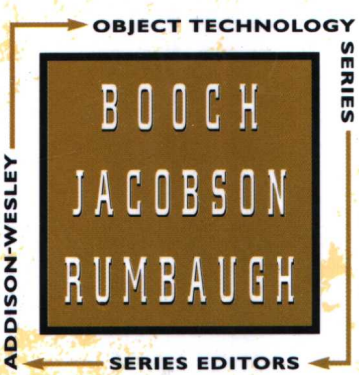
软件工程技术丛书

统一软件开发过程

The Unified Software Development Process



(美) Ivar Jacobson
Grady Booch 著
James Rumbaugh

周伯生 冯学民 樊东平 译

机械工业出版社
China Machine Press

Addison-Wesley

软件工程技术丛书

统一软件开发过程

Ivar Jacobson

（美） Grady Booch 著

James Rumbaugh

周伯生 冯学民 樊东平 译

机械工业出版社
China Machine Press

本书是由UML的三位创始人Ivar Jacobson，Grady Booch，James Rumbaugh亲自撰写的。全书给出了一种以UML作为建模语言进行软件开发的过程指导。书中的内容不是UML固有的组成部分，因为UML只是一种建模语言，并不包括过程指导。实际上，UML独立于过程的特点可以使之用于不同的软件开发过程。但是本书介绍的软件开发过程是三位作者在开发UML时一直在头脑中思考的内容，因此很切合UML的特点。本书对于如何运用UML的概念进行软件开发提供了详细指导，适合参与软件开发的各类人员使用，尤其适合软件项目开发组成员阅读。

本书版权登记号：图字：01-1999-3680

图书在版编目(CIP)数据

统一软件开发过程/（美）雅各布森（Jacobson, I.），（美）布谢（Booch, G.），（美）朗博（Rumbaugh, J.）著；周伯生等译. - 北京：机械工业出版社，2002.1

(软件工程技术丛书)

书名原文：The Unified Software Development Process

ISBN 7-111-07572-2

I. 统… II. ①雅… ②布… ③朗… ④周… III. 软件开发 IV.TP311-52

中国版本图书馆CIP 数据核字（2001）第047726号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：姚 蕾

北京市密云县印刷厂印刷·新华书店北京发行所发行

2002年1月第1版·2002年6月第4次印刷

787mm × 1092mm 1/16 · 24.25印张

印数：15 001-19 000册

定价：45.00元

译者序

随着计算机硬件性能的不断提高和价格的不断下降，其应用领域也在不断扩大。人们在越来越多的领域希望把更多、更难的问题交给计算机去解决。这使得计算机软件的规模和复杂性与日俱增，因而软件技术不断地受到新的挑战。20世纪60年代软件危机的出现就是因为系统的复杂性超出了人们在当时的技术条件下所能驾御的程度。此后在软件领域，从学术界到工业界，人们一直在为寻求更先进的软件方法与技术而奋斗。每当出现一种先进的方法与技术，都会使软件危机得到一定程度的缓和。然而这种进步又立刻促使人们把更多、更复杂的问题交给计算机去解决。于是又需要更先进的方法与技术。

开发一个具有一定规模和复杂性的软件系统和编写一个简单的程序大不一样。其间的差别，借用G. Booch的比喻，如同建造一座大厦和搭一个狗窝的差别。大型的、复杂的软件系统的开发是一项工程，必须按工程学的方法组织软件的生产与管理，必须经过分析、设计、实现、测试、维护等一系列的软件生命周期阶段。这是人们从软件危机中获得的最重要的教益。这一认识促使了软件工程学的诞生。编程仍然是重要的，但是更具有决定意义的是系统建模。只有在分析和设计阶段建立了良好的系统模型，才有可能保证工程的正确实施。正是由于这一原因，许多在编程领域首先出现的新方法和新技术，总是很快地拓展到软件生命周期的分析与设计阶段。

面向对象方法正是经历了这样的发展过程，它首先在编程领域兴起，作为一种崭新的程序设计范型引起世人瞩目。继Smalltalk-80之后，20世纪80年代又有一大批面向对象的编程语言问世，标志着面向对象方法走向成熟和实用。此时，面向对象方法开始向系统设计阶段延伸，出现了如Booch86、GOOD（通用面向对象的开发）、HOOD（层次式面向对象的设计）、OOSD（面向对象的结构设计）等一批OOD（“面向对象的设计”或“面向对象的开发”的缩写）方法。但是这些早期的OOD方法不是以面向对象的分析（OOA）为基础的，而主要是基于结构化分析。到1989年之后，面向对象方法的研究重点开始转向软件生命周期的分析阶段，并将OOA和OOD密切地联系在一起，出现了一大批面向对象的分析与设计（OOA&D）方法，如Booch方法、Coad/Yourdon方法、Firesmith方法、Jacobson的OOSE、Martin/Odell方法、Rumbaugh等人的OMT、Shlaer/Mellor方法等等。截至1994年，公开发表并具有一定影响的OOA&D方法已达50余种。这种繁荣的局面表明面向对象方法已经深入到分析与设计领域，并随着面向对象的测试、集成与演化技术的出现而发展为一套贯穿整个软件生命周期的方法体系。目前，大多数较先进的软件开发组织已经从分析、设计到编程、测试阶段全面地采用面向对象方法，使面向对象无可置疑地成为当前软件领域的主流技术。

各种面向对象的分析与设计方法都为面向对象理论与技术的发展做出了贡献。这些方法各有自己的优点和缺点，同时在各自不同范围内拥有自己的用户群。各种方法的主导思想

以及所采用的主要概念与原则大体上是一致的，但是也存在不少差异。这些差异所带来的问题是，不利于面向对象方法向一致的方向发展，也会给用户的选择带来一些困惑。为此，Rational公司的G. Booch和J. Rumbaugh决定将他们各自的方法结合起来成为一种方法。1995年10月发布了第1个版本，称作“统一方法”（Unified Method 0.8）。此时OOSE的作者I. Jacobson也加入了Rational公司，于是也加入了统一行动。1996年6月发布了第2个版本UML0.9。鉴于统一行动的产物只是一种建模语言，而不是一种建模方法（因为不包含过程指导），所以自0.9版本起，改称“统一建模语言”（Unified Modeling Language）。在此过程中，由Rational公司发起成立了UML伙伴组织。开始时有12家公司加入，共同推出了UML1.0版本，并于1997年1月提交到对象管理组织（OMG）申请作为一种标准建模语言。此后，又把几家分头向OMG提交建模语言提案的公司扩大到UML伙伴组织中，并为反映他们的意见对UML进一步做了修改，产生了UML1.1版本。该版本于1997年11月4日被OMG采纳。此后UML还在继续改进，目前最新的版本是UML1.3。

关于UML的历史、发起的动机、目标、权衡的问题等，这里不想做更多的介绍，因为读者很快会从这套丛书中的《UML用户指南》的前言中看到更详细的叙述。这里想着重指出的是以下三点：第一点是UML的三位发起人G. Booch、J. Rumbaugh和I. Jacobson是从事面向对象研究的著名专家，他们各自的方法和著作在该领域均具有很大的影响。第二点是众多的大公司加入了UML阵营，对UML的制定和推广提供了强有力的支持。第三点是UML经过数年的努力终于被OMG采纳，成为该组织承认的一种标准建模语言。总之，UML是吸收多种方法的成果、凝结了许多组织和个人智慧的产物。

UML是一种用于对软件密集型系统进行可视化、详述、构造和文档化的建模语言，主要适用于分析与设计阶段的系统建模。UML最主要的特点是表达能力丰富。因为它从各种OOA&D方法中吸取了大量的概念，并在“UML语义”、“UML表示法指南”、“对象约束语言规格说明”等UML文献中对这些概念的语义、图形表示法和使用规则做了完整而详细的定义。可以说，UML对系统模型的表达能力超出了以往任何一种OOA&D方法。当然，随之而来的问题是，它的复杂性也超出了以往任何一种方法。

UML的问世受到计算机软件界的广泛重视，因为它代表了一种积极的方向——多种方法相互借鉴、相互融合、趋于一致、走向标准化。建模语言的标准化将为软件开发商及其用户带来诸多便利。因此，在美国等国家已有大量的软件开发组织开始用UML进行系统建模。学习和使用UML已经成为一种潮流。我国软件界对UML也相当关注。许多研究人员和技术人员已在数年前开始学习和研究UML。更有许多人想学习UML，但苦于找不到合适的书籍。由于UML的复杂性，仅通过UML的标准文献来学习和使用它确实不是一件轻松的事。以往国内外也曾发表过一些介绍或评述UML的著作或论文，但是与UML的丰富内容相比，这些介绍远不能满足读者的要求。

值得高兴的是，UML的三位主要设计者G. Booch、J. Rumbaugh和I. Jacobson现在已亲自撰写了这套详细阐述UML的著作，由Addison Wesley于1999年出版。这套著作对UML进行了详细、深入而准确的介绍和论述，而且语言生动、深入浅出、实例丰富、图文并茂。这是一套教会读者掌握和使用UML的教材和指导手册，而不是枯燥的标准文献。对于想学

习和使用UML的广大读者，这是一套难得的好书。为了使中国的读者能够更好地从中受益，我们在机械工业出版社华章公司的恳切建议下，分头翻译了这三本书，即《UML用户指南》、《UML参考手册》和《统一软件开发过程》。

三本原著都是由这三位作者合著，既各自独立，又有很强的内在联系。其中《UML用户指南》介绍了UML的基础知识，包括UML的术语、规则和语言特点，以及如何运用该语言去解决常见的建模问题，初学者学习UML最好从阅读该书开始。《UML参考手册》对UML的组成和概念做了详细的介绍，包括这些概念的语义、语法、表示法和用途，是一本适合软件专业人员使用的方便而全面的参考读物。《统一软件开发过程》给出了一种以UML作为建模语言进行软件开发的过程指导。其内容不是UML固有的组成部分，因为被OMG采纳的UML只是一种建模语言，并不包含过程指导。实际上UML是独立于过程的，可以用于不同的软件过程。但是该书介绍的软件开发过程是三位作者在开发UML时一直在头脑中思考的，因此很切合UML的特点。该书对于如何运用UML的概念进行软件开发提供了详细指导，适合软件专业人员使用。

鉴于UML本身以及这套著作的重要意义，译者在翻译这些著作时采取了特别慎重和严谨的态度。力求准确和通顺。在翻译过程中，一个重要问题是要使这套书中的专业术语的中文译法取得一致。这三本书的译者以往曾分别开展过一些与UML有关的研究和写作，对有些术语的译法互有差异。本次翻译工作中，所有译者在机械工业出版社华章公司的组织下进行了多次讨论、研究和交流，首先对所有专业术语的译法统一意见，达成共识。其中某些术语的译法颇难定夺：既要确切反映英文本意，又要符合中文习惯，还要避免与国内已习惯于与其他英文词对应的中文相混淆。经过反复切磋，大部分问题都得到满意的解决。对个别有争议的问题，在充分讨论的基础上采取放弃己见，服从大局的态度。如此形成了一个译法一致的词汇表。此后在翻译过程中还经常以各种交流方式进行磋商和沟通。最终使这套丛书能以一致的面貌呈献给读者。我们也希望这些工作能为UML术语今后在中文翻译中的统一贡献一份力量。

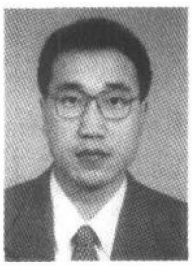
在科技著作的翻译中，保证准确和通顺的关键因素不仅仅是外文水平，还取决于译者真正了解所涉及的技术内容。这套著作的内容远远超出了UML的标准文献，因为除了介绍UML的语法、语义、使用规则之外，其中还包含许多学术思想、技术策略和实践经验。在翻译中遇到的许多疑难问题，是通过进一步研究UML以及有关的学术和技术问题而得到解决的，从而避免了许多讹误。因此，这套著作的翻译不仅是文字方面的工作，还包含译者在技术上的研究。我们希望这些研究最终通过较准确的翻译文字使读者受益。同时诚恳地希望广大读者对可能存在的疏漏和错误之处给予批评和指正。

译　者

2000年11月于北京

译者简介

周伯生，男，1935年生。北京航空航天大学计算机科学与工程系教授，计算机软件专业博士生导师，专攻软件工程与过程工程。1988年创办北航软件工程研究所；1984年2月～1986年7月，任国家科委与美国ISSI公司合作项目总体组组长；1986年1月～1995年12月，任国家七五和八五科技攻关软件工程及环境总体组成员；1991年10月以来，任北航与美国FunSoft公司“过程工程环境”合作项目技术负责人。



冯学民，男，1968年生。北京航空航天大学软件工程研究所讲师，主要从事软件工程方面的研究及应用项目的开发。

樊东平，男，1972年生，博士。北京航空航天大学计算机科学与工程系讲师，主要从事软件工程及其支持环境方面的研究。

前　言

一些人持有这样的观点，专业公司应该围绕受过严格技能训练的人员组建。这些人员了解他们所要做的工作，并且能正确地去做！他们在工作时，几乎不需要从组织获取相关的方针或者规程上的指导。

然而，这种想法在绝大多数情况下都是错误的，对于软件开发就更是如此。的确，软件开发人员都是经过严格训练的专业人员，但该专业毕竟还很年轻。因此，开发人员需要有组织的指导，在本书中，我们称之为“软件开发过程”。此外，因为在编写这本书的过程中，我们将以前的零散方法集中在一起，所以我们觉得更正确的叫法应该是“统一过程”。这种统一不仅仅集中了三位作者的工作，还有机地结合了众多的致力于UML研究与应用的企业和个人（也包括Rational软件公司中众多的相当关键的人员）的成果。它从数以百计的用户组织（在客户站点上使用过过程的较早版本）的实践经验中吸取了大量重要的信息。

举例来说，一个交响乐队的指挥在一场表演中要做的事情很少，仅仅是告诉演奏者何时开始演奏并帮助他们协调一致。作为指挥的他或她所需要做的工作如此之少，是因为他在排练和演出的准备过程中对乐队进行过指导，而且乐队里的每个成员不但能够非常熟练地演奏自己的乐器，而且还能够独立于其他成员进行演奏。在这里，我们更深层的用意是想说明，每个音乐家都遵循着由作曲家早已安排好的一个“过程”。正是乐谱提供了大量的对演奏加以指导的“方针和规程”。相反，软件开发人员不是独立行动的，而是与其他开发人员和用户相互协作的。他们没有“乐谱”可以遵循，直到有了一个过程（*process*）。

尤其在其软件系统负有重大使命（比如金融、空中交通管制、国防和电信系统等）的公司或组织里，对过程的需求就更加迫切。在这里，我们的意思是，业务的成功管理或公共任务的执行依赖于支持它的软件。这类软件系统正变得更加复杂，要求投入到市场的时间要尽量短，对它们的开发也相应地变得更加困难。基于以上原因，软件工业需要一个指导开发人员的过程，就像交响乐团需要一个作曲家的乐谱来指导演奏一样。

什么是软件开发过程？

一个过程定义了为达到某个确定的目标，需要什么人在什么时间以何种方式做何种工作。对于软件工程而言，其目标是构造一个新的软件产品或者完善一个旧的软件产品。一个有效的过程为有效地开发高质量的软件提供准则，它获取并提出当前技术条件下可行的最佳实践方案。因此，它可降低风险并增强预见性。总的效果是要发扬一种共同的构想与文化。

我们需要一个可用于指导顾客、用户、开发人员和项目经理等参与者的过程。我们需要一个能将软件产业有史以来第一次有机地结合在一起的最好的过程——任何旧的开发过程都

无法做到这一点。最后，我们需要一个广泛适用的过程，以便所有的项目相关人员都能理解自己在所进行的开发工作中的角色。

一个软件开发过程还要能随时间的推移不断地进化。在进化过程中，它能及时调整自己以适应技术、工具、人员以及组织模式的变化。

- **技术**：过程必须建立在当时可以使用的实用技术（如编程语言、操作系统、计算机系统、网络能力、开发环境等）基础之上。例如，20年前可视化建模技术还不是主流技术，因为它的代价太高。因此，那时的过程建造者不得不采取手工绘图的方式，这大大限制了过程发起者把模型引入过程的程度。
- **工具**：过程和工具必须同时发展。工具是过程的一部分。换一个角度来讲，一个广泛应用的过程能对支持它的工具的开发起到促进作用。
- **人员**：过程建造者必须将运作过程所需的技能作一定的限制，使得这些技能是现有开发人员已经具备的或者通过培训能迅速掌握的。在许多领域中，现在都可以嵌入曾经需要广泛技能的技术，比如在基于计算机的工具中要检查模型图的一致性。
- **组织模式**：软件开发人员并不是像交响乐演奏家那样各自独立的专家。他们远非一百多年前Frederick W. Taylor在其《科学的管理》中所论及的自动化工人。过程建立者要让过程适合当前的实际情况——例如，虚拟的组织形态，通过高速线路远距离工作，小公司中的部分业主、支取薪水的雇员、合同工和外协转包商等的结合，以及软件开发人员的持续短缺。

过程工程师需要权衡以上四种因素。此外，这种权衡不仅现在必须存在，而且还要保持到将来。过程建造者必须将过程设计成可以进化的，就像软件工程师设法开发不仅在目前可用而且还能在未来岁月里不断进化的软件。一个过程要经历数年的时间才能达到稳定和成熟的水平，能够经得起商业产品开发的严格考验，并将其使用风险保持在合理的水平。不仅没有通过实践充分检验过的过程有风险，而且开发新产品本身也要冒巨大的风险。基于对以上因素的考虑，过程可以保持相对稳定。在没有充分权衡技术、工具、人员和组织的情况下使用过程将会有相当大的风险。

本书的目的

本书主要阐述当我们在开发统一建模语言（UML）时经常会想到的软件过程。UML给了我们一种标准的方法，对软件密集型系统中的制品进行可视化描述、详细说明、构建、文档化以及相互交流。当然，我们都意识到这样一种语言必须应用到软件开发的全过程。UML不是目标，而是手段。最终的目标是健壮的、有弹性的和可扩充的软件应用系统。要达到这样的目标，既需要过程，又需要语言，而本书编写的目的就是要阐明过程。虽然我们提供了有关UML的简要附录，但这个附录并不是全面的或详尽的。对于详细的UML指南，请查阅《UML用户指南》一书[11]。对于全面的UML参考，请查阅《UML参考手册》一书[12]。

本书的读者

参与软件开发的任何人员都可使用统一软件开发过程。其中主要是为处理生命周期活动（需求、分析、设计、实现和测试——即生成UML模型的工作）的开发组成员编写的。因此，举例来说，本书与分析人员和最终用户（他们详细描述系统所需的结构和行为）、应用开发人员（他们设计满足这些需求的系统）、编程人员（他们将设计转换成可执行代码）、测试人员（他们验证与确认系统的结构和行为）、构件开发人员（他们创建与收录构件）以及项目和产品经理等都有关系。

本书读者最好是基本掌握了面向对象概念。如果具有软件开发以及面向对象程序设计语言方面的经验就更好，但这并不是必需的。

本书的学习方法

本书用大部分的笔墨阐述在UML中认为是重点的活动——需求、分析和设计。正是在这些重点活动中，过程开发了复杂软件系统的**构架**（*architecture*）。然而，我们将开发过程作为一个整体来对待，虽然缺乏细节。最后要运行的仍然是一个可执行程序。要达到这一目标，需要全体开发人员的共同努力和项目相关人员的大力支持。正如你将要看到的，过程取决于大量不同的活动，会产生大量的制品并必须加以追踪。对于所有这些活动，必须加以管理。

任何一本书都无法完整地涵盖一个综合而完整的生命周期过程。这样一本书必须包括设计指南、制品模板、质量指示、项目管理、配置管理、度量等内容，在此无法一一列举。但随着在线访问的产生，前面无法枚举的东西现在已可利用，可以当成新的开发指令加以更新。为此我们向你推荐“Rational统一过程”，它是一个可从Web下载的软件产品，可用来指导软件开发组进行更有效的软件开发实践（要了解更多的信息，请参见http://www.rational.com）。Rational统一过程涵盖了整个软件生命周期，拓展了本书所阐述的统一过程的范围，并提供了本书所没有包含的或者是一笔带过的其他工作流，例如业务建模、项目管理和配置管理等。

统一过程的历史

统一过程是稳定的，因为它是经过30年的发展和实际运用后推出的最终产品。像一个产品那样，它的发展历程（如图1）从“对象工厂过程”开始（1987年发布），经过“Rational对象工厂过程”（1997年发布），到“Rational统一过程”（1998年发布）。它的发展受到了诸多因素的影响，在这里并不一一列举这些因素（我们实际上也并不了解所有这些因素），而把这个问题留给软件考古学家去研究，我们将描述Ericsson方法、Rational方法以及其他一些因素对统一过程的影响。

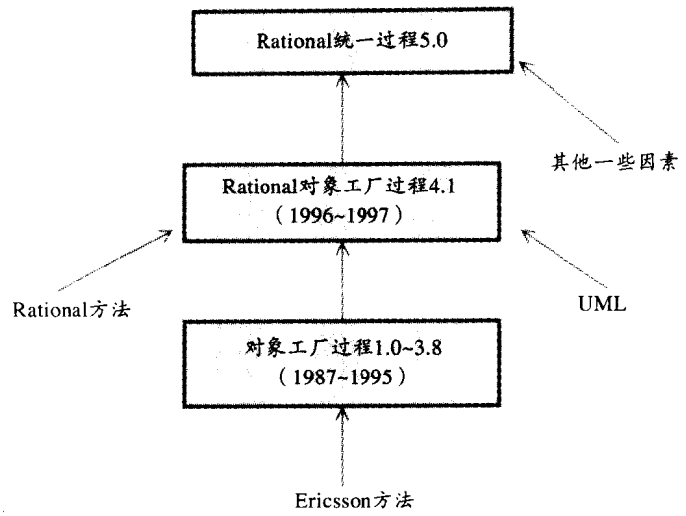


图1 统一过程的发展历程（在灰色的矩形框中指出了产品的版本号）

Ericsson方法

统一过程有着很深的渊源。用Peter F. Drucker的话来说，过程是一个“基于知识的创新”。他忠告我们：“新知识的出现与将其精华转化为可用的技术之间存在着相当大的一段距离，因而，在这一新的技术以产品、过程或服务的形式在市场上出现之前，还会经历另一段相当长的时期”。[1]

产生这段相当长的前导期的原因之一是，基于知识的创新是将很多不同类型的知识融汇贯通，而这是要花时间的。另一个原因是人们需要时间来吸收那些新的思想，将其传播给他人，才能使新思想真正有效。

要阐明统一过程的发展历程，首先让我们回到1967年去具体地概括一下Ericsson（爱立信公司）所取得的成果[14]、[15]、[16]。Ericsson将整个系统模型化成相互联系的模块（在UML中，称之为“子系统”，并当做“构件”实现）的集合。他们用底层的模块装配成较高层的子系统，使整个系统更易于管理。通过遍历以前曾详细描述过的业务实例（现在称为“用况”）来发现这些模块。对于每一个用况，识别出其实现所需的相关模块。了解了每个模块的职能后，他们就编制每一个模块的规格说明，设计出一批带有接口的静态模块图，并将其组合成子系统。这些模块图直接对应于UML的类图或包图的简化版本——称之为“简化”是因为它仅仅表示了用于通信的关联。

设计活动的第一个工作产品是软件的**构架描述**（*architecture description*）。它基于对最关键的需求的理解，简要描述每个模块以及如何将它们组合成子系统。一组模块图描述了模块及其相互联系。在其联系上，信号（即一种消息）用于模块之间的通信。所有消息都在消息库中逐个地加以描述。软件构架描述和消息库是指导开发工作的关键文档，同时也可用于向

客户介绍整个系统。在那时（1968年），客户还不大习惯软件开发人员以类似于工程蓝图的方式向客户介绍软件产品。

对于每个用况，工程师既要准备顺序图，还要准备协作图（现在，在UML中这方面得到了进一步的发展）。这些图描述了模块是如何通过动态通信来实现该用况的。工程师以状态图（只包括状态和转换）和状态转换图（UML活动图的简化版本）的形式编制规格说明。这种使用具有良好定义的接口的模块来设计系统的方法是成功的关键。例如，现在通过另一个提供相同接口的模块来替换其中的某个模块，可以为一个新客户创建该系统的一个新配置。

现在，模块不只是子程序或者源代码构件；它们被编译成可执行的模块，逐个地被安装到目标机器中，并被确定能与其他的可执行模块一起运行。此外，对一个运行中的（如电话呼叫）系统，必须能够在系统运行时就能安装新模块或者替换旧模块。这样的系统不能因为要做更改而停止，就好比不能给一辆正以60英里时速行驶的轿车替换轮胎一样。

本质上讲，这种方法就是我们今天所说的**基于构件的开发**（*Component-Based Development*, CBD）。Ivar Jacobson是该开发方法的创始人。经过“对象前时代”的多年努力，他指引着该方法进化成为一种软件开发过程。

规格说明与描述语言（SDL）

这一时期具有重大意义的进展是：1976年，CCITT（国际电话电报咨询委员会，电信领域的国际标准化组织）公布了用以描述电信系统中功能行为的**规格说明与描述语言**（*Specification and Description Language*, SDL）。该标准受Ericsson方法的重大影响，用一系列相互关联的模块来刻画一个系统，各模块之间仅仅通过消息（标准中称之为“信号”）来相互通信。每一个模块都拥有一系列的“过程”，它们是由SDL术语表示的主动类。一个过程拥有实例，非常类似于面向对象术语中的类。过程实例通过消息产生交互。SDL所推荐使用的图可以看成是UML中的类图、活动图、协作图和顺序图的特化。

因此，SDL是一种特化的对象建模标准。SDL被定期地更新，现在仍有10 000多个开发人员在使用，并且仍得到一些软件工具供应商的支持。SDL是在20多年前开发的，远超前于所处的时代。但是，SDL是在对象建模还没有成熟的时候开发的，看来它将会被UML所替代，而后者在1997年已成为一种标准。

对象工厂

1987年，Ivar Jacobson 离开了Ericsson公司，并在斯德哥尔摩建立了“Objectory AB”。在后来的八年时间里，他和助手们一起开发了一个称为对象工厂（Objectory，它是“Object Factory”的缩写）的过程产品，并将该产品推广到电信以外的许多行业和瑞典以外的许多国家。

尽管Ivar Jacobson在Ericsson的工作中已经提出了用况（*use case*）的概念，但直到现在

它才有了一个名称（是在1987年的OOPSLA讨论会上提出的），并随着图示化技术的发展，这种思想才扩展到多个不同的应用领域。正是用况使得开发过程更加清晰；正是构架引导了开发人员并让项目相关人员也走到了开发的前台。

各种相继的工作流表征为一系列的模型：需求—用况、分析、设计、实现和测试。一个模型是对一个系统的一种刻画。在该系列中模型之间的关系对开发人员而言非常重要，可看作是从模型序列的一端到另一端某个功能特征的演进路线。事实上可跟踪性已成为用况驱动的开发过程的先决条件。开发者可以通过模型顺序跟踪到对应于某个用况的源代码，或者当出现问题时能够进行逆向跟踪。

对象工厂过程在发展中产生了一系列的版本，从1988年的Objectory 1.0版本到1995年的第一个在线版本（Objectory 3.8版本）（参阅[2]中的Objectory介绍）。

需要着重指出的是，对象工厂产品本身也被看成是一个系统。这种将过程描述成一个系统产品的方法为从早期的（对象工厂）版本开发新的（对象工厂）版本提供了良好的途径。这种设计对象工厂的方法可以很轻松地加以裁剪以适应不同开发组织的需要。事实上，对象工厂软件开发过程的一个独特的特征是：其本身就是工程化的。

对象工厂的开发实践也为如何去巧妙地设计业务通常运作的过程提供了经验。同样的原理是适用的，并在1995年的一本书中进行了讨论[3]。

Rational方法

Rational软件公司在1995年秋季引进了Objectory AB，并在已有的软件开发过程基础之上统一了基本规则，从而获得了新的突破。Rational公司开展了大量的软件开发实践，其中很多是对对象工厂已有内容的补充。

例如，James E. Archer Jr. 和Michael T. Devlin在1986年的回忆[4]中曾提到："Rational于1981年开始研制一个能提高大型软件开发效率的交互式环境。"他们还进一步提到，在这项工程中，面向对象设计、抽象、信息隐藏、可复用性以及原型化等技术都显得很重要。

自1981年以来，大量的书籍、文章和内部文献资料都详细地描述了Rational的开发过程。然而，对于过程的建立最具贡献意义的两个因素也许是对构架和迭代式开发的强调。例如，1990年Devlin写了一篇关于构架驱动的迭代式开发过程的前瞻性文章；Philippe Kruchten（在Rational公司中负责构架实践方面的工作）也发表过一些有关迭代和构架的文章。

我们引用其中一点，一篇有关构架的文章一般都表达了四个方面的视图：逻辑视图、过程视图、物理视图和开发视图，另外再加上一种用用况或者场景阐述前面四种视图的视图[6]。根据Kruchten在几个大型项目中的实践经验，拥有一组视图比试图把所有东西都塞进一种视图中更有价值。多视图使得项目相关人员和开发人员都能通过适当的角度去发掘他们所想要的东西。

一些人曾认为迭代式开发有些混乱无序和漫无边际。在迭代中引进四阶段法（初始、细化、构造和移交）可以设计出更好的结构和更好的迭代控制过程。阶段法强制迭代按照一

定的顺序进行。阶段的详细计划编制和阶段内迭代的具体排序是Walker Royce和Rich Reitman的小组共同努力的结果，此外也要归功于Grady Booch和Philippe Kruchten的长期参与。

Booch是Rational公司创立和发展的见证人。1996年，他在一本书中声明了两条关于构架和迭代的“首要原则”：

- “构架驱动方式的开发方法通常是开发最复杂的软件项目的最好方法。”
- “一个成功的面向对象的项目必须采用一种迭代增量式的过程。”[7]

Rational对象工厂过程：1995~1997

在合并前，对象工厂3.8版本已经说明了作为产品的软件开发过程是如何进行开发和建模的。它设计了软件开发过程的原始构架，建立了一套记录过程结果的模型。在用况建模、分析和设计等方面取得了很好的进展，但在不同于用况的需求管理、实现和测试等其他方面却没有取得太大进展。此外，在项目管理、配置管理、实施以及开发环境的准备（工具和过程选取）等方面它也很少涉足。

现在，我们将Rational的经验和实践加入，从而形成了Rational对象工厂过程4.1版本。特别是，新版本中加入了阶段和受控的迭代方法。构架以构架描述的形式清晰地展现，并成为软件开发组织的“圣经”。此外，建立了构架的精确定义，把它看成是系统组织的重要部分，并把它描述为模型的构架视图，从而使迭代式开发方法从相对一般的概念发展成为把构架放在首位的风险驱动的开发方法。

当前，UML还在继续发展，并用作Rational对象工厂过程（Rational Objectory Process, ROP）的建模语言。本书的三位作者是UML的创始人。以Philippe Kruchten为首的过程开发组，通过加强项目管理（例如，基于Royce的研究成果[8]），弥补了ROP的一些不足。

统一建模语言

在20世纪90年代初期出现了众多的面向对象方法，因此需要一种统一的和一致的可视化语言来表达这些方法，这种需求在相当长的一段时间里是非常明显的。

举例来说，在这段时间里，Grady Booch提出了Booch方法[9]，James Rumbaugh是通用电气公司对象建模技术（Object Modeling Technique, OMT）研发中心的首席开发人员[10]，1994年10月他加入到Rational公司时，两人与Rational公司的客户一道，开始统一他们的方法。1995年10月他们发布了UM（统一方法）的0.8版本。差不多在同一时间，Ivar Jacobson也加入到Rational公司。

这三个人一起工作，发布了UML（统一建模语言）0.9版本。该版本的发布带动了其他的方法学研究人员和众多的公司，包括IBM、HP和Microsoft等，他们为标准的进化做出了应有的贡献。1997年11月，在经历了标准化过程后，UML 1.1版本由对象管理组织（Object

Management Group, OMG）公布为标准。详细信息请参阅《UML用户手册》[11]和《UML参考手册》[12]。

Rational对象工厂过程的所有模型都使用UML。

Rational统一过程（RUP）

在这段时间里，Rational公司收购或者兼并了其他一些软件工具公司。每个公司都带来了在过程方面的一些专门技术，从而进一步发展了Rational对象工厂过程：

- Requisite公司为Rational带来了需求管理方面的经验。
- SQA 公司为配合其测试工具曾经开发了一个测试过程，给Rational在这方面增加了进一步的经验。
- Pure-Atria公司为Rational增加了配置管理方面的经验。
- Performance Awareness公司为Rational增加了性能测试和负载测试方面的经验。
- Vigortech公司为Rational带来了数据工程方面的专门技术。

该过程还基于[3]扩展了针对业务建模的新工作流，用于从业务过程提取为该业务服务的软件的需求。该过程也涉及到由用况（基于Objectory AB所做的工作）所驱动的用户界面设计。

到1998年中期，Rational对象工厂过程已经完全成熟，能够支持整个软件开发生命周期。这样做的结果是，它统一了广泛的来自不同来源（既包括上面所提到的三位作者提出的方法和思想，还包括许多Rational 和UML得以导出和发展的其他来源）的贡献。1998年6月，Rational公司发布了该产品的新版本——Rational统一过程（RUP）5.0版本[13]。这个Rational公司专有过程的很多内容现在以本书的形式第一次向公众公开。

名称的变化反映了统一是从多个角度进行的：开发方法的统一，使用统一建模语言以及很多方法论研究人员（不仅包括Rational公司中的研究人员，还包括几百个多年以来一直应用该过程的客户站点中的研究人员）的研究成果的统一。

致谢

本书的编写、出版和发行是许多人员辛勤劳动的结晶，在此列举他们的名字并向他们致以诚挚的谢意。

为本书的编写做出贡献的人员

Birgitte Lønvig 准备了Interbank系统示例，并用所有模型来说明，这是本书的主要示例。

Partrik Jonsson从Rational对象工厂文献中选取材料，并按所建议的章节顺序对它们加以整理。他还帮助准备了示例。在此过程中，他还就如何更好地介绍统一过程发表了自己的一些意见。

Ware Myers从开始策划时就参与了本书的编写，是他将主要作者提供的最初手稿改写得更通俗易懂。

在审阅人员中，我们尤其要感谢Kurt Bittner、Cris Kobryn、和Earl Ecklund, Jr。此外，我们要特别感谢Walker Royce、Philippe Kruchten、Dean Leffingwell、Martin Griss、Maria Ericsson和Bruce Katz。审阅人员还包括Pete McBreen、Glenn Jones、Johan Galle、N. Venu Gopal、David Rine、Mary Loomis、Marie Lenzi、Janet Gardner以及一些没有留下姓名的审阅者，在此也向他们表示感谢。

Rational公司的Terry Quatrani对本书第1章到第5章的英文进行了修改完善。Karen Tongish对全书进行了编辑。在此，谨向两位表示感谢。

我们还要特别感谢Stefan Bylund，他审阅了全书的草稿并详细地提出了改进意见，其中很多都被我们采纳。他为本书质量的提高做出了巨大贡献。

多年来帮助我们的人员

我们还要感谢多年来一直帮助我们“建立正确的过程”，并通过不同方式支持我们工作的人们。特别要感谢下面一些人：Stefan Ahlquist、Ali Ali、Gunilla Andersson、Kjell S. Andersson、Sten-Erik Bergner、Dave Bernstein、Kurt Bitter、Per Bjork、Hans Brandtberg、Mark Broms、Stefan Bylund、Ann Carlbrand、Ingemar Carlsson、Margaret Chan、Magnus Christerson、Geoff Clemm、Catherine Connor、Hakan Dahl、Stephane Desjardins、Mike Devlin、Hakan Dyrhage、Susanne Dyrhage、Staffan Ehnebom、Christian Ehrenborg、Maria Ericsson、Gunnar M. Eriksson、Iain Gavin、Carlo Goti、Sam Guckenheimer、Bjorn Gullbrand、Sunny Gupta、Marten Gustafsson、Bjorn Gustafsson、Lars Hallmarken、David Hanslip、Per Hedfors、Barbara Hedlund、Jorgen Hellberg、Joachim Herzog、Kelli Houston、Agneta Jacobson、Sten Jacobson、Paer Jansson、Hakan Jansson、Christer Johansson、Ingemar Johnsson、Patrik Jonsson、Dan Jonsson、Bruce Katz、Kurt Katzeff、Kevin Kelly、Anthony Kesterton、Per Kilgren、Rudi Koster、Per Kroll、Ron Krubeck、Mikael Larsson、Bud Lawson、Dean Leffingwell、Rolf Leidhammar、Hakan Lidstrom、Lars Lindroos、Fredrik Lindstrom、Chris Littlejohns、Andrew Lyons、Jas Madhur、Bruce Malasky、Chris McClenaghan、Christian Meck、 Sue Mickel、Jorma Mobrin、Christer Nilsson、Rune Nilsson、Anders Nordin、Jan-Erik Nordin、 Roger Oberg、Benny Odenteg、Erik Ornulf、Gunnar Overgaard、Karin Palmkvist、Fabio Peruzzi、 Janne Pettersson、Gary Pollice、Tonya Prince、Leslee Probasco、Terry Quatrani、Anders Rockstrom、Walker Royce、Goran Schefte、Jeff Schuster、John Smith、John Smith、Kjell Sorme、 Ian Spence、Birgitta Spiridon、Fredrik Stromberg、Goran Sundelof、Per Sundquist、Per-Olof Thysselius、Mike Tudball、Karin Villers、Ctirad Vrana、Stefan Wallin、Roland Wester、Lars Wetterborg、Brian White、Lars Wiktorin、Charlotte Wranne、Jan Wunsche。

此外，多年来许多人为本书的主要作者提供了个人支持，为此向他们表示衷心的感谢。他们是：Dines Bjorner、Tore Bingefors、Dave Bulman、Larry Constantine、Goran Hemdal、

Bo Hedfors、Tom Love、Nils Lennmarker、Lars-Olof Noren、Dave Thomas以及Lars-Erik Thorelli。

最后尤其要感谢以下人员

Mike Devlin，Rational软件公司的总裁，感谢他始终坚信对象工厂过程可以作为一种产品，从而帮助全世界的软件开发人员开发软件，还要感谢他一直支持利用有效的软件过程作为软件工具开发的“驱动器”。

最后，我们还要感谢Philippe Kruchten，Rational统一过程（RUP）的主管，以及Rational过程组的全体成员，是他们将对象工厂中最好的东西与Rational公司的丰富经验及UML综合集成在一起，并最大程度地保留了各自的有价值的东西。此外，如果没有Philippe的个人承诺和坚忍不拔的意志，Rational过程组就不可能完成构造世界上所能见到的最好的软件过程这项任务，我们也不可能成功地完成本书的编写。

过程的突破

通过本书和相关的书籍、在线版本以及工具，我们可以得出这样的结论：软件开发过程已经成熟。统一过程从众多的来源获得灵感。现在，它已被广泛采用。它提供了一种使管理人员、开发人员和项目相关人员能从中获得对统一软件开发过程了解的公共媒体。

然而，我们仍有大量的工作要做。开发人员必须学习统一的工作方法，项目相关人员和管理人员必须支持他们。对于许多软件组织而言，这种突破仅仅是一种潜在的可能，而你可以将其变为现实。

Ivar Jacobson
Palo Alto, California
December 1998
ivar@rational.com

参考资料

[1] Peter F. Drucker, “The Discipline of Innovation,” *Harvard Business Review,* May-June, 1985; reprinted Nov.-Dec. 1998, pp. 149–157.

[2] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard, *Object-Oriented Software Engineering: A Use-Case Driven Approach,* Reading, MA: Addison-Wesley, 1992.

[3] Ivar Jacobson, Maria Ericsson, and Agneta Jacobson, *The Object Advantage: Business Process Reengineering with Object Technology,* Reading, MA: