HZ BOOKS
华章教育

高等学校计算机专业规划教材

UML建模分析与设计

基于MDA的软件开发

杜德慧 编著

UML Modeling and Design

Software Development Based on MDA

机械工业出版社
China Machine Press

高等学校计算机专业规划教材

UML建模分析与设计

基于MDA的软件开发

杜德慧 编著

UML Modeling and Design

Software Development Based on MDA

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

UML 建模分析与设计：基于 MDA 的软件开发 / 杜德慧编著．—北京：机械工业出版社，2018.9
（高等学校计算机专业规划教材）

ISBN 978-7-111-60959-9

I. U… II. 杜… III. ①面向对象语言－程序设计－高等学校－教材 ②软件开发－高等学校－教材 IV. TP312.8

中国版本图书馆 CIP 数据核字（2018）第 217210 号

本书根据新的 UML 建模标准，重点讲述 UML 的基本概念及建模元素，以模型驱动的方式从不同的视角构建系统的模型，包括静态模型和动态模型。其中，重点讲述 UML 的用例模型、类模型、活动图模型、状态机模型、顺序图模型等，并结合实际案例帮助读者掌握灵活使用 UML 的各种模型图来设计、构建系统的模型的方法。

本书可作为高等院校软件工程、计算机及相关专业的教材和教学参考书，也可以作为渴望掌握 UML 及基于 UML 的模型驱动式软件开发方法的软件开发者的参考书。

出版发行：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：佘 洁 责任校对：李秋荣
印 刷：北京市荣盛彩色印刷有限公司 版 次：2018 年 10 月第 1 版第 1 次印刷
开 本：185mm×260mm 1/16 印 张：14.5
书 号：ISBN 978-7-111-60959-9 定 价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

前　言

模型驱动式软件开发方法已经成功应用于大型、复杂软件系统的设计和开发，受到工业界和学术界的一致认可。模型驱动开发的核心是，根据系统的需求构建、设计系统的模型，并借助模型转换及代码生成技术等实现快速开发高质量的软件系统。其中，模型是整个软件开发过程中的主要制品之一，一切工作都将围绕模型的设计、构建、模拟、验证展开。这种开发方法将快速应用到特定的领域，能够有效提高面向特定领域的软件设计、开发的效率和质量。因此，如何使用标准的建模语言构建系统的模型是软件设计者面临的一个主要问题。本书根据新的 UML 建模标准，重点讲述 UML 的基本概念及建模元素，并结合具体的案例分析，以模型驱动的方式从不同的视角构建系统的模型，包括静态模型和动态模型。

本书目标

通过阅读本书，读者可得到以下几方面的收获：

- 掌握模型驱动开发方法的基本思想、开发过程。
- 掌握 UML 的基本概念、模型、建模规则，学会如何使用 UML。
- 以 UML 为基础建模语言，结合模型驱动开发方法进行实际案例分析、建模、开发。

本书的组织

鉴于 UML 在软件设计、开发过程中的重要作用，故撰写本书。本书可作为高等院校软件工程、计算机及相关专业的教材和教学参考书，也可以作为渴望掌握 UML 及基于 UML 的模型驱动式软件开发方法的软件开发者的参考书。本书共 16 章，其中，第 1 章概述模型驱动开发方法及 UML 在模型驱动开发方法中的重要作用，并明确指出本书将结合 RUP 开发过程和基于 UML 的模型驱动开发方法进行实际案例的设计、开发。第 2 章介绍 UML 的发展历程及其包含的主要建模元素。第 3 章综述 UML 所提供的公共机制，这些公共的建模机制将用在后续章节的各种模型的构建过程中。第 4 ～ 8 章遵循“用例驱动、以架构为中心、迭代增量开发”的思想，详细介绍 UML 用例图、类图、状态图、顺序图、活动图等，内容涵盖 UML 的静态结构建模及动态结构建模，充分体现了 UML 的多视角建模方法的有效性。其中，第 4 章详细介绍 UML 的用例图，并重点讲述使用用例图建模系统的需求。本章内容是全书

的重点部分，充分体现了“用例驱动”，后续章节将逐步介绍如何围绕用例图设计系统的静态结构和动态行为模型。第 5 章介绍 UML 类图，重点介绍类图的基本概念、类之间的各种关系。第 6 章介绍状态机模型，重点介绍状态图的基本建模元素，并详细介绍了状态机的语义模型及各种语法表示。第 7 章介绍的交互模型包括两种类型：顺序图和通信图。前者强调对象之间按照时间的先后进行消息交互，后者强调对象之间的拓扑结构，对象通过消息交互实现某一功能。两种模型图在语义上是等价的。第 8 章介绍活动图模型并详细讨论了使用活动图模型建模系统的业务流程及操作的实现过程。活动图模型强调的是活动与活动之间的控制流程。第 9 章介绍接口、类型和角色的基本概念，重点介绍如何使用接口建模系统中的接缝。第 10 章介绍包模型，它是 UML 建模过程中的产物，主要用于帮助划分系统的逻辑结构，以及帮助人们更好地理解系统的组成。第 11 章介绍构件模型，它用于建模系统的功能模块划分，重点介绍了构件的接口表示及构件之间的关系表示。第 12 章介绍 UML 的部署图，它主要用于对如何将软件系统部署到硬件节点上建模。第 13 章重点介绍最新的 UML 扩展语言 SysML 和 MARTE，向读者展示了如何使用 UML 支持的扩展机制进行建模语言的扩展，以满足特定领域的需求。第 14 ～ 16 章通过完整的案例分析展示了 UML 的各种模型的具体应用，以帮助读者进一步归纳、总结各种 UML 模型在实际建模过程中的应用。

本书的特色是以模型驱动式软件开发为指导，以 UML 的多视角建模为主线，结合案例开发全面介绍基于 UML 的建模方法，帮助读者掌握 UML 的语言构成、建模方法及具体应用。此外，每章配备相应的习题，以帮助读者掌握各章的知识点。

致教师

本书旨在提供 UML 的一个广泛而深入的概览，可以作为高年级本科生或者一年级研究生的 UML 建模课程的教材。根据授课学时、学生的背景和教师的兴趣，可以选择性地教授本书的各个章节。例如，如果想完整讲授 UML 的所有建模元素，可以逐章教授前 13 章的授课内容。若想结合具体的案例讲述各个模型的实际使用情况，可以考虑增加后面的第 14 ～ 16 章的内容，作为实际动手操作的案例练习。

每一章最后都给出了习题、思考题，可帮助学生更好地理解每一章的内容。有些习题可作为研究讨论课题。本书的参考文献可以帮助你查找正文中提供的概念和方法的来源、相关课题的深入讨论和可能的扩展研究文献。

致学生

我们希望本书能够帮助你了解和掌握 UML 所包含的基本建模元素、各种模型，

并能够熟练使用 UML 的建模方法，以模型驱动的方式开发软件系统。特别是，你可以了解模型驱动开发的核心思想及开发理念，并结合 UML，将其应用于你的具体软件开发过程。

为了更好地使用本书，你需要具备的预备知识包括：

- 基本的面向对象设计的知识，掌握一种面向对象开发语言。
- 软件开发的基本知识、软件工程的背景，了解常用的软件开发过程。

另外，需要说明的是本书中的内容是按照 UML 支持的各种模型组织的，为了更好地体现模型驱动开发的思想，我们将各种 UML 图形称为模型，这样更符合模型驱动的思想，构建系统的各种模型是整个软件开发过程中的主要工作。

在本书的组织、撰写过程中，研究生黄平、白新、管春琳、咎慧、敖义等参与了相关章节的模型图的制作、修改，以及文字的校对工作，在此特别感谢他们的辛勤付出。此外，在书稿的形成之初，我们将它用作本科专业课程的教材，在使用过程中，本科生孙雨晶、侯峥欣等对本书的第 14 ～ 16 章提出了中肯的修改意见。由于能力和时间有限，书稿中难免存在一些缺陷和不足之处，望读者不吝指教。

作者

2018 年 6 月

教学建议

本书内容分为16章，每一章的主要内容与课堂教学的学时安排如下。

第1章简单介绍模型驱动式软件开发方法、建模的重要性、UML建模在模型驱动开发中的地位和作用，以及迭代、增量的软件开发过程RUP。此外，补充讲解了面向对象的核心思想及关键技术，作为本书的基础知识部分。（建议学时：2）

第2章系统、详细地介绍UML的发展历程，概述UML建模语言的特点、使用的建模场景，并简要介绍UML标准建模语言所提供的基本建模构造块及其支持的公共建模机制和扩展机制。本章内容为后续章节的内容进行了铺垫。（建议学时：4）

第3章对UML的公共机制进行详细讲解，介绍注释、修饰、扩展机制的具体应用，为后续章节中详细的建模过程提供支持。此外，介绍了UML的扩展机制及其使用原则。本章内容是UML的基础部分，后续章节的案例介绍中会用到本章讲解的基本建模原则。（建议学时：4）

第4章遵循本书提出的“用例驱动、以架构为中心、迭代增量开发”方法，重点讲述如何构建用例模型及其包含的基本建模元素、常用的建模方法等。本章内容是本书的重点。（建议学时：8）

第5章针对如何构建系统的静态模型，从描述系统的用例模型出发，介绍如何构建类模型以及类模型的基本建模元素，重点掌握类的概念及类之间的关系，并对类图的建模方法、正向工程等进行了系统讨论。（建议学时：6）

第6章针对如何构建系统的动态行为模型之一——状态图，介绍其基本建模元素（包括状态、迁移、事件等），讨论如何基于状态图通过正向工程生成系统的代码，重点讨论状态图的基本概念、建模方法。（建议学时：6）

第7章详细介绍顺序图模型的基本概念、建模方法、建模原则等，结合实际案例详细讲述如何使用顺序图建模对象之间的消息交互，强调事件发生的时间顺序。此外，介绍了与顺序图语义上等价的通信图。（建议学时：6）

第8章系统地介绍活动图模型，包括活动图的基本概念，如活动、控制流等，重点介绍如何使用活动图建模系统的业务流程。（建议学时：6）

第9章详细介绍UML所提供的接口、类型、角色等建模元素，并讨论如何使用这些建模元素对系统中的接缝进行建模，以及对系统中的静态类型和动态类型建模。

（建议学时：4）

第 10 章详细介绍如何使用包模型对系统的逻辑结构进行划分，详细讨论包模型的基本概念及常用建模技术。（建议学时：4）

第 11 章详细介绍构件模型的基本概念、构件及其接口的表示方式，详细讨论构件模型的建模方法及常用建模技术。（建议学时：4）

第 12 章详细介绍如何使用部署模型建模系统的硬件、软件之间的部署情况，并讨论部署图的常见建模场景。（建议学时：4）

第 13 章针对 UML 的扩展机制的具体使用情况，系统地介绍两种具有代表性的 UML 扩展语言——SysML 和 MARTE，它们代表了 UML 的最新发展方向。（建议学时：2）

第 14 ～ 16 章是本书的案例分析部分。为了更好地展示前 13 章的建模元素、建模方法的具体使用，最后 3 章详细讨论了三个具体案例，分别从不同的建模视角展示 UML 的各个模型的建模能力，以便读者深入掌握、理解 UML 及其具体使用方法。

建议前 13 章的教学学时数为 60 学时（第 13 章可以作为选读的内容），任课教师可以根据教学安排来调整学时和选择重点介绍的内容。在前 13 章内容的讲述过程中，可以结合后 3 章的案例进行展开讲解，也可以最后统一讲解案例。

目　录

第1章 概 述

1.1 模型驱动开发方法

传统的软件开发以手工编码为主，随着软件系统的规模日益增大及其复杂性不断增加，传统的软件开发方法已无法满足软件系统快速发展的需求，人们迫切需要解决软件开发面临的危机，实现快速开发高质量软件系统的目标。此外，需求分析和设计阶段产生的文档、UML 模型与代码之间的同步也变得越来越困难。为了解决这些问题，对象管理组织（Object Management Group，OMG）提出模型驱动架构（Model-Driven Architecture，MDA），其目标是将软件的开发行为提升到更高的抽象层次——模型层。以模型驱动的方式开发软件，整个开发过程中产生的核心制品是模型而不仅仅是代码，从需求分析、设计实现，到最终的代码生成阶段，每个阶段的模型制品可借助模型转换工具，实现不同抽象层次的模型之间的转换，最终将系统的模型映射生成 C++ 代码或者 Java 代码等。目前，现有的建模工具已经能够实现部分或全部的代码自动生成，大大提高了软件开发的效率和质量。例如商业版本的 UML 建模工具 Enterprise Architecture（EA），能够支持 UML 建模及模型到代码的生成。模型已然成为开发过程的主要产品，设计、构建模型成为软件开发过程中的重要任务。模型驱动的软件开发方法围绕着模型的分析、设计、重用、模型到代码的自动生成技术展开，能够有效提高软件系统开发的效率。

以模型驱动架构为基础，软件开发方法也随之发生了改变，其中，具有代表性的模型驱动式软件开发方法（Model Driven Software Development，MDSE）是软件工程发展的一个重要方向。该方法是一种以建模和模型转换（Model Transformation）为主要途径的软件开发方法，目前已经被广泛应用于大规模、复杂的软件开发。与其他软件开发方法相比，模型驱动开发方法的特点主要表现于该方法更加关注为不同领域的知识构造其抽象描述，即领域模型（Domain Model）。基于这些代表领域概念的模型开发软件系统，并通过自动（半自动）的模型转换完成从设计向实现的过渡，从而最终完成整个软件系统的开发。

模型驱动开发方法的优势在于，其使用更接近于人的理解和认知的模型，尤其是可视化模型，有利于设计人员将注意力集中在与业务逻辑相关的信息上，而不用过早地考虑与平台相关的底层实现细节。尤其是在面对不同应用领域时，模型驱动方法强调使用方便灵活的领域相关建模语言（Domain-Specific Modeling Language，

DSML）构造系统的模型，基于领域知识实现领域专家、设计人员、系统工程师以及架构师等不同人员之间的良好沟通。

对象管理组织（OMG）提出的以模型为中心的软件开发框架性标准——模型驱动体系结构，受到了来自学术界和工业界的普遍关注。MDA 整合了 OMG 在建模语言、模型存储以及模型交换等方面的一系列标准，形成了一套基于模型技术的软件系统开发方法和标准体系。

MDA 是以模型为中心的软件开发模式，其将模型分为三个抽象层次：计算无关的模型（Computing Independent Model，CIM）、平台独立模型（Platform Independent Model，PIM）和平台相关模型（Platform Specific Model，PSM）。借助模型转换技术，可将抽象层次高的模型转换为抽象层次低的模型，通常，人们首先根据系统的需求，构建系统的 CIM，主要用于领域知识的共享。设计者的主要任务是构建系统的 PIM，然后，通过模型转换生成相应的 PSM，最终自动或者半自动地生成相应的代码。通过使用模型驱动的软件开发，能够有效提高软件开发的效率和质量，实现系统模型的重用，对于大型的、复杂系统的软件开发具有重要意义。

在后续章节中，我们会详细讲述如何遵循模型驱动式软件开发方法，使用 UML 构建系统的 PIM，为解决大规模、复杂软件的开发提供可行的方案。

模型驱动开发方法的架构层次图如图 1-1 所示：从 M0 层到 M3 层，建模的抽象层次越来越高，其中 M0 层是实例层，主要针对应用系统中涉及的实例或者要建模的具体实例对象，例如某个具体的电影（哈利波特）。因此，M0 层代表的是现实世界，是我们需要设计、开发的实际系统。M1 层是对 M0 层的抽象，如将电影名称抽象为 Video 类，M1 层即建模待开发的系统；M2 层是对 M1 层的抽象，即抽象出 M1 层本质的、共性的建模元素，M2 层也称为建模语言层，这里具体表现为如何使用 UML 类图中的建模元素如 class、attribute 等描述 M1 层的类图；对 M2 层继续进行抽象将得到 M3 层，用于建模元模型的层次，即元建模的基本结构 MOF，其提供了基本的建模元素及建模方法应该遵循的元建模机制。事实上这种层次化的建模方法并不是 MDA 软件开发所特有的，在计算机科学中很多领域如数据仓库相关的技术、BNF 范式定义的某个语言的语法等，都采用了类似的层次化思想。

1.2　了解建模

建模技术能够有效帮助人们将复杂的问题简单化，并帮助人们快速构建出系统。在工业生产、飞机制造业等行业中，由于系统的规模庞大、系统内部结构复杂，通常人们会首先构建出系统的模型，然后再展开大规模的生产。如果没有建模，而是直接开始构建系统，必然会导致很多错误，甚至会影响系统的质量和安全。

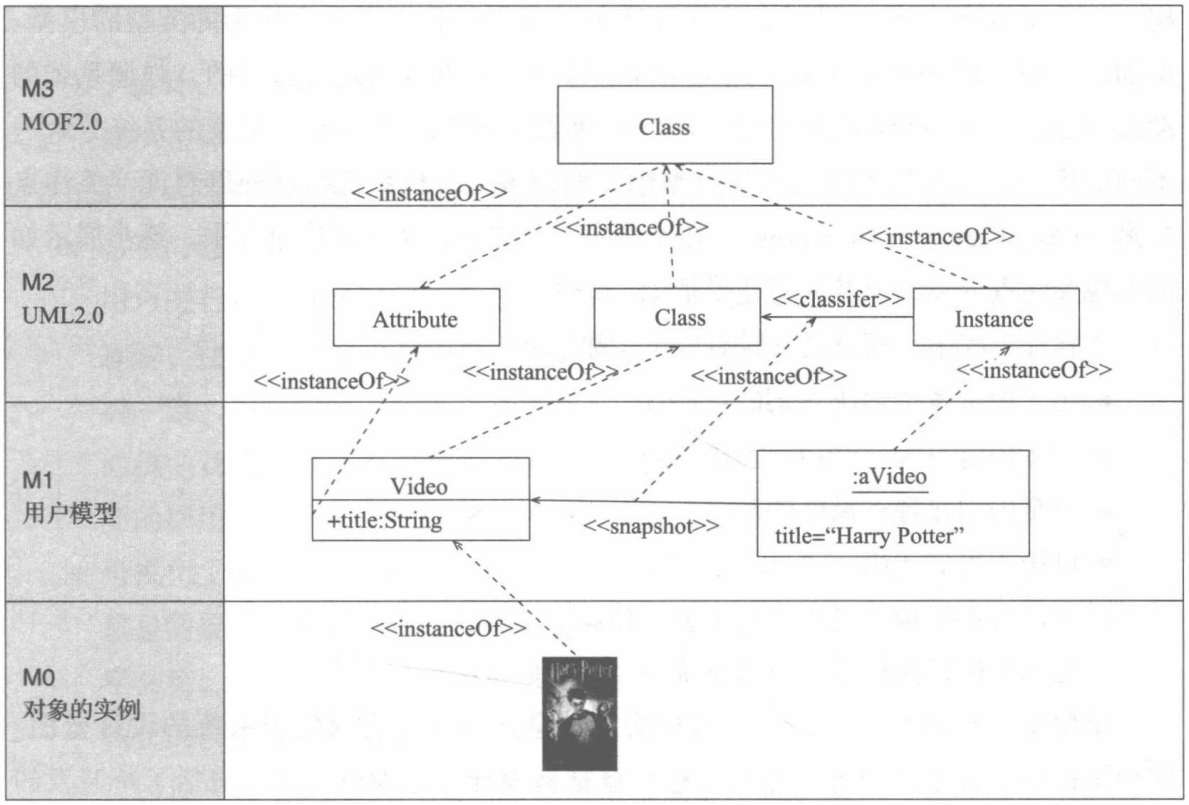

图 1-1 模型驱动开发方法的架构层次图

在软件开发过程中，随着软件系统的复杂性的增加，开发高质量的软件系统同样面临着各种挑战。软件开发也需要对系统进行建模，构建出软件系统的模型，这是提高软件质量的有效方法之一。大量的事实已经表明，不成功的软件项目其失败的原因各不相同，然而所有成功的软件项目在很多方面都是相似的。成功的软件组织有很多成功的因素，其中共同的一点就是在开发过程中有效地采用了建模技术。因此，通过对建模技术的合理应用，构建出满足系统需求的、正确的模型，将直接影响软件系统的质量。

通过上面的例子我们大致了解了建模这一概念，那么什么是建模呢？在了解建模之前我们首先须了解模型的定义。

模型（Model）是对现实世界的抽象、简化，是针对某一特定的目的而构建的、对某个实际问题或客观事物进行抽象后的一种形式化表达。建模的目的是便于理解、维护、演化现实的系统，帮助人们更好地了解、掌握现实的本质，并帮助开发者快速构建高质量的系统。

模型对系统的利害关系做出陈述，从特定的视点上将可以描述的细节从系统中分离，保留模型中最核心、最关键的地方。一个好的模型包括那些有广泛影响的主要元素，而忽略那些与给定的抽象层次不相关的次要元素。每个系统都可以从不同的方面、视角和采用不同的模型来描述，不同视角的模型一起刻画了整个系统的结

构、行为（功能）。换句话说，在实际建模过程中，我们需要构建不同类型的模型。例如，模型可以是结构性的，强调系统的组织；它也可以是行为性的，强调系统的动态方面，用于系统特定的功能。在工程实践中，工程师通常将复杂的系统分解为不同的模型，从而从不同的关注点帮助人们理解、分析系统，这种思想即“关注点分离”（Separation of Concerns）。在本书中，我们会在实际建模过程中，逐步展示如何实现关注点分离，以及如何建模抽象。

在软件工程中，模型通常具有以下用途：

- 用于展示各种解决方案的可能性。
- 用于构造系统，并用于测试阶段。
- 可帮助用户理解最终的产品。
- 可用于生成代码。
- 可作为系统构建过程中产生的文档。
- 可用于模拟执行待开发的系统。

模型是建模的核心所在，是对现实的抽象、简化，它提供了系统的设计蓝图。模型可以包含详细的设计，也可以包含概括性设计，这种设计高度概括了待开发的目标系统的结构、功能。好的模型包括那些具有高度抽象性的元素。每个系统都可以由不同的模型、从不同的方面来描述，因此，每个模型从语义上来说都是系统的抽象。与组成最终系统的代码和构件相比，系统的模型显得简单得多，也更容易理解，并可以支持模型的重用。此外，软件系统的模型可以协助开发人员审查、交流并校验系统，也可以帮助一个软件开发小组组织和协调他们的工作。

建模（modeling）是一种设计、构建模型的过程。它通过对客观事物建立一种抽象来表征事物并获得对事物本身的理解，同时把这种理解概念化，将这些逻辑概念组织起来，构成一种对所观察的对象的内部结构和工作原理的清晰表达。建模总是意味着突出和省略：突出必要的细节，省略不相关的内容。但是，怎么判断必要和不相关的内容呢？这一问题并没有标准的答案。相反，它取决于建造模型的目标以及模型的使用者。总之，通过建模，我们能获得复杂系统的抽象表示，能更好地理解所要构建的系统。建模的过程涉及建模方法、建模技术、建模语言、建模工具等，其在系统开发过程中的作用变得越来越重要。

软件建模即软件分析、设计建模，体现了软件设计的思想，在系统需求和系统实现之间架起了一座桥梁。软件工程师按照设计人员建立的模型，开发出符合设计目标的软件系统，而且软件的维护、演化也基于软件分析模型。因此，软件的分析建模在整个软件开发过程中具有重要的意义。软件建模阶段所产生的制品——模型的质量将严重影响整个软件系统的质量。

1.3 建模的重要性

关于设计、开发复杂的软件系统，首先要对复杂的系统进行分析、建模，因为人们很难从整体上理解复杂的系统，必须通过分析、构建该系统的模型，从而实现对系统的深入理解和掌握。通常来说，一个系统的模型具有以下特征：

- 易于沟通（communication）：为了创建正确的、满足用户需求的系统，需要与用户进行良好的沟通，有效地获取用户的需求。因此，需要给出每个人都能理解、使用的术语，使用户与设计者、开发者对于系统需求的理解、认识达成一致。沟通能够有效地捕获需求，在开发者与设计者之间达成对系统需求的统一认识，这是整个软件开发的起始阶段，对软件系统的开发具有至关重要的作用。因此，开发过程中构建的模型必须易于沟通、理解。
- 可视化（visualization）：对客户、专业人员和使用者而言，所有与系统有关的信息需要以大家都能够理解的方式表示出来。然而，根据实际经验，比起用文字交流，图形化的方式更加直观、易于理解，也更加容易被人接受。因此，可视化的模型有助于模型使用者之间的沟通与交流，更容易被人们理解和接受。
- 可验证（verification）：对于所构建的模型，其完整性、一致性和正确性是可验证的。也就是说，我们可以借助已有的模型验证分析技术，验证模型的完整性、一致性和正确性，从而确保模型是正确的、安全的。模型具有可验证性，对于确保模型的高质量是必不可少的。

从上述几个特性，我们可以看出，模型的构建对于整个软件开发过程有着重要的意义，可视化的模型使参与软件开发的人员之间的沟通交流更顺畅高效。此外，模型也便于人们对复杂的系统进行抽象，并且能够支持使用模型检测、验证技术分析模型的正确性。构建模型是开发大规模、高质量的软件系统的有效途径之一。

1.4 UML 建模

一个项目的成功是许多因素作用的结果，其中，有一个通用的建模语言标准至关重要。通常，建模语言必须包括以下几个部分：

- 模型元素——基本的建模概念和语义
- 符号——模型元素的视觉渲染、符号表示
- 准则——行业内使用的习语

面对日益复杂的系统，可视化和建模变得越来越重要。软件建模迫切需要一种标准化的建模语言来有效地建模系统，统一建模语言（Unified Modeling Language，UML）应运而生，其因良好的语言定义和广泛的工具支撑得到了工业界和学术界的广泛认可，被广泛应用于建立面向对象和基于组件的软件系统。