使用什么使能工具很关键。选择适用的工具的工作应由所有参与项目的成员预先商定并列入计划中。

	需求确定	方案评估	产品/过程详细设计	安装和开始运行
全面质量管理				
设计标准		●	□	□
质量功能标准	●	□	□	□
过程控制计划		□	●	□
成本质量分析			●	□
价值工程		●	□	□
统计过程控制				
计算机集成工程和制造				
仿真和分析		●		
CAD/CAM/CAE		●		
计算机辅助工艺设计			●	□
成组技术	□	●	□	□
有限元建模		●		
电子数据转换	□	●	□	
选择和评估方法		●	□	□
实体建模		●	□	□
及时生产能力改进				
装配设计		□	●	□
可加工性设计		□	●	□
生产能力设计		□	●	□
周期分析		□		●

● 主要应用 □ 次要应用

图 12.1-7 使能工具在并行工程各阶段使用情况

(4) 人的系统

并行工程是以人为中心的设计，成功实施并行工程的重要因素在人，所以这类使能工具也层出不穷。如项目管理、通信、人工智能应用等等。

值得一提的是，实施并行工程的方法很多。各企业在具体项目实施中采用的组织形式、管理方法及技术手段均不相同，应根据实际情况逐步推广。在并行工程实施过程中，也期待有更多适合我国实际运行环境的

使能工具出现。

5 并行设计实例

汽车电气电缆互联设计

汽车内的电气电缆互联和装配是长期困扰生产厂家的技术难题，因为它跨越了电气系统设计和机械设计两个截然不同的领域。而在汽车的设计过程中，这两者又是互相结合、密不可分的。传统的设计过程是一个串行过程(图 12.1-8)，当汽车车身和零部件的设计和组装到位以后，才能开始电气互联的设计和装配。由此带来两个方面的问题：一是设计中存在的问题往往发现得太晚，整个汽车产品的开发周期被大幅度延长；二是电气设计师和机械设计师互不关联，各自的产品在装配过程中必然会发生问题，经常需要更改设计，使整个产品的开发成本大幅度上升。如果电气和机械两部分的设计不能紧密结合在一起，碰到的还将不仅是产品上市时间的推迟，产品上市后，厂家还会遇到产品保修的麻烦问题。

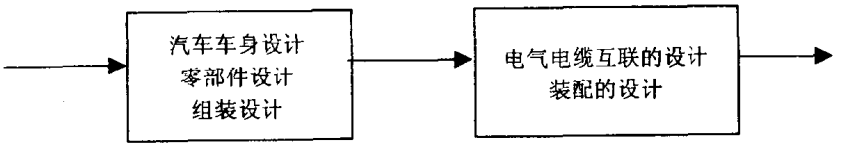


图 12.1-8 机电串行设计

因此，美国的 CAD 供应商提供了一种自动化设计工具，这是基于 CAD 软件的混合建模与并行装配的设计技术，使得机械设计师和电气设计师可同时设计一个汽车产品的装配模型，互相参照地完成各自的设计任务，并在三维数字化模型上完成电缆线与零部件之间的干涉检查、设计规则检查、电缆线物性计算。由于设计不合理而带来的扭曲以及装配不合理带来的磨损都将被避免。这种并行工程方式使机械设计和电气设计融合为一体，使厂家在产品开发周期和可靠性方面赢得竞争优势，见图 12.1-9。

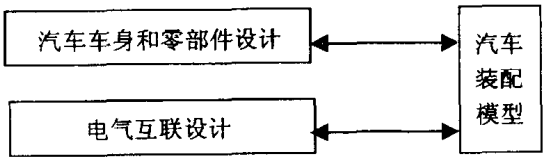


图 12.1-9 机电并行设计过程

第 2 章　并行产品开发过程建模及冲突的预消解

1　并行产品开发过程模型

1.1　并行产品开发过程的定义

产品开发过程是指某产品从(形成)概念到正式投产之前这一工作流程。

支持并行工程的产品开发过程，简称并行产品开发过程(Concurrent Product Development Process，CPDP)，是多功能小组在计算机软硬件工具和网络通信环境的支持下，考虑产品及其全生命周期中有关信息，以缩短产品开发时间，提高产品开发一次成功率为目标而设计的开发活动流程。

在此定义中，跨学科的多功能小组和支持环境是并行产品开发得以顺利进行的基本保证。活动流程是一系列逻辑相关的活动的集合，是并行产品开发过程的核心。活动间的信息流关系、多功能小组的组织关系以及资源约束等是设计该流程必不可少的背景资料。换言之，对并行产品开发过程的完整描述除了开发活动流程外，还应辅以对产品数据、组织关系和资源约束的描述。

1.2　并行设计过程的基本元素

提炼基本元素是实现对并行产品开发过程进行形式化描述的基础。现将活动(Activity)、成员(Person)、角色(Role)、资源(Resource)和产品数据(Product Data)作为其基本元素，其中活动隐含了功能目标与约束。

1.2.1　活动(Activity)

产品开发中的活动有狭义与广义之分。狭义的活动基本上等同于任务，可以描述为“对某对象的处理”，这儿采用的是广义的活动的概念，它在定义时需要同时指定组织、资源和数据三个方面的内容。表达为“某成员(以某角色的名义)，用某种资源，完成对某数据对象的处理”。进一步地，可给出如下定义：

活动是指在特定的目标驱动、组织形式和资源保证下完成的对产品数据的处理。

更为一般地，可将活动 A 抽象为输入流、控制机制、支持机制和输出流组成的四元组，并表示为

$$A = (In, Con, Sup, Out) \quad (12.2\text{-}1)$$

其中，In：活动的输入流，表示执行活动所必备的输入数据。一个活动可以有若干个输入流，它们可以来自数据对象或其他任务的输出流。

Out：活动的输出流，表示活动执行后所产生的输出数据。一个活动可以有若干个输出流，他们来自数据对象。作为输出，他们可供应给其他活动。

Con＝｛Con 1，Con 2，…，Con m｝表示活动的控制机制(如触发机制)，它隐含了功能目标与约束。

Sup ＝｛Sup 1，Sup 2，…，Sup n｝表示活动的支持机制，它包含组组织对象和资源对象。

上述定义可用面向对象的表达方式描述如下：

```
Class Activity {
int Activity _ Id                          //活动号
string Activity _ Name                     //活动名称
string Activity _ Description              //活动描述
int Activity _ Status     //活动当前所处的状态，分为三种：未执行、执行中和完成
Function Execute _ Controler (Controler 1, Controler2,...)          //活动的控制机制
Function Execute _ Controler _ Relationship (Relationship)          //活动控制机制间的关系
Function Execute _ Person ( Person _ Id1,Person _ Id2,...)          //活动的执行人员
Function Execute _ Role ( Role _ Id1, Role _ Id2, ...)              //活动的执行角色
Function Resource ( Resource _ Id1, Resource _ Id2,...)             //活动所需的资源
Function Activity _ Data _ In ( Data _ Id1, Data _ Id2, ...)        //活动的输入数据
Function Activity _ Data ( Data _ Id1, Data _ Id2, ...)             //活动处理的数据
Function Activity _ Data _ Out ( Data _ Id1, Data _ Id2, ...)       //活动的输出数据
...
}
```

具有以下特性：

层次性和分解性：层次性使得处于高层的活动对其下层活动具有广泛的指导和约束能力；而分解性表明一个复杂的设计活动可以进一步分解为更具操作性的相对简单的活动。

组织和资源关联性：对一定的组织和资源而言，只有限定在组织和资源边界内的活动才可能是有效活动，超出该边界，活动就失去了意义。

1.2.2　成员(Person)

作为活动的执行主体，组织中的人员，即组织的成员，在并行产品开发过程中扮演着极其重要的作用。一般地，它所具有的属性可用面向对象的表达方式描述如下：

```
Class Person {
int Person _ Id                          //成员号，一般为工号
string Person _ Name                     //成员名称
string Person _ Description              //有关该成员的说明资料
Function Person _ In _ Team ( Team _ Id1, Team _ Id2,... )          //成员所在的小组
Function Person _ In _ Project (Project _ Id1, Project _ Id2,... )  //成员所参与的项目
Function Person _ In _ Department (Department _ Id)                 //成员所属的部门
Function Person _ Is ____ Role (Role _ Id1, Role _ Id2,... )        //成员所承担的角色
...
}
```

在上述模型中，有“Role(角色)”属性，它相当于一般企业中的岗位。将成员与角色有机地结合起来，构成组织结构的核心部分，是并行产品开发过程组织结构的重要特点之一。

1.2.3　角色(Role)

角色是对组织专业化程度的一种规定，是组织结构的基本单位之一，它必须具备两种属性：

横向专业化：对角色所具有的专业知识的规定，是角色执行具体任务的基础。

纵向专业化：对角色所具有权利的规定，也就是对通常所说的管理职能的规定。

角色作为组织中专业化程度的规定，是一种抽象的概念，它必须由具体的成员来承担，成员是角色的载体。一个成员可以承担多种角色，一个角色也可以由多个成员来承担。为了更明确地表示角色这一概念，可以给出如下定义：

概念角色：当一个角色尚未由具体的成员来承担时，称该角色为概念角色。

实体角色：当一个角色由具体的成员来承担时，称该角色为实体角色。

概念角色，在企业中一般表现为岗位空缺。一旦空岗得到填补，即组织中的成员承担了角色，则该角色就转化为实体角色。

概念角色和实体角色的定义将组织设计分为两个阶段：概念角色设计阶段和实体角色设计阶段。概念角色设计阶段根据问题的求解需要设计相应的求解角色，而实体角色设计阶段将概念角色分配给具体的成员来承担，形成实体角色。

角色是体现组织动态性的重要因素，在并行产品开发过程中发挥着不可或缺的作用。角色的介入把原本动静相杂的复杂问题简化为一静（角色关系矩阵）、一动（成员角色关系矩阵）两个问题，使组织能够以相对稳定的基本结构来应对人事的变化，提高了组织的动态适应性。对角色的上述认识可以描述如下：

```
Class Role {
int Role _ Id                            //角色号
string Role _ Name                       //角色名称
string Role _ Description                //角色说明，如对专业水平的要求，职责范围，仲裁权利级别等等
string Role _ Type                       //当前角色类型，分为概念角色和实体角色两类
Function Role _ Person _ Can (Person _ Id1, Person _ Id2,... )      //可胜任该角色的成员
Function Role _ In _ Project (Project _ Id1, Project _ Id2,... )    //角色所属的项目
Function Role _ Person _ Is (Person _ Id1, Person _ Id2,... )       //承担该角色的成员
(针对实体角色)
...
}
```

1.2.4　资源(Resource)

资源描述了执行某活动所需的物品、工具或信息等等。广义的资源定义可用图 12.2-1 表示。

由于并行产品开发过程是以人为主体的设计活动，人在并行工程中占有重要的地位，人力资源作为组织构成的基础而在上节以成员与角色的方式予以描述；信息资源既是触发产品开发活动的主要因素，又是活动结束的主要成果，故将其列入产品数据而在下节详述。因此，在这一节中并行产品开发所需要的资源主要为：

硬件设备：指产品开发所需的各种硬件，如计算机、绘图仪等；

软件工具：指产品开发所需的各种通用工具(如 CAD、CAE、Office 等)和专用支持工具(如油泵仿真设计系统等等)。

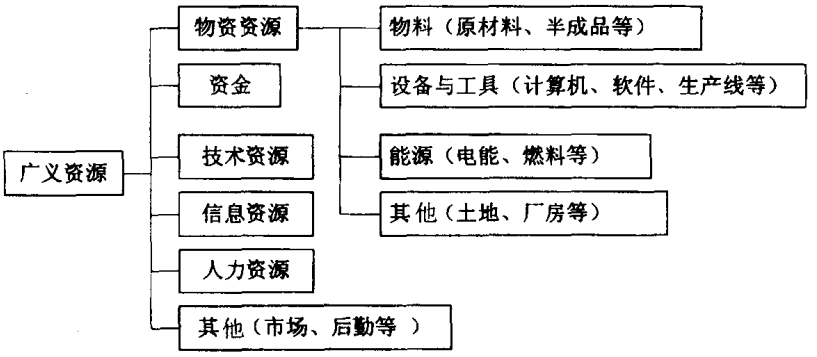


图 12.2-1　广义资源定义

并可给出如下定义：

并行产品开发过程中所涉及的资源是指保证整个产品开发过程能够顺利进行的硬件设备和软件工具的类型和属性的集合。

设资源空间为 Re，可描述为

$$Re = (Tr, fr) \tag{12.2-2}$$

其中　Tr——资源的论域，对于不同的资源类型，其论域是不同的；

fr(·)——论域的属性，也就是某种资源类型所对应的属性，如完好程度、占用情况等等。

在并行产品开发过程中，更为关心的是资源的竞争性问题。对竞争性资源在难以及时补充的情况下，要结合开发活动作好规划与调度。

上述定义可用面向对象的表达方式描述如下：

```
Class Resource {
int Resource _Id                          //资源号
string Resource _Name                     //资源名称
string Resource _Description              //资源说明
int Resource _Volume                      //资源可用数量(对竞争性资源而言)
Function Resource _Type( type1, type2)    //资源类型, type1={开发设备,开发工具}
// type2={竞争性资源,非竞争性资源}
...
}
```

1.2.5　产品数据(Product Data)

产品数据指在并行产品开发过程中所需要的或产生的与产品相关的各种类型的中间数据和最终数据，这些数据具有以下特点：

数据与活动紧密关联：从开发过程的角度看，产品数据与开发活动之间具有紧密的耦合关系，即一方面，开发活动导致了新的产品数据的产生，或旧产品数据的状态或版本的变化，或视图的变化；另一方面，产品数据又是促使产品开发活动开始和表示产品开发活动结束的标志。定义并维护二者之间的有机联系，以达到数据与过程的融合在并行产品开发中显得尤为重要；

数量大，类型多：产品开发，特别是复杂的产品开发将产生大量的不同类型的数据。这些数据可以分为描述开发对象的一般产品数据(如产品模型、BOM 表、设计文件、计算书、工艺文件和 NC 程序等)、产品开发的过程数据(如任务单、通知书、会签意见、变更和修改通知单等)和产品支持数据(如各种技术标准、规范、标

准件和通用件数据等)。它们以结构化或非结构化的形式存在于产品开发的不同阶段中,并能够被不同的开发活动所共享。这些数据对开发活动提供不同层次上的支持。

需要良好的组织:上述庞大、繁杂的数据必须以某种良好的组织方式组织起来,即建立不同数据之间的连接关系,才可能给开发活动提供良好的支持。

产品数据的这些特点和由此导致的需求正是产品数据管理(Product Data Management,PDM)产生的背景。目前,主流的PDM软件都提供了良好的电子仓库服务功能(Vault Service,包括数据存储、检索、修改、浏览及版本管理等)、产品结构管理功能(Structure Management,如以BOM为核心的数据管理)、产品配置管理功能(Configure Management,提供不同产品结构的不同视图和描述),使得PDM成为并行产品开发中不可或缺的支撑平台。

关于产品数据管理更为详细的介绍见第5章,此不赘述。

基于以上对产品数据的认识,可将其要点描述为:

```
Class Product _ Data {
int Product _ Data _ Id                    //产品数据号
string Product _ Data _ Name               //产品数据名称
string Product _ Data _ Type               //产品数据类型
string Product _ Data _ Description        //产品数据说明
string Product _ Data _ Creator            //产品数据的产生者
string Product _ Data _ Status             //产品数据的状态,如产生中,校核、审批、归档等
string Product _ Data _ In _ Vault         //产品数据的存储
...
}
```

1.3　并行设计过程的基本视图

1.3.1　并行设计过程基本元素之间的关系——子视图

本节将元素与元素之间的相互关系表达为各种子视图。这些关系可见表12.2-1。

由于元素之间的关系是双向的,因而该表具有对称性。本表只标出上三角部分。

某些元素之间的关系可由其他元素之间的关系推导而来,本表中用"+"来表示这种关系。如"活动"与"成员"的关系可由活动与角色的关系(活动分配矩阵)和角色与成员的关系(成员角色矩阵)推出,表示为"活动分配矩阵+成员角色矩阵"。

在活动元素中已明确定义了活动的输入数据、处理数据和输出数据,即产品数据与活动的关系已包含在活动之中,因此资源与产品数据的关系也就可由资源与活动的关系而得到。

显然,只需定义11种关系(即子视图)就可明确表示出并行产品开发过程五种基本元素之间的相互关系。这11个子视图绝大多数以矩阵的形式表示,其中数据关系矩阵就是产品数据模型。

这11个子视图的分类与组合就构成了过程的基本视图集,基本视图集包括活动流视图、组织视图、资源视图和产品数据视图。

由此,得到并行产品开发过程的基本视图集及其所包含的子视图如表12.2-2所示。

表12.2-1　产品开发过程基本元素之间的关系

元素	活动	成员	角色	资源	产品数据
活动	活动网络图	活动分配矩阵 +成员角色矩阵	活动分配矩阵	资源需求矩阵	(包含在活动中)
成员		IPT(集成产品开发团队)结构图	成员角色矩阵	资源使用权限矩阵 +成员角色矩阵	数据访问权限矩阵 +数据存储矩阵 +成员角色矩阵
角色			角色关系矩阵	资源使用权限矩阵	数据访问权限矩阵 +数据存储矩阵
资源				资源关系矩阵	(资源需求矩阵)
产品数据					数据关系矩阵

表 12.2-2　并行产品开发过程的基本视图集

基本视图集	所　含　子　视　图
活动流视图	活动网络图
组织视图	IPT 结构图、角色关系矩阵、成员角色矩阵、活动分配矩阵、数据访问权限矩阵、资源使用权限矩阵
资源视图	资源关系矩阵、资源需求矩阵
产品数据视图	数据存储矩阵、数据关系矩阵

1.3.2　活动流视图

活动与活动之间的关系，构成了活动流视图。正如活动是并行产品开发过程基本元素的核心，活动流视图也是并行产品开发过程基本视图的核心。

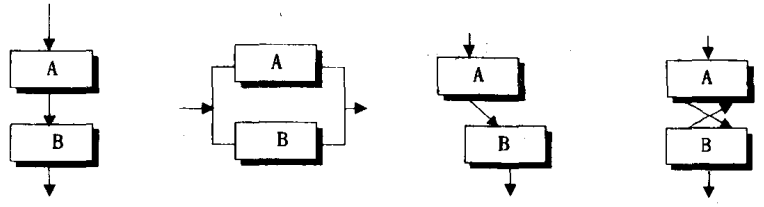
活动流是描述整个并行产品开发过程全部活动集合上的一个关系。

即设活动集合为 A，则活动流 R 是

$$R=\{(x,y)\mid x\in A,\ y\in A\} \quad (12.2\text{-}3)$$

其中，$x\notin\phi$，$y\notin\phi$。

在并行产品开发过程中，各活动在空间、时间或逻辑上存在着不同的联系。对于不同联系类型的活动在处理上需要采取不同的方式。现从并行产品开发过程建模的需要出发，将活动关系分为以下四类，见图 12.2-2。



a) 串行关系　　b) 平行关系　　c) 并行关系　　d) 交互耦合关

图 12.2-2　活动关系分类

其中

串行关系表示一个活动必须至少得到另一个活动的最后输出信息后才能开始，表现为活动串行开展；

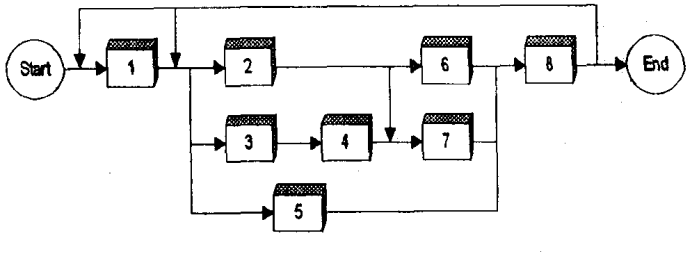
平行关系表示若干个活动之间相互独立，无信息交互，表现为活动可同时开展；

并行关系表示一个活动在得到另一个活动的部分输出信息后就可以开始（该部分信息不是在活动的最后阶段产生），表现为活动异步开展；

交互耦合关系表示活动间存在双向的信息联系，即活动 A 的进行需要活动 B 的信息，同时，活动 B 的进行也需要活动 A 的信息，表现为需经过多次的迭代、反复才能完成。

必须强调的是，活动之间的关系并不是绝对的，而会随着活动粒度的划分变化而变化，这就为活动重组从而提高并行度提供了理论基础。

活动间复杂关系的表达一直是一个难题。传统的项目管理一般采用确定型活动网络图，但这种网络图中不允许有任何强连通成分，故只能表达出活动之间的先后时序关系，而不能反映活动间的并行、交互及反馈关系。还可采用随机型网络来表达并行产品开发过程的活动流程，这种网络节点允许自环和回路，并允许出现概率分支（可进行方案优选）。图 12.2-3 为用这种方法表示的喷油泵开发过程的活动流视图．该活动流视图中，如活动 3 与活动 4 是串行关系，而活动 2 与活动 3、活动 4 是平行关系。



活动 1：柱塞偶件关键参数设计
活动 2：出油阀设计
活动 3：柱塞偶件设计
活动 4：柱塞弹簧设计
活动 5：喷油泵其它零件设计
活动 6：出油阀弹簧设计
活动 7：凸轮设计
活动 8：喷油泵装配仿真

图 12.2-3　喷油泵开发过程的活动流视图

1.3.3　组织视图

相对于工业时代的宝塔型组织结构而言，信息时代组织结构的特征是扁平化，以及由此而来的对市场变化的快速反应。在这种背景下，开发组织的组成策略以及组成形式成为现代产品开发过程中提高产品开发质量，缩短产品开发周期的重要因速。此外，组织构成不能够仅仅被视为一个组织成员的确定过程。作为并行产品开发的一个环节，组织设计还需要与产品结构、产品开发活动等进行有机的协调和综合。这意味着，组

织的构成还应考虑到产品开发活动、产品数据、开发资源等约束。因此，一个较为完整的组织视图应包括成员与成员之间的关系、角色与角色之间的关系、成员与角色之间的关系、角色与活动之间的关系、角色与资源之间的关系、以及角色与产品数据之间的关系。

(1) 成员与成员之间的关系——集成产品开发小组(IPT)

为适应日趋激烈的市场竞争，一种被普遍认同的观点是，开发组织的结构将由传统的强调分工和隶属关系的递阶组织结构向动态的、生物结构型的组织方式转变。较为理想的动态组织结构是扁平式组织结构，但就目前的并行工程环境而言，比较适合的是项目—一职能型组织，波音公司在开发波音 737-X 时就采用了这种组织结构，取得了较大的成功，他们将这种组织形式称为集成产品开发小组(IPT：Integrated Product Team)。这种组织结构既保留了传统的静态的职能部门，又能根据项目组成动态的项目工作小组，很适合目前状况下的产品并行开发模式。图 12.2-4 为 IPT 的组织结构图。

从图中可以看到，左边是由相对固定的职能部门组成的静态组织，它只是形式上的组织，实际操作则按照右边的动态组织。企业领导通过项目开发部直接领导各个项目小组的负责人。各个项目小组的负责人则领导各个项目小组。这种动态的组织结构有以下特点：

打破部门界限，减少控制层次，组织结构趋于扁平化；

把不同部门的专业人员组织在一起，信息容易沟通，有助于激发员工积极性，发挥专业特长。

对某个项目，便于迅速集成，快速响应，可适应复杂多变的市场环境。

企业对主管甚至每个员工授权增大，便于灵活决策，提高效率。

项目组随产品生命周期的开始和结束而建立和解体，体现了组织的动态性。

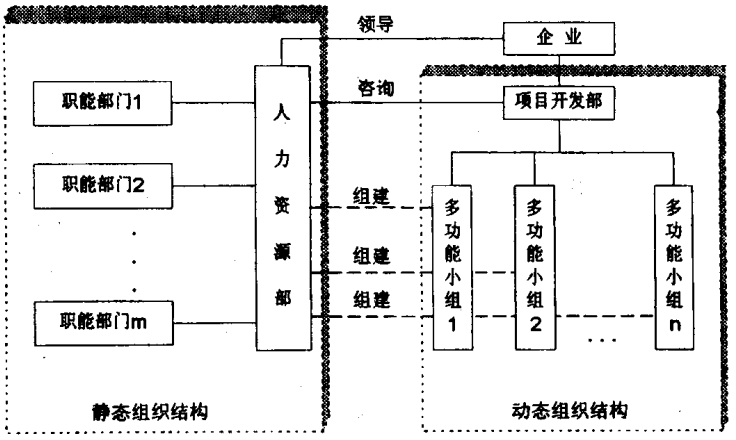


图 12.2-4　IPT 的组织结构图

(2) 角色与角色之间的关系——角色关系矩阵 $M_{Ro\text{-}Ro}$

角色与角色之间的关系——角色关系矩阵是组织视图的一个核心内容。角色作为对组织专业化程度的一种规定，必须具有两种属性：横向专业化和纵向专业化。其中，纵向专业化是对角色所具有权利的规定，也就是通常所说的对管理职能的规定，这种规定是一个角色对其上级角色进行服从，对其可控角色进行控制和协调以及对其对等角色进行通讯的基础。角色关系矩阵实际上就是对各个角色的纵向专业化属性进行归纳和抽象的结果，如图 12.2-5 所示。其表达式为

$$(M_{Ro\text{-}Ro})_{ij}=\begin{cases}sp & \text{角色 i is \_ superior \_ of 角色 j}\\ jn & \text{角色 i is \_ junior \_ of 角色 j}\\ pt & \text{角色 i is \_ partnre \_ of 角色 j}\\ / & \text{角色 i 与角色 j 无直接关系}\end{cases} \tag{12.2-4}$$

	Role		
Role	Role-ID, Name	Role-ID, Name	...
Role-ID, Name	/	sp	
Role-ID, Name	jn	/	
...			

图 12.2-5　角色关系矩阵 $M_{Ro\text{-}Ro}$

(3) 成员与角色之间的关系——成员角色矩阵 $M_{P\text{-}Ro}$

角色作为组织中专业化程度的规定，是一种抽象的概念，它必须由具体的成员来承担，成员是角色的载体。一个成员可以承担多种角色，一个角色也可以由多

个成员来承担。成员与角色之间的关系可以用成员角色矩阵 $M_{P\text{-}Ro}$ 来表示，如图 12.2-6 所示，其表达式为

$$(M_{P\text{-}Ro})_{ij}=\begin{cases}1 & \text{第 i 个成员可以承担第 j 个角色}\\ 0 & \text{第 i 个成员不可以承担第 j 角色}\end{cases} \tag{12.2-5}$$

	Role		
Person	Role-ID, Name	Role-ID, Name	...
Person-ID, Name	0	1	
Person-ID, Name	1	1	
...			

图 12.2-6　成员角色矩阵 $M_{P\text{-}Ro}$

此处的“可以”包含两层含义：①该成员有能力承担该角色；②小组同意该成员承担该角色，即小组对该成员进行了授权。

角色及成员角色关系矩阵的定义使得原本与人直接相关的问题（如分工、权限等）都转变为与角色相关。我们知道，在并行产品开发过程中，角色是相对稳定的，而成员则由于生病请假或辞职等原因而会有一定的变动。因此，角色的介入把原本动静相杂的复杂问题简化为一静、一动两个问题，使组织能够以相对稳定的基本结构来应对人事的变化，提高了组织的动态性。

（4）角色与活动之间的关系——活动分配矩阵 $M_{Ro\text{-}A}$

活动与角色之间的关系可由活动分配矩阵 $M_{Ro\text{-}A}$（图 12.2-7）来表示。其表达式为

$$(M_{Ro\text{-}A})_{ij}=\begin{cases}1 & \text{第 i 个角色承担第 j 个活动}\\ 0 & \text{第 i 个角色未承担第 j 个活动}\end{cases} \tag{12.2-6}$$

	Activity		
Role	Activity-ID, Name	Activity-ID, Name	...
Role-ID, Name	1	1	
Role-ID, Name	1	0	
...			

图 12.2-7　活动分配矩阵 $M_{Ro\text{-}A}$

（5）色与产品数据的关系——数据访问权限矩阵 $M_{Ro\text{-}V}$

在并行产品开发过程中，每个小组成员对不同的产品数据拥有不同的访问权限，如查询、修改、复制、删除、归档等等。在组织结构中应定义角色对数据的访问权限，小组成员根据成员角色矩阵中的授权而获得相应的权限。

需要说明的是，产品开发过程中涉及到的数据量很大，如果针对每一种数据定义访问权限，即使是可行的，也非上策。在 PDM 中定义了电子仓库（Vault）这一概念，它是一种逻辑上的划分。因此，本文就通过定义每种角色对不同 Vault 的访问权限，来达到对 Vault 中的数据进行访问和控制的目的。相应的数据访问权限矩阵见图 12.2-8，其中，T（Transfer）表示可在 Vault 间传递数据。其表达式为

$$(M_{Ro\text{-}V})_{ij}=\begin{cases}Q & \text{第 i 个角色可查询第 j 个 Vault}\\ M & \text{第 i 个角色可修改第 j 个 Vault}\\ D & \text{第 i 个角色可删除第 j 个 Vault}\\ B & \text{第 i 个角色可归档第 j 个 Vault}\\ / & \text{第 i 个角色不可访问第 j 个 Vault}\end{cases} \tag{12.2-7}$$

	Vault		
Role	Vault-ID, Name	Vault-ID, Name	...
Role-ID, Name	Q	M	
Role-ID, Name	Q	/	
...			

图 12.2-8　数据访问权限矩阵 $M_{Ro\text{-}V}$

（6）角色与资源的关系——资源使用权限矩阵 $M_{Ro\text{-}Re}$

在并行产品开发过程中，每个小组成员对资源都拥有不同的使用权限，如拥有（Own）、可使用（Use）、不可使用（/）等。在本文的组织结构中，只定义角色对资源的使用权限，小组成员根据成员角色矩阵中的授权而获得相应的权限。资源使用权限矩阵如图 12.2-9 所示。其表达式为

$$(M_{Ro\text{-}Re})_{ij}=\begin{cases}O & \text{第 i 个角色拥有第 j 个资源}\\ U & \text{第 i 个角色可使用第 j 个资源}\\ / & \text{第 i 个角色可不可使用第 j 个资源}\\ \vdots\end{cases} \tag{12.2-8}$$

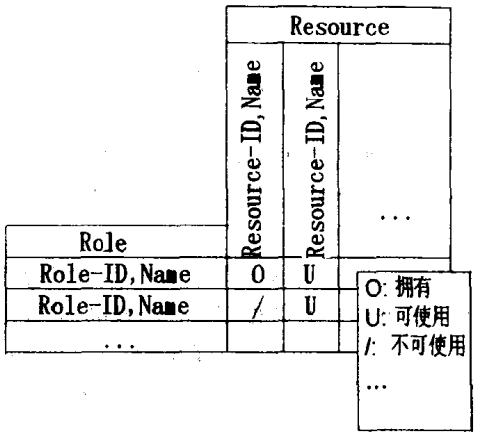


图 12.2-9　资源使用权限矩阵 $M_{Ro\text{-}Re}$

综上所述，一个完整的组织视图应包括(图 12.2-10)：IPT 结构图，成员角色矩阵，活动分配矩阵，数据访问权限矩阵，以及资源使用权限矩阵。

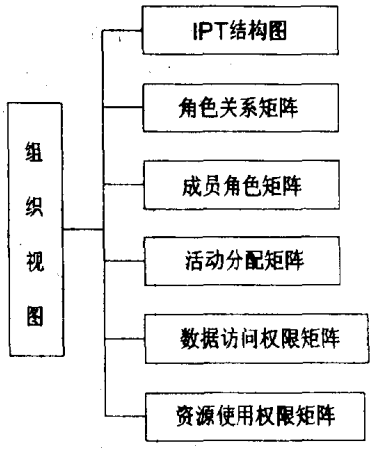


图 12.2-10　组织视图

1.3.4　资源视图

并行产品开发过程基本元素中，资源与资源之间的关系及活动与资源之间的关系共同构成了过程的资源视图。

(1) 资源与资源之间的关系——资源关系矩阵 $M_{Re\text{-}Re}$

已知并行产品开发过程中的资源分为硬件设备和软件工具两大类。软件工具必须借助于一定的硬件设备才能起作用，而硬件设备中，如绘图仪与计算机之间也同样存在这种相关性。本文称资源之间的这种关系为资源的依赖关系。进一步地，可将这种依赖关系分为强依赖关系和弱依赖关系。

若某特定资源与另一个特定资源必须同时使用，则称它们之间的关系为强依赖关系，如某型绘图仪与其驱动程序之间的关系。

若某特定资源与另一个特定资源之间虽然相互依赖，但并非必须同时使用，则称它们之间的关系为弱依赖关系，如某绘图仪与某计算机之间的关系。

不同资源之间的这种关系可用资源关系矩阵 $M_{Re\text{-}Re}$ 来表示，如图 12.2-11。其表达式为

$$(M_{Re\text{-}Re})_{ij}=\begin{cases}0 & \text{资源 i 与资源 j 之间没有依赖关系}\\ 1 & \text{资源 i 与资源 j 之间是弱依赖关系}\\ 2 & \text{资源 i 与资源 j 之间是强依赖关系}\end{cases} \tag{12.2-9}$$

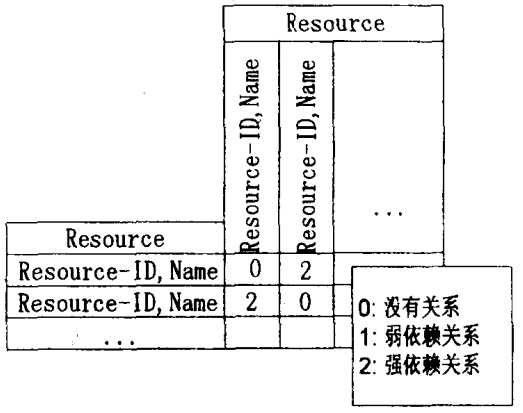


图 12.2-11　资源关系矩阵 $M_{Re\text{-}Re}$

(2) 活动与资源之间的关系——资源需求矩阵 $M_{A\text{-}Re}$

活动必须由小组成员借助一定的资源才能完成，即活动对特定资源存在需求关系。这种关系可用资源需求矩阵 $M_{A\text{-}Re}$ 来表示，见图 12.2-12。其表达式为

$$(M_{A\text{-}Re})_{ij}=\begin{cases}1 & \text{第 i 个活动需要第 j 个资源}\\ 0 & \text{第 i 个活动不需要第 j 个资源}\end{cases} \tag{12.2-10}$$

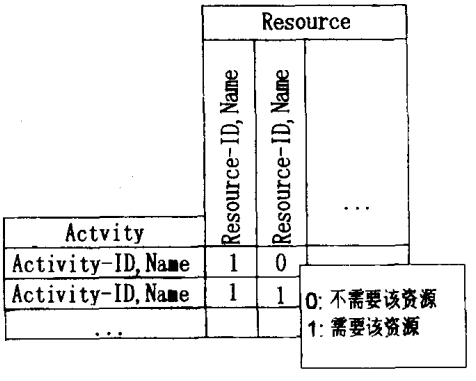


图 12.2-12　资源需求矩阵 $M_{A\text{-}Re}$

由于资源与资源之间的关系在资源关系矩阵中已得到定义，因此，在资源需求矩阵中只需注明活动的关键资源，特别是竞争性资源，其它一般性资源可由资源关系矩阵推出。

1.3.5　产品数据视图

产品数据视图负责对产品数据的组织和管理，为了达到这一目的，它需要回答两个基本问题：产品数据放在何处，产品数据与产品数据之间具有怎样的关系。

(1) 数据存储矩阵

定义产品数据的存储位置(物理的或逻辑的)是对产品数据进行管理的基础。在创建一个产品数据时，应对其处于不同状态时(如产生中，校核、归档等)的存储位置做出规划，因而数据的存储矩阵是动态变化的。

(2) 数据关系矩阵

产品数据与产品数据之间的关系因数据类型的不同而有多种形式。一般地，产品数据可分为两大类：产品结构数据(如表示产品结构信息的各种图形数据)和产品描述数据(如设计文件等字符型数据)。它们之间的关系可用数据关系矩阵描述。由于产品数据视图是产品数据管理的主要研究内容，本节对其将不作更深入的展开。

1.4　并行设计过程的递阶集成多视图模型

为了全面表达并行产品开发过程，特在上述这些视图的基础上，构造一个递阶集成多视图模型(The Hierarchical Integration Multi _ views Model，简称 HIMM)(图 12.2-13)。该模型具有以下特征：

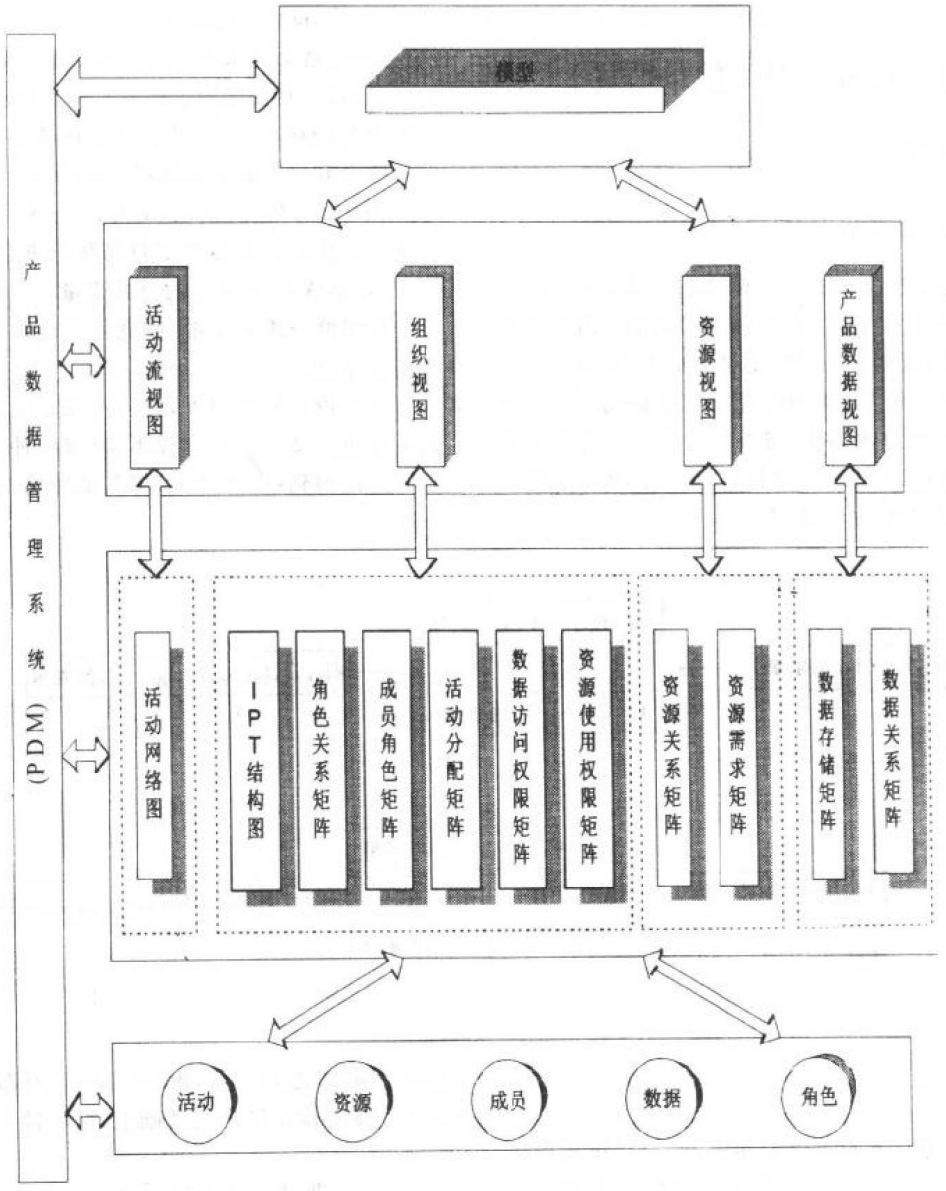


图 12.2-13　并行产品开发过程的递阶集成多视图模型