外计算机科学经典教材

Mc Graw Hill Education

Object-Oriented Technology
From Diagram to Code with Visual Paradigm for UML

面向对象技术
——使用 VP-UML 实现图到代码的转换

Curtis HK Tsang
(美) Clarence SW Lau 著
Ying K Leung
杨明军 译

清华大学出版社

国外计算机科学经典教材

面向对象技术

—— 使用 VP-UML 实现图到代码的转换

Curtis HK Tsang
Clarence SW Lau
Ying K Leung
著

杨明军
译

清华大学出版社
北京

Curtis HK Tsang，Clarence SW Lau，Ying K Leung
Object-Oriented Technology From Diagram to Code with Visual Paradigm for UML
EISBN：0-07-124046-2

图书在版编目(CIP)数据

面向对象技术——使用 VP-UML 实现图到代码的转换/(美)常(Tsang,C,H.) 等著；杨明军 译. —北京：清华大学出版社，2007.2

书名原文：Object-Oriented Technology From Diagram to Code with Visual Paradigm for UML

国外计算机科学经典教材

ISBN 978-7-302-14115-0

Ⅰ. 面… Ⅱ. ①常… ②杨… Ⅲ. 面向对象语言，UML—程序设计—教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2006)第 134542 号

责任编辑：王　军　徐燕萍
装帧设计：孔祥丰
责任校对：成凤进
责任印制：孟凡玉
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　**邮购热线**：010-62786544
投稿咨询：010-62772015　**客户服务**：010-62776969
印 刷 者：北京市世界知识印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：21.5　**字　数**：446 千字
内附光盘 1 张
版　　次：2007 年 2 月第 1 版　　**印　次**：2007 年 2 月第1 次印刷
印　　数：1～4000
定　　价：46.00 元

出 版 说 明

近年来，我国的高等教育特别是计算机学科教育，进行了一系列大的调整和改革，亟需一批门类齐全、具有国际先进水平的计算机经典教材，以适应我国当前计算机科学的教学需要。通过使用国外优秀的计算机科学经典教材，可以了解并吸收国际先进的教学思想和教学方法，使我国的计算机科学教育能够跟上国际计算机教育发展的步伐，从而培养出更多具有国际水准的计算机专业人才，增强我国计算机产业的核心竞争力。为此，我们从国外多家知名的出版机构 Pearson、McGraw-Hill、John Wiley & Sons、Springer、Thomson 等精选、引进了这套“国外计算机科学经典教材”。

作为世界级的图书出版机构，Pearson、McGraw-Hill、John Wiley & Sons、Springer、Thomson 通过与世界级的计算机教育大师携手，每年都为全球的计算机高等教育奉献大量的优秀教材。清华大学出版社和这些世界知名的出版机构长期保持着紧密友好的合作关系，这次引进的“国外计算机科学经典教材”便全是出自上述这些出版机构。同时，为了组织该套教材的出版，我们在国内聘请了一批知名的专家和教授，成立了专门的教材编审委员会。

教材编审委员会的运作从教材的选题阶段即开始启动，各位委员根据国内外高等院校计算机科学及相关专业的现有课程体系，并结合各个专业的培养方向，从上述这些出版机构出版的计算机系列教材中精心挑选针对性强的题材，以保证该套教材的优秀性和领先性，避免出现“低质重复引进”或“高质消化不良”的现象。

为了保证出版质量，我们为该套教材配备了一批经验丰富的编辑、排版、校对人员，制定了更加严格的出版流程。本套教材的译者，全部由对应专业的高校教师或拥有相关经验的 IT 专家担任。每本教材的责编在翻译伊始，就定期不间断地与该书的译者进行交流与反馈。为了尽可能地保留与发扬教材原著的精华，在经过翻译、排版和传统的三审三校之后，我们还请编审委员或相关的专家教授对文稿进行审读，以最大程度地弥补和修正在前面一系列加工过程中对教材造成的误差和瑕疵。

由于时间紧迫和受全体制作人员自身能力所限，该套教材在出版过程中很可能还存在一些遗憾，欢迎广大师生来电来信批评指正。同时，也欢迎读者朋友积极向我们推荐各类优秀的国外计算机教材，共同为我国高等院校计算机教育事业贡献力量。

清华大学出版社

国外计算机科学经典教材

编审委员会

前　言

现在计算机科学专业和软件工程专业的学生都必须选取像面向对象分析与设计或者面向对象软件工程之类的课程作为本科学业的一部分。但是，尽管市面上已经有各种各样的面向对象技术方面的图书，但是似乎缺少一本内容全面、能够涵盖整个软件开发过程的教科书：从建模、分析到实现，也就是从理论到实践。因此，学生们对面向对象这项强大技术的认识常常是不完整的，最糟糕的是一些学生甚至对它有非常大的误解。

导致这个现象的原因可能归咎于以下几个因素。第一，UML 表示法非常丰富，很多学生不知道如何系统地使用它们。第二，统一过程没有规定在不同条件下需要什么模型；实际上一些模型甚至工作流只是可选的。第三，市面上的相关图书没有提供完整地涵盖整个开发生命周期的实用内容，学生在将所学理论应用到实际环境中时会遇到麻烦。第四，没有一个统一的方法能够应用于任意类型的问题。学生甚至从业者往往倾向于遵循某种“经过验证的”方法学，但最终他们都会发现这些方法学根本不能用于他们的系统。

最基本的问题是很多学生和从业者不知道软件开发方法有 3 个关键元素：过程、表示法和技术，以及应该如何将其系统地加以应用，以一种系统化的方式有效地利用开发系统(环境)提供的便利。大多数讨论 UML 的图书都只关注表示法，也可能会有少量关于统一过程的内容。很多关于 UML 的图书都包含了软件工具，但是这些工具只能用来演示如何能够将软件开发过程自动化。软件厂商通常只在他们的手册中讨论自己的工具，可以理解的是，他们会对自己所采用的技术有所偏好。但是，很多产品与 UML 标准并不兼容。

本书的起因是由于需要一本涵盖整个软件开发生命周期的书，它能够在构建大型系统所涉及的各个步骤中为学生和从业者提供指导。读者可能来自不同的开发阶段，从建模与分析阶段到实现阶段，通过使用 CASE 工具 VP-UML，他们将体会到“从图到代码”功能的威力。每章都有一个小型案例研究，这可以帮助读者理解在实践中如何应用这些知识。我们还提出了视图校正技术(View Alignment Techniques，VAT)框架，它可以为不同类型应用的方法定制提供便利。基于 VAT 框架，我们将描述活动分析方法(Activity Analysis Approach，AAA)，它特别适合于交互密集型系统的开发。在本书中，软件开发的实用方面将通过一个获奖的 CASE 工具 VP-UML 得到演示。

我们的梦想是使大型软件系统的开发成为一个简单容易的任务。我相信所提出的 VAT 框架将有助于从业者和学生建立他们自己的方法学，以适合各自的需要，从而使失败和恐惧远离系统开发。

目　录

chapter

1 引 论

1.1 概论

对于任何软件项目，其中最重要的就是交付系统的可靠性，并能够满足客户的需求和期望。然而，很多权威报告显示，很大一部分已交付的系统只是简单地交付，而没有真正使用起来。对于大型项目或者当系统开发过程中涉及较大开发团队的时候，这个问题尤为敏感。

在过去的几年中，为了解决这个问题，人们提出了很多软件开发方法。然而，并没有一种合适的技术或者启发式法可用来指导设计师灵活地使用这些开发方法，并根据不同的环境进行裁剪。因此，设计师倾向于严格按照这些方法行事，导致所交付的系统并不能满足客户的需求。

本章着重讨论软件开发领域中的几个常见问题，并讨论用于系统分析和设计的面向对象方法。本章将描述软件开发方法的 3 个组件，同时还包括表示系统、过程、技术和本书采用的 CASE 工具。

1.2 本章要点

通过本章的学习，可以掌握如下几点：

- 描述面向对象方法给软件开发带来的好处；
- 讨论软件开发中 3 个关键组件的作用；
- 在一个较高的层面上描述统一建模语言(Unified Modeling Language，UML)、统一过程(Unified Process，UP)、UML 视图校正技术(View Alignment Techniques，VAT)和 Visual Paradigm for UML 技术(Visual Paradigm for UML，VP-UML)；
- 理解本书内容组织。

1.3 软件工程方法

开发可靠软件是一个高成本的劳动力密集型行业。已经有无数关于软件项目失败的报告，因此软件开发是一项高风险的投资。在过去的几十年间，软件工业经历了快速的增长，这使得大型软件系统的开发对规范方法的需求变得非常突出。今天，开发者采用已验证的软件工程方法和完善的项目管理实践来保证所构建的软件不但满足客户的功能需求，还要及时交付，并且不会超出项目预算。

因为软件是不可见的，所以找出软件中存在的所有 Bug 是相当困难的。实际上，不管准备投入多少资源和精力，完全没有 Bug 的复杂软件系统只能是软件开发者的梦想。虽然人们必须接受这类系统不可能完全没有 Bug 的事实，但是必须要考虑到软件故障所造成的后果。现在，软件系统广泛应用于政府、商业组织和日常活动的方方面面，支撑着这些系统的平稳运行。因此，软件系统故障总会影响到人们的生活，并可能引发很多灾难，甚至关系到生命安全。因而，质量是软件工业的一个非常重要的问题。解决这个问题的最常见的办法是采用已验证的过程来开发软件系统。

尽管质量问题非常重要，但是开发大型软件系统面临的最大挑战是开发这些项目所需时间很长，系统需求总是会由于各种原因发生改变。经常发生这样的事情，即在项目开始的时候，客户并不清楚他们具体想要的是什么。这样一来，当交付项目的时候，客户可能会发现系统并不像他们预期的那样。技术的快速变化本身也可能是个问题。如果项目开发时间很长，那么在项目开发期间可能会发生技术变更的情况。项目经理经常左右为难，一方面不得不修改系统设计以采用新技术，但是另一方面会导致费用超出预算，并且延长了开发时间，或者构建了一个无关紧要的系统，仅仅是完成了交付，但是系统并没有真正用起来。

开发大型系统的另一个困难是开发过程通常会涉及一个由众多专业人员构成的团队，这些专业人员都是各自领域的专家，因此，团队成员之间进行高效地沟通是极其重要的。实际上，很多时候项目失败的首要原因是不良沟通和人员因素问题，而非技术问题。

软件工程将系统化的、规范的、高质量的方法应用到软件开发、运行和维护过程中去。软件工程建立在良好的工程概念之上，并且实际上很多人将软件工程规范开发比作建筑行业，已经从以人工活动为主转变为细化的工业流程。

与其他工程规范一样，软件工程师在真正实现系统之前要构建软件系统的模型。在软件开发过程中，建模是一项非常重要的活动，通常在最终设计和实现软件系统之前，软件工程师要花费很多时间在不同的抽象层次上开发模型。模型是一种高效的沟通手段，特别是在那些不需要详细信息的场合。例如，通常使用高度抽象的拓扑图来表示运输系统中的行车路线。在软件系统中，不同的涉众(Stakeholder)总是需要物理系统的不同方面的信息。例如，乘客需要知道车费和某条公交路线的停靠站

位置；公交车司机则需要某项特定公交车服务精确的路线信息；公交车站管理员需要知道所有从本站发出以及到达本站公交车的时间表。为了迎合这些涉众的不同需要，需要为他们创建不同的模型，具体如下：

为乘客建立的模型。可以用一条直线并在上面画一个圆圈来表示，给出公交车停靠站名字，可能还有相关的车费。

为公交车司机建立的模型。它可以是一张显示公交车服务覆盖区域的简化路线图。为了向司机提供更多的信息，该图中还包括街道名称和道路。

为汽车路线规划者建立的模型。这个模型中可能包含各个汽车路线的路径构成的详细道路图。每条汽车路线的路径都被标记出来，并用不同的颜色显示。

一个模型可以包含一个或者多个视图，每个视图表示系统的某个特定方面。例如，乘客模型包含了车费视图和路径视图。车费视图为某条路线上的不同停靠站提供车费信息，而路径视图则提供包含相关街道的路线信息。基于系统不同视图的模型必须是一致的。例如，建筑物的三维模型必须与同一建筑物的不同正视图(模型)保持一致。进而应该能够使用某种合适的表示法(语言)来表达模型，涉众能够理解这个表示法(Notation)。对于软件开发而言，可以使用3个正交的视图来适当地描述系统：

- 包含软件系统中数据转换的功能视图；
- 包含系统结构以及与之相关的数据的静态视图；
- 包含软件系统中事务顺序或者过程的动态视图。

广义地讲，有两大基本的软件开发方法：即结构化方法和面向对象方法。前者自20世纪70年代就非常流行，因为传统的过程式语言为其提供了足够的支持。随着20世纪90年代面向对象编程语言(比如C++和Java)的发明，面向对象方法已经变得越来越流行。

可以按照对系统的不同视图进行建模的方式及其相关过程，对这两种软件开发方法进行比较。结构化方法以系统的功能视图为中心，在开发的不同阶段使用不同的模型。当从一个开发阶段进行到下一阶段时，当前阶段的模型也将转变到下一阶段的模型。结构化方法有3大弱点：

首先，因为结构化方法以系统的功能视图为中心，所以当系统的功能发生改变时，系统的分析、设计模型以及实现都要发生很大的变化。

其次，在结构化方法中，只要早期创建的模型发生改变(因为需求发生改变或者纠正前面的错误)，就需要进行模型变换工作。在分析阶段，数据流图(Data Flow Diagram，DFD)用来系统建模，系统由一组函数构成，函数之间有数据流。在设计阶段，系统被建模成结构图，由函数层次结构组成。如果系统的功能发生改变，就需要重新来一遍，即重新执行一遍分析和设计阶段，而在这些阶段需要投入相当多的时间和精力。

第三，在结构化方法中基本上不存在动态视图。DFD 由两个视图构成：功能视图和静态视图。图形界面的使用以及软件系统日益增加的复杂度使得动态视图变得越来越重要。软件模块的结构由结构图指定，这些图可由 DFD 变形得到。然而，系统的很多动态行为不能由函数之间的数据依赖关系推导出来。使用结构化方法很难做出动态模型，即使引入了控制流图(Control Flow Diagram，CFD)也同样如此，因为并没有在动态视图中正确地为系统建模。

由于结构化方法具有上述不足，因此将它与面向对象方法进行比较的时候，结构化方法的性价比很低。面向对象方法将软件系统建模为一组相互协作的对象。对象通过发送和接收消息的方式与其他对象交互，在这些过程中操作对象的数据。面向对象方法使得软件工程师进行软件系统一致模型的开发变得更加简单，因为在整个开发过程中，只使用了同样的一组模型。因而，在不同的阶段不需要浪费时间和精力进行模型变换和更新。

而且，使用面向对象方法开发的系统结构要比使用结构化方法更加稳定。这是因为针对面向对象系统的改变已经被限定了，仅仅发生在对象自身内部，因此修改工作要比使用结构化方法设计的系统更加容易。所以，使用面向对象方法开发的软件系统可以降低开发和维护成本。

1.4 可视化建模

人类大脑在同一时刻只能处理有限的信息。模型通过创建真实系统的抽象层次化描述，有助于降低复杂性。通过抽象创建模型是人类认识世界的一项基本技能，而对于开发大型软件系统而言，它是重要的第一步。

在创建模型时，根据那些经过仔细设计的规则将信息分为不同的层次，使得它们既不太抽象也不会有太多局限。尽管建模对于人类而言是一个很自然的过程，但是为软件系统开发一个适当的模型则可能是软件工程领域最困难的一个方面。这是因为常常会有多个解决方案；独立工作的不同观察者总会得到不同的模型。因此，开发一个系统化过程来判断在不同级别进行抽象以构造一个合理的一致模型是非常有用的。如果能够遵循一个经过验证的步骤检查列表来构造模型，就不会忽略到重要的特性或者关键的需求。

可视化建模是一种使用标准图形来表示从不同透视图描述系统的技术。例如，从静态结构的角度来看，可以使用类图来表示系统。在类图(如图 1-1 所示)中，一个对象或者类(对象的类型)可以用方形表示，对象或者类之间的关系可以用连接对象或者类的连线表示。

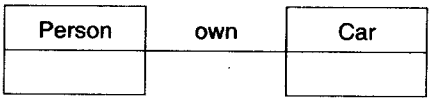

图 1-1 真实对象的类示例

可视化建模技术得到了广泛使用，特别是在大型系统的软件开发中。在软件开发过程中，可视化建模可以应用于以下场合：

- 捕获业务对象和逻辑；
- 分析和设计应用；
- 管理复杂性；
- 定义软件架构；
- 独立于实现语言为系统建模。

随着数年来面向对象方法日益成熟并逐渐流行(经过在 20 世纪 90 年代早期在面向对象联谊会上的很多争论)，UML 最终被接受为可视化建模语言，为软件开发系统建立模型。UML 涵盖了使用面向对象方法开发系统常用的所有模型。

1.5　软件开发方法

根据 Budgen(1994 年)的定义，软件开发方法主要由 3 部分组成：过程，表示系统或建模表示法以及技术、启发式法、步骤或者规程(如图 1-2 所示)。设计过程对应为从问题空间到解决空间。在这个过程中，为设计师给出了一些选项，必须做出选择或者判断。开发方法的技术部分为他们提供了一些关于正确选择的启发式法和指导，通过这种方式可以辅助设计师完成设计。典型的是，在这个过程中的每个任务或者活动结束的时候，产生系统工件(Artifact)。这些工件使用推荐的表示法从一个或者多个视点(模型)，在不同的抽象级别上进行表示，这种表示法既可以为初始问题(需求)的结构建模，也可以为将要实现的解决方案建模。

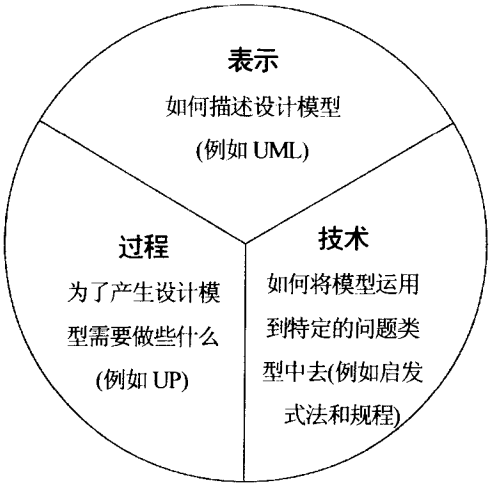

图 1-2　方法的 3 个组成部分

1. 表示法的作用

表示法(notation)是系统不同涉众之间使用的公共语言。对于软件开发而言，表示法有助于开发者进行以下活动。

- 捕获系统需求。使用的表示法应该可以被用户和开发者理解。
- 通过开发合适的分析模型分析系统。使用恰当的表示法表达模型，这样开发者就可以快速、方便地从中提取信息。
- 开发系统设计。使用恰当的表示法来开发和表达设计模型，使设计师和程序员都可以理解。系统设计师可能要操作分析模型，并在此过程中做出设计决策。
- 实现、测试和部署系统。这也需要使用合适的表示法来表示这些活动的工件，使之可以被系统设计师、程序员和系统测试人员所理解。

为了支持上面这些活动，理想的表示法应该满足以下条件：

- 有利于团队成员和客户之间的高效沟通；
- 无歧义地描述用户的需求；
- 提供丰富的语义，以捕获所有重要的战术和战略决策；
- 为人类提供一个逻辑框架，以便于对模型进行推理；
- 有利于使用工具实现自动化，至少是实现模型构建过程的自动化。

2. 过程的作用

如果没有规划好规程，那么系统开发将导致开发时间延期、费用超支，甚至是项目不能完工。因此，开发者特别是初级程序员，在开发系统时一定要遵循经过验证的过程或者规程，这样才能够在合理的预算和时间要求下完成一个可用的系统。然而，没有一个合适的过程能够适合于所有环境。因此，选中的过程只能用来指导开发者在开发系统过程中应用适当的技术。与此同时，应该允许熟练的开发者能够按照他们自己的方式组织开发步骤，这样能够鼓励创造和创新。理想情况下，过程应该具备以下特性：

- 具有良好管理的迭代和增量式生命周期，在不影响创新性的情况下提供必要的控制；
- 在整个软件开发生命周期(Software Development Life Cycle，SDLC)内为开发者嵌入方法、规程和启发式法以完成分析、设计和实现；
- 在迭代和增量式开发过程中，为复杂问题的解决提供指导；
- 根据已知的或者已经建模的需求识别那些不是非常明显的需求。

3. 技术的作用

在软件开发过程开始时，一般是从客户那里捕获系统需求，并使用合适的建模

表示法(比如 UML)来表示这些需求。因为建模表示法提供了丰富的模型，很多开发者遇到的一个共同问题是，他们不知道确定设计需要哪些模型，以及在过程中如何创建模型。

方法的技术部分的主要目的是提供一组指导意见和启发式法，辅助开发者系统地开发必需的设计模型和实现。方法的技术部分应该包含以下内容：

- 一组用于产生和验证设计与初始需求和规约的指南。
- 一组设计师可用来确保设计结构和设计模型的一致性的启发式法。如果设计由一个设计团队完成的话，这一点尤为重要，因为这些设计师需要确保他们的模型是一致和连贯的。
- 一个能够捕获设计关键特性的系统，以补充设计师的领域知识。

1.6 表示、过程、技术和工具

在本书余下内容中，采用 UML 作为表示系统，采用 UP 作为过程，并采用 VAT 作为方法的技术部分。将使用一个名为 Visual Paradigm for UML 的全功能版的 UML CASE 工具来演示模型构建过程。下面简要地介绍一下各个元素。

1. UML 概述

20 世纪 80 年代末期和 90 年代初期，不同实践者和研究人员提出了大量的面向对象分析和设计方法。UML 是大量讨论和无数论证的结果。UML 表示法现在已被对象管理组织(Object Management Group，OMG)接受，作为表示面向对象分析与设计模型的标准方法。它已经很快成为构建面向对象软件的标准。这个表示法合并了前面由面向对象技术领域内最受尊敬的三位学者(即 J. Rumbaugh、G. Booch 和 I. Jacobson，有时也被称为“三友”)提出的建模技术中最好的部分。

OMG 规范规定：

UML 是一种图形语言，用来可视化、规定、构造和记录软件密集系统的工件。UML 为编写系统蓝本提供了一个标准化方法，包括诸如业务过程和系统功能这类的抽象概念，还包括像程序语言语句、数据库设计和可复用的软件组件这类具体的东西。

通过采用像 UML 这类标准表示法，系统开发人员与领域专家(用户)之间就可以非常容易地进行高效的沟通。使用标准表示法要比其他方法(比如自然语言或者代码)更能精确地传达概念。自然语言很不精确，并且用其来表达更加复杂的概念的时候将变得非常复杂。而代码尽管非常精确，但是太注重细节，就需要太多的实现工作。像 UML 这类标准表示法，在保证一定精确度的情况下，也能提供重要的细节信息。

系统开发面临的一个最大挑战是在一个合理的成本要求(按时间和费用计算)下

构建一个满足用户需求的系统。与领域专家的沟通比较困难，因为领域专家和系统开发者使用的技术术语不同。UML 为迎合系统的不同涉众提供了不同的抽象级别。例如，用例模型可以用作用户和系统开发者之间的公共语言。用例模型通过定义用户可以观察到的结果，提供了确定系统功能的方法。

不熟悉 UML 的读者可以参考附录 B，其中更详细地描述 UML 表示法。

2. UP 概述

UP 广泛应用于软件开发过程中。在 UP 中，经过多次迭代增量地构建整个系统，在这个过程中，设计师可以进行需求捕获、分析、设计、实现和测试等任务。在整个过程中都从系统用户那里获取反馈。在早期迭代中，设计师常常更加关注需求捕获和分析，而在后期迭代中，关注实现和测试。实际上，迭代可以划分为 4 个阶段：开始阶段、细化阶段、构造阶段和转换阶段。每个阶段都有不同的关注点。

在同一主题区域中的 UP 工作活动以工作流(Workflow)进行分类。例如，设计工作流(Design Workflow)包括所有与系统设计相关的活动。如图 1-3 所示为一些示例工作流和它们在 UP 不同阶段中的相对工作量。因为每个阶段都有特定的重点或者关注点，所以随着系统开发工作的开展，工作流的相对工作量不断地发生改变(纵轴)。

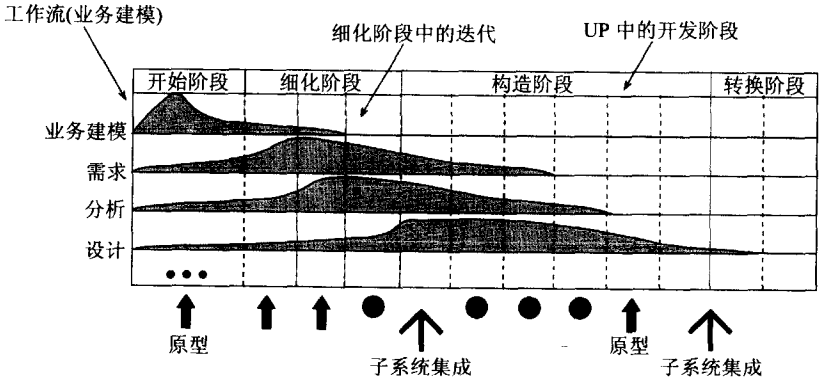

图 1-3 示例工作流和随着时间推进它们的相对工作

3. VAT 概述

VAT 的关键概念是基于不同透视图的模型必须包含一些公共的元素(相连元素)。因此，可以简单地从一个模型开始，并通过识别和使用相连元素生成(局部的)模型。通过向一部分完成的模型中填充(详细的)缺失信息，并识别出更多相连元素，可以创建其他透视图的模型。在这个迭代和增量式过程中，可以创建所有需要的模型。当开发完描述系统不同透视图的所有模型之后，通过将这些视图(模型)适当地调整，就能够形成一个完整的一致的系统图。因此，不但能够保证模型是一致的，还能系统地识别出开发这些模型的顺序。换句话说，VAT 可以辅助设计人员在开发系统时定制他们的方法。