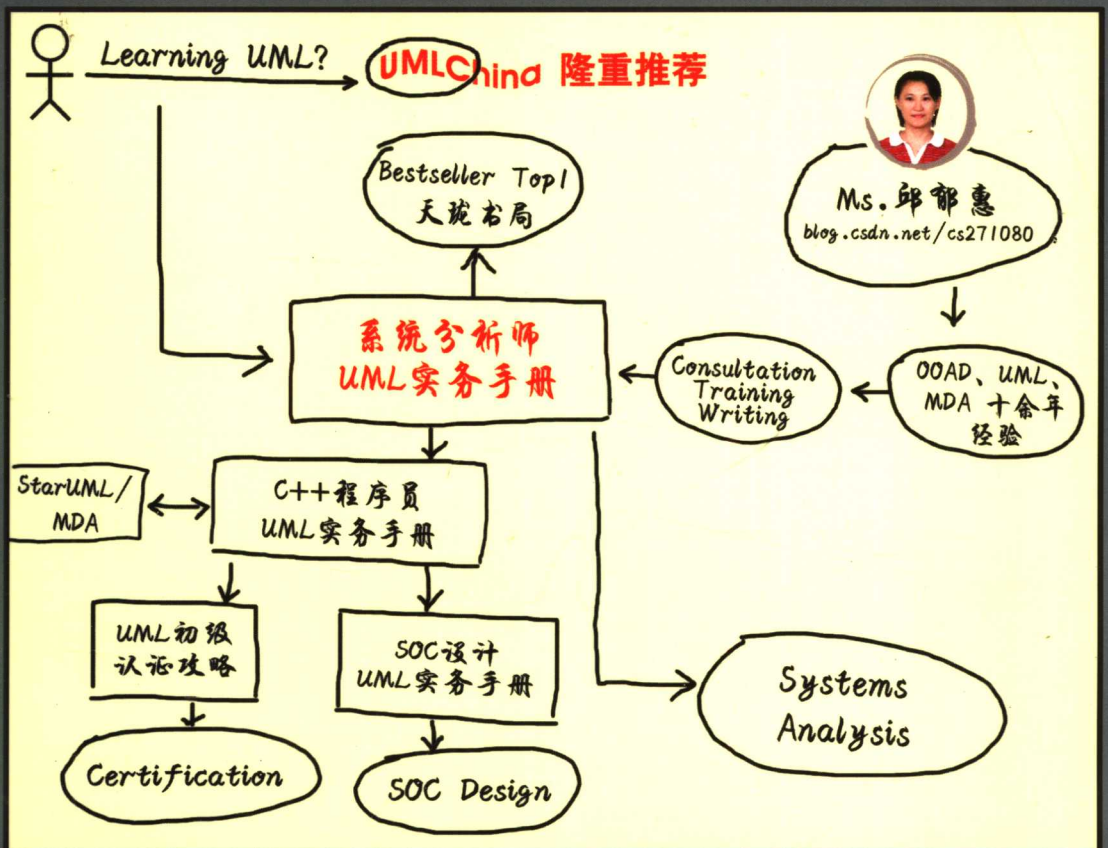
HZ BOOKS

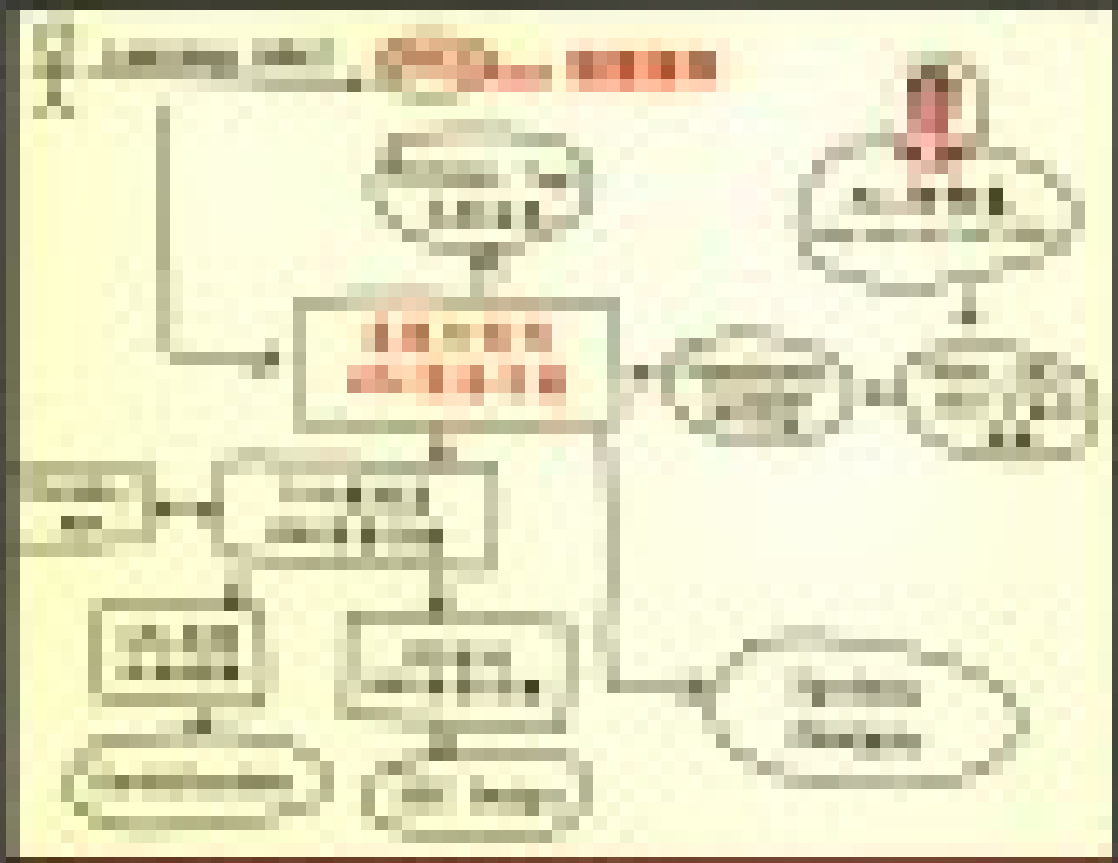
原创精品系列



系统分析师 UML实务手册

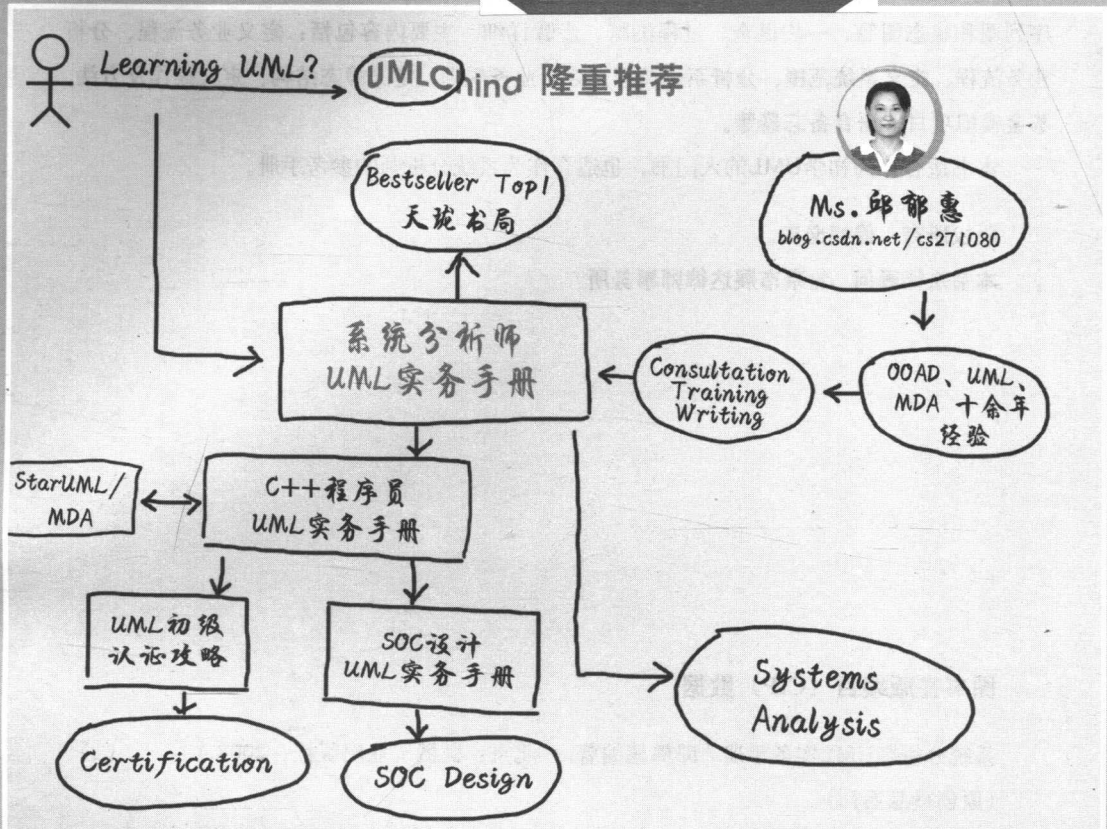
邱郁惠 编著

机械工业出版社
China Machine Press



系统分析师
UML实务手册

原创精品系列



系统分析师 UML实务手册

邱郁惠 编著

机械工业出版社
China Machine Press

本书通过一个完整的仿真实例，从需求到生成UML的用例图及叙述、活动图、类图、序列图和状态图等，一应俱全，过程细腻，步骤详细。主要内容包括：定义业务流程、分析业务流程、定义系统范围、分析系统流程、分析业务规则、定义静态结构、定义操作及方法、基金模拟项目、语音备忘器等。

本书适合作为初学UML的入门书，也适合作为系统分析师的参考手册。

图书在版编目（CIP）数据

系统分析师UML实务手册 / 邱郁慧编著. —北京：机械工业出版社，2008.4
（原创精品系列）

ISBN 978-7-111-23738-9

Ⅰ. 系…　Ⅱ. 邱…　Ⅲ. 面向对象语言，UML－程序设计　Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第036288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：王春华
北京京北制版厂印刷 · 新华书店北京发行所发行
2008年4月第1版第1次印刷
186mm × 240mm · 12.75印张
标准书号：ISBN 978-7-111-23738-9
定价：29.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线（010）68326294

推荐序

市场上已经有了很多UML书籍。在China-pub上搜“UML”关键词，已经有187个结果(2008年3月)。之前，台湾的OO领军人物高焕堂的UML书也在内地出版。既然如此，为什么还要购买和阅读邱郁惠老师带来的UML书籍呢？您可能会这样问。

因为这是由女性写就的建模书。软件开发这个行业，向来被认为不适合女性从事。在这个行业的开发人员，随着年纪增长，继续从事技术工作的本来就已经很少，女性更是少之又少，而邱郁惠老师10多年来一直研究OOAD、UML、MDA。由于作者是女性，所以本套书籍风格非常细腻，是真正的Step by Step。作者使用大量的截图，详细跟踪工具的每一步操作，一步步地画出UML图形；出现代码的地方，作者都给出了详细的、可以执行的代码。入门最难，对于新手来说，本书是极好的上手读物。

因为本套书籍的覆盖面广。《系统分析师UML实务手册》一书适合于开发企业应用的系统分析师阅读，通过一个基金系统的模拟案例，展示了从业务建模、需求到分析的过程；《C++程序员UML实务手册》一书专门为广大的C++程序员而写，使得C++程序员能借助UML进行建模，如虎添翼；《SOC设计UML实务手册》一书则面向硬件设计人员，通过一个录音芯片的模拟案例，展示UML如何与SystemC合作进行芯片设计。

因为本书使用的是开源的工具。有些UML书籍内容虽然不错，但是使用的是商业的UML工具来示例。商业工具往往价格不菲，出于学习需要的读者无法承受也不愿意付出代价来购买相应的工具来跟着实作，使得效果大打折扣。本书中所有的UML图都使用StarUML制作，这是一款开源、免费的UML工具，而且功能相当全面，这样，读者就免除了购买工具的苦恼。

希望本书成为真正能让您上手的第一本UML书。

UMLChina首席专家 潘加宇

前　言

本书有两大特色，第一个特色：我假想开发一套基金交易系统，以此做为贯穿每一个分析步骤的仿真案例。特别地，我仿真了SA与客户之间的对话，以此展示出SA在访谈当下会有什么样的提问，以及有什么样的思考，最后又画出了什么样的UML图。

SA的提问及思考的部分是我参与项目及教学的多年经验积累。我参与过的大部分项目都需要对项目成员进行边教边做的项目训练，为了能够在最短的时间内把这些成员训练好，让他们可以立即参与项目，除了基本的理论知识外，更多时候，我传授给他们的其实是实务上的提问技巧，同时也会告诉他们我是如何思考的。这样一来，他们不仅在不耽误太多项目时间的情况下学到UML，而且更能够应付项目工作。

当然，我的提问与思考绝非应用及学习UML的唯一方法，也不会是最正确或迅速的方法。同时，在项目时间紧迫的情况下，项目成员对UML的认知和体会绝对会有所局限。而我所做的这一切，无非是想让项目成员可以有比较低的门槛，对于在项目上应用UML可以有小小的成功经验。因为我十分相信，有了这样的正面的、成功的项目经验，将有助于点燃项目成员日后愿意花更多的时间和精力去深入学习、体会及应用UML。

第二个特色：我将分析步骤编号成CIM-1~3、PIM-1~4，一共七个分析步骤，这样的编号是我从研究DoDAF（美国国防部系统架构框架）所得来的灵感。这样做的一个主因是，在应用UML图时，同一款UML图可以有不同的用途，而初学者经常会有所混淆。

最常见的例子是，使用案例图（use case diagram）可以用来表达企业流程，也可以用来表达系统服务；如果在项目中，既进行企业流程的分析，又作系统服务的分析的话，我发现UML的初学者经常会对此产生困扰——为什么同样是使用案例图，但是一张叫做企业用例图（business use case diagram），但另一张却又叫做系统用例图（system use case diagram）。

为了降低这种困扰，我试着不以UML图为主，而是以分析设计步骤为主，告诉成员每一个步骤的重点是什么，以及采用的UML图？以此来降低UML初学者的学习门槛，同时也大幅节省了项目成员教育训练的时间。而且有了这样的思维，项目成员也可以认知到，UML图并不是非用不可的关键，每一个分析步骤所要呈现的观点才是真正的重点，所以日后有更好的技术时，当然可以将任何一款UML图取而代之。

此外，在进行项目时，一开始就可以评估哪几个分析步骤一定要做，哪几个分析步骤可以视情况添加，这样有助于时间的评估，也有助于经费的预算。项目成员也会很清楚地知道现在

做到了哪个步骤，接下来会进行哪个步骤，以及步骤之间的相关性。再者，可以将每一个分析步骤视为一个组件，尝试不同的排列组合，当然这个部分需要顾问的配合才能为项目打造出剪裁合宜的开发流程。

本书预设的读者

本书着重介绍UML的精华概念，删去琐碎且少用的概念，强调UML的实务性。相当有助于UML初学者大步跨过UML理论泥沼，UML现学现卖、立即上手。

再者，本书也非常适合熟知UML，或者一知半解却不知如何将UML应用于实务中的读者，书中提到非常多UML实务应用上的诀窍，可以让这类读者迅速脱离菜鸟生涯。

至于，学校学生就更适合读这本书了，这样可以缩短理论和实务的落差，尽快适用实务，to make more money!

本书结构

本书首先在第1章介绍UML的核心概念；然后在第2章中将分析的七大步骤浓缩成一个案例；随后在第3～9章分别详细研究七大步骤，并且在第10章中汇总整个范例；第11章则试着将七大步骤用于嵌入式系统开发。

第1章主要介绍UML，并且讲述重要的UML概念，奠定UML及对象导向(Object-Oriented)基础。同时，这章也会提到MDA（Model-Driven Architecture）概念，看看它与UML两者搭配造成的新潮流。

第2章主要将本书提到的分析七步骤浓缩成一章，让您可以先睹为快，只要花费一会儿的功夫就可以知道整本书的重点。

第3～9章贯穿基金系统范例，从头到尾细腻地讲述分析七大步骤，同时还会包含UML实务应用上的时机、诀窍、产出以及UML免费工具——StarUML的操作。

第10章归纳汇总出完整的基金系统范例，将分布在第3～9章的拼图一次展现，让您可以看到整个范例的全貌。

第11章特别将书中提到的分析七步骤用于嵌入式系统，示范本书在不同领域的应用。

目　录

第1章　为什么系统分析员需要学习UML

1.1　概述

系统分析员（System Analyst）的工作相当辛苦，他们站在用户与开发人员的中间，作为两者之间的沟通桥梁。系统分析员一方面需要向用户搜集并理清需求（Requirements），另一头又得急忙向开发人员提出清晰且明确的需求。

在项目进行期间，系统分析员除了得请神明保佑自己最好别误解或遗漏需求外，还得面对用户变更需求的反复性格，以及开发人员不愿因需求变更而白做工的强硬态度。这一切现象让系统分析员心力交瘁、焦头烂额。

在OO（Object-Oriented，面向对象的）与UML（Unified Modeling Language，统一建模语言）成了挡不住的潮流之后，程序员（Programmer）大量使用C++、Java等OO程序语言，同时也进一步带动设计师（System Designer）使用UML来表达关于OO设计。所以，设计师拿到系统分析文件后所做的第1件事情，便是将非OO文件转成OO的UML图，随后才能进行复杂的设计，并且生成各式的UML图，交由程序员按图编码。

然而，非OO的需求文件转成OO的UML图，不仅缺乏效率而且错误百出。许多公司开始意识到这样的问题，纷纷要求系统分析员学习OO概念，并且采用UML编写系分文件。这样一来，OO概念从分析开始，通过设计，一路贯穿到实现，沟通零误差。

UML是一套用来表达OO分析设计的国际标准语言，从1997年发展至今，吸引了相当多的爱好者，也发展出各式付费或免费的UML工具。挑选一套UML工具，作为系统分析员、设计师和程序员的工作平台，有助于提高工作效率。系统分析员生成的UML文件，可以交由设计师添加设计细节，最后再交由程序员按图编码。

1.2　UML并非万能

有些系统分析员对UML怀有高度期望，希望采用UML来搜集及编写需求之后，可以不再误解或遗漏需求，或者可以降低需求变更。不难想见，系统分析员经常得面对这些问题，当然期望学了UML之后，可以一劳永逸地解决掉这些问题。可是UML并非万能，无法根除这些本质性的问题，不过也不必悲观，总是有对策可以来处置需求误解、遗漏或变更的情况。

人跟人的沟通，本来就会产生误解，更何况用户与系统分析员的专业背景不同，所以当然

会在访谈的过程中充满大大小小的误解，既是在所难免，也就无法避免。再者，人脑并非计算机，谈着谈着，总是有多多少少的遗漏，无法在区区几次匆忙的访谈中，明确又清晰地条理分明。因而想在用户与系统分析师之间没有误会，这无疑是强人所难。

既然，访谈之中会有误解与遗漏，那日后反反复复多次地通过电话、电子邮件、会面来变更需求，或者是经过展示之后才发现错误而变更需求的烦人事件，也就不可抗拒地自然而然地发生了。

虽然，UML无法根除这些本质性的问题，但我们还是有对策可以面对这样的现象，试图减轻系统分析员的工作压力。对策如下：

- 使用UML图引导访谈，降低遗漏需求的情况。UML提供10多款不同功能的图，可以让系统分析员在访谈过程中，通过多款不同的图来理清需求各种不同角度的面貌，降低遗漏。而且，每款图都有它独特的组成图标，在绘制每一个图示时，都将引导系统分析员提出适时且重要的问题，以便搜集与理清需求。
- 快速生成可执行的程序片段，通过展示来凸显误解。
- 封装变化，让需求发生变化时，可以追踪到变化之处，迅速改版，并且不让变化起涟漪效应，向外扩散。

所以，在本书里，我们将要求系统分析员学习多款UML图，并且在编写需求时，不仅得生成让用户签字的文件，还得同时生成让设计师能接续设计的UML图件。这样的要求对于系统分析员确实相当为难，但是想要跟用户确认需求、签字同意，除了提交用户能够看懂的文字（Text）之外，别无选择。然而，想要满脑子OO概念的设计师和程序员，可以高效率且零误解地看懂系统分析员生成的需求文件，除了通过具OO概念的UML图外，目前看来似乎别无它法。

1.3 UML图

在本书中，系统分析员将学到两大类的UML图：行为图（Behavior Diagrams）与结构图（Structure Diagrams）。

行为图将引导系统分析员分析且理清“系统该做些什么”？系统分析员在绘制行为图时，可以聚焦在系统多方面的动作，像是系统与用户之间的交互，或者是某种对象（Object）因为事件的刺激以至于发生某些反应动作，以及一群对象交互完成某项服务，等等。系统分析员在访谈期间或结束之后，可以通过这些不同功能的行为图，获知系统多方面的行为。

系统分析员主要会学到下列的UML图，它们都归属于行为类：

- 用例图（Use Case Diagram）
- 活动图（Activity Diagram）
- 状态图（State Machine Diagram）
- 序列图（Sequence Diagram）

结构图将引导系统分析员获知“系统的重要组成元素是什么”？通常，通过结构图将引导系统分析员理清业务里（Business）的专有名词，以及专有名词之间的关系，以此作为系统的核心结构。其余，技术面的结构设计就留待设计师去伤脑筋了，这部分并不是系统分析员的工作职责。

系统分析员主要会学的UML图是类图（Class Diagram），属于结构图。

1.4 重要的OO及UML概念

OO是一种比较直接的设计思维，在设计软件时，让软件世界里的程序对象对应真实世界里的具体事物，以此来模拟真实世界的运作情况。观察真实世界可以发现，真实世界由各式各样的事物所组成，每种事物都有它特有的结构和行为，而且在联系起不同事物之后，还能够展现出丰富多元的能力。所以，OO软件也由各式各样的软件对象所组成，并且合力提供多样化的服务，以此参与企业运作过程。

OO概念是UML的基础，也就是说，系统分析员在学习使用UML的同时，其实就是在应用OO概念。换个角度来看，UML与OO两者互为表里，系统分析员脑子里运用的是OO概念，但是表达出来的需求文件内容却是使用UML图。

UML最大的特色在于它是图形语言，因此享有图形思考与表达的优势。所以在本书里，我们除了要求系统分析员在编写需求文件时需以UML图为主、文字为辅外，还要求系统分析员在访谈及讨论会议期间，皆需以UML图引导整个会议进行，并且以生成UML图作为其会议结果。

- 访谈即是绘制UML图。
- 讨论即是修改UML图。

想象一下，系统分析员带着安装了UML工具的笔记本电脑，到用户或领域专家的办公处所进行访谈，访谈的结果就是UML图。系统分析员启动UML工具的同时，就是访谈的开始。系统分析员开始动手绘制第一张UML图，也同步开始请教用户各项问题，一边绘图一边跟用户澄清认知，以便完成UML图。按照这样的过程，生成一张又一张的UML图。当这些UML图都绘制并且经过确认之后，系统分析员打印UML图交给用户，并且将UML文件传回给公司，完成此次访谈会议。

在访谈记录正式成为需求文件期间，用户有任何补充或者系统分析员有任何疑问，都切记以手上的UML图为讨论依据。同样地，系统分析员提交文件给设计师之后，如果设计师有任何疑问，需要开讨论会议的话，也必须以UML图为讨论依据。没有准备好UML图，就不要浪费时间开无谓的讨论会议，与会人员没有修正UML图，讨论会议就不算结束。

在了解OO与UML的相关性之后，接下来我们会介绍几项重要的OO概念，当然它们同时也是UML里的重要概念。

1.4.1 对象

真实世界由琳琅满目的事物所构成，但并非所有的事物都适合对应成软件对象（Object）。对于系统分析员而言，候选对象应该同时符合下列两项条件：

1. 在企业运作过程中，业务人员会使用到的专业事物或概念。

2. 而且在信息化时，系统也会用到，或者需要保存。

系统分析员在访谈业务人员时，可以提出类似下述的问句，以便获知重要的对象。如：

- 在执行这项工作时，你们会用到哪些专业概念？（探问对象）
- 你们在执行这项工作时，会需要用到哪些数据？（探问对象）

诸如此类的问句，有助于找到重要的对象。软件公司可以多向资深的系统分析员搜集此类的智慧问句，甚至编写成系统分析员通用的工作手册。

1.4.2 属性与操作

当试图去了解真实世界中任一事物时，有多元的角度可以去探寻。不过，系统分析员并不需要这么样深入了解对象。对于任何一种对象，系统分析员只需要针对下列两项问题去探寻：

1. 对象需要记录哪些属性（Attributes）？

2. 对象可以提供哪些操作（Operations）？

上述的问题可以说得更白话些，或许系统分析员可以向业务人员作如下的提问：

- 某物会记录什么数据呢？（探问属性）
- 某物可以提供我们哪些数据呢？（探问属性）
- 通过某物，可以让我们查到哪些数据吗？（探问属性）
- 某物可以做什么用呢？（探问操作）
- 有了某物之后，我们可以拿它来做什么事呢？（探问操作）

假设我们要开发基金交易平台，现在来向业务人员进行访谈，期间提到了“基金账户”。想象一下，系统分析员跟业务人员之间的模拟对话，大概会像这样子：

系统分析员问：基金账户会记录什么数据呢？（探问属性）

业务人员答：基金账户里头主要会记录总成本、总现值、总损益、总报酬率。

系统分析员问：有了基金账户之后，我们可以拿它来做什么？（探问操作）

业务人员答：有了基金账户之后，只要里头存有足够的钱，你就可以用它来申购基金，或者赎回基金。

访谈之后，系统分析员便得到了这个项目属性与操作的概貌，如图1-1所示。

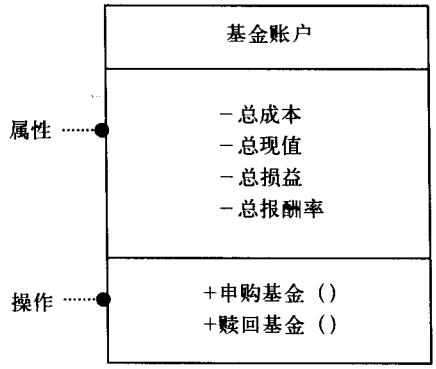


图1-1 基金账户的属性与操作

针对对象的各项属性，系统分析员还需要进一步了解它在企业里的定义、数据类型（Data Type）、可能的范围值或者初始值，更别忘了了解这项属性是怎么跑出来的？或许，系统分析员可以向业务人员作如下的提问：

- 可以请您（业务人员）用简单的一、两句话，解释某属性是什么吗？（探问属性定义）
- 可以请您举个例子吗？（判断属性的数据类型）
- 请问某属性有范围值吗？（判断属性的数据类型以及字段大小）
  - √可被接受的数字，最大最小为何？（数字类型）
  - √可被接受的字符串，最长最短为何？（字符串类型）
  - √预设的项目，有哪几个？项目异动的频率？（枚举类型）
- 请问某属性有初始值吗？（探问属性的初始值）
- 怎样做才能够得到某属性值（Attribute Value）？（探问属性值的获得方法）
  - √请问谁会提供这项属性值？（键入值）
  - √请问可以向哪里查询这项属性值？（查询值）
  - √请问计算公式为何？（计算值）
  - √请问可有独特的编码方式？（流水码或特定编码）

1.4.3 操作与方法

针对对象的操作，系统分析员还必须进一步探问how，了解操作的实现方法（Method）。简言之，操作是对象的what，而方法则是对象的how。然而，在我们用程序实现出对象之前，对象是死的，它哪会有什么方法来执行操作。所以，我们其实是想获知业务人员惯用的操作方法，然后将人为的操作方法转移给对象，成为对象的操作方法。

系统分析员访谈业务人员时，主要得获知方法的执行步骤（Procedure）、所需或者产生的数据、计算公式，以及企业的特殊约束。也许，系统分析员可以参考下列问题，向业务人员提问：

- 您（业务人员）通常都怎么执行某操作的呢？可以告诉我主要的执行步骤吗？（探问执行步骤）
- 请告诉我这些执行步骤会用到什么数据？以及会生成什么数据？（探问数据的输入及输出）
- 请告诉我这些执行步骤会需要使用到计算公式吗？（探问计算公式）
- 在执行某操作时，有没有什么重要的约束需要注意或遵守的？（探问特殊约束）

假设针对基金账户对象的“申购基金”这项操作的执行步骤，系统分析员向业务人员访谈的过程中，可能出现如下述的模拟对话：

系统分析员问：您通常都怎么执行“申购基金”这件事呢？可以告诉我，主要的执行步骤吗？（探问执行步骤）

业务人员答：是这样的，申购基金分为两种：一种是单笔申购，另一种是定期定额申购。

系统分析员问：谁在什么时候会决定是哪一种？还是可以同时包含两种？（判断是否要拆成两项操作）

业务人员答：客户一开始在申购基金时，就得二选一，决定是要单笔申购某档基金，还是定期定额申购。

（对于客户而言，单笔申购基金或者定期定额申购基金是两项不同的目的，所以系统分析员决定将原先的图1-1里的申购基金操作一分为二，改成图1-2的模样。）

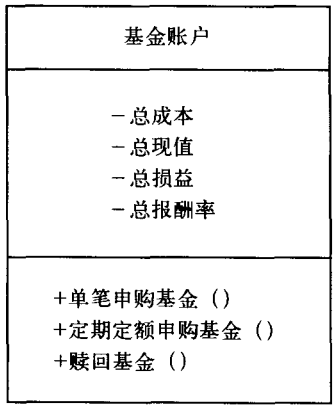


图1-2　申购基金操作一分为二

系统分析员问：请您挑比较简单好懂的一种，告诉我您主要的执行步骤？（探问执行步骤）

业务人员答：都很好懂啦，我就先说单笔申购的情况，好了。客户告诉我们单笔申购的基金名称、信托金额以及扣款账号，那我们这边只要确认这档基金有贩卖，然后客户指定的扣款账号里头有足够的余额可以付信托金额和申购手续费，这样就可以申购了。确认申购之后，我们会从客户指定的扣款账号中，支出信托金额和申购手续费。最后，我们会给客户一个申购交易的凭证号码，就完成单笔申购了。

系统分析员记录“单笔申购基金”的主要执行步骤如下：

1. 客户告知（基金名称）、（信托金额）以及（扣款账号）。

2. 业务人员确认（该档基金销售中）。

3. 业务人员确认客户指定的（扣款账号）里头有足够的（余额），可以用来支付（信托金额）和（申购手续费）。

4. 确认申购之后，我们会从客户指定的（扣款账号）中，支出（信托金额）和（申购手续费）。

5. 最后，我们会给客户一个（申购交易的凭证号码），就完成单笔申购了。

随后，系统分析员可以请业务人员协助确认这份记录，同时给予补充。但是，对于单笔申购基金的操作方法还没访谈完，系统分析员还得获知并确认输出入数据、计算公式以及特殊约束。

1.4.4 封装

面对真实世界中的多数物品，我们是“不知”亦能用，多数事件，我们也是“不知”亦能行。当然，我们对多数的事物也不是全然无知，只不过通常是有限度的知。因此，在物品或事件所蕴含的细节都被封装（Encapsulation）起来的情况下，我们还是可以使用物品、参与事件，毫无困难。

所以，当我们打造软件对象来仿真真实世界里的事物时，也模拟了这种封装特性。由于软件对象的封装性，所以对象之间仅透露足以进行交互的低限信息。对于对象的封装性，系统分析员要掌握下列要点：

- 已知操作。对象通常仅对其他对象透露自身的操作，彼此之间通过调用（Call）已知的操作来交互。
- 封装属性。每个对象封装属性值，不透露给其他对象。
- 封装方法。每个对象封装方法，仅对其他对象透露操作，但不透露其方法。

因此，系统分析员要特别注意，在分析规划对象的方法时，如果需要与其他对象交互，甚至是使用到对象本身的属性或操作时，切记要严守下列三条：

1. 不得直接提及对象的属性。

2. 也不得假设对象的执行方法。

3. 仅能够使用对象的操作。

从上述可以发现，我们为了保护对象的封装特性，会牺牲掉许多便利性。好比喝冰冻饮料时不能打开杯盖直接畅饮，而非得戳一个小洞，通过吸管小口小口吸取，十分不便。但使用吸管有一个好处，不小心打翻时，不会溅得四处都是，然后还得花钱再买一罐。

严守对象的封装性，有一个好处，当需求发生变化而需要改写代码时，变化会被局限在对象的属性和方法中，不会起涟漪效应，也不会发生牵一发而动全身的连锁反应。由于软件内部

的组成对象易于汰旧换新，所以软件的使用寿命延长，后续的维护成本也偏低，企业因此得到高投资报酬率，不过眼前首先要付出的代价是较高的开发成本。

1.4.5 类

“物以类聚”点出了对象（Object）与类（Class）之间的关联。针对一群相似的对象，我们本能地将它们视为一类（同一类）。访谈时，系统分析员听着业务人员谈论某一事物，甚至请他举出两三个具体且容易理解的实例，试图通过理解且分析这两三个同类型的实例，定义出适用于这些实例的类。

简言之，系统分析员通过分析真实世界中同类型的实例，以便定义出适用于软件世界的类。经过设计之后，程序员会为这些类编码。随后在软件系统执行期间，系统内部将遵照各类独自的定义，创建各类不同功用的对象。最后，这些不同类的对象将调用彼此的操作，合力完成各项系统功能。基金账户类与对象如图1-3所示。

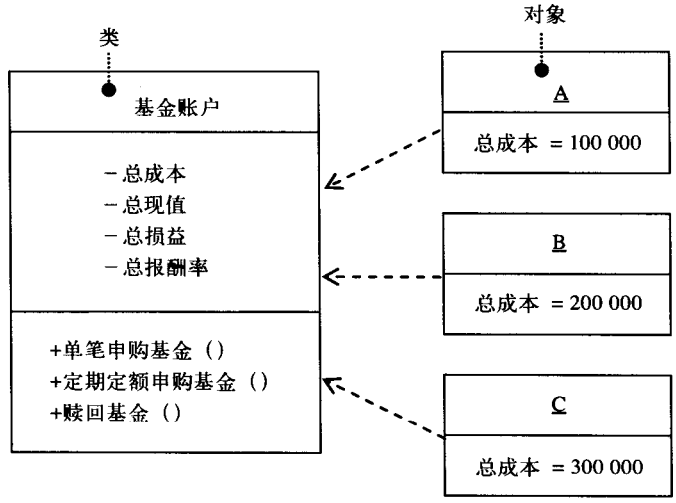


图1-3 基金账户类与对象

所以，一个简化后的开发程序以及系统运作情况，大致如上所述。归结起来，类与其对象之间细微的关联如下：

- （类）定义属性与操作，且所属（对象）共有这些属性与操作。
- 虽然同类（对象）共有属性，可是每一个（对象）却独有属性值。每个基金账户对象共有“总成本”这项属性，可是每个基金账户对象的总成本值都不同。请看图1-3，A基金账户的总成本值是100 000元，B基金账户的总成本值是200 000元，C基金账户的总成本值是300 000元，每个基金账户对象独有自己的总成本值。