软 件 工 程 技 术 丛 书

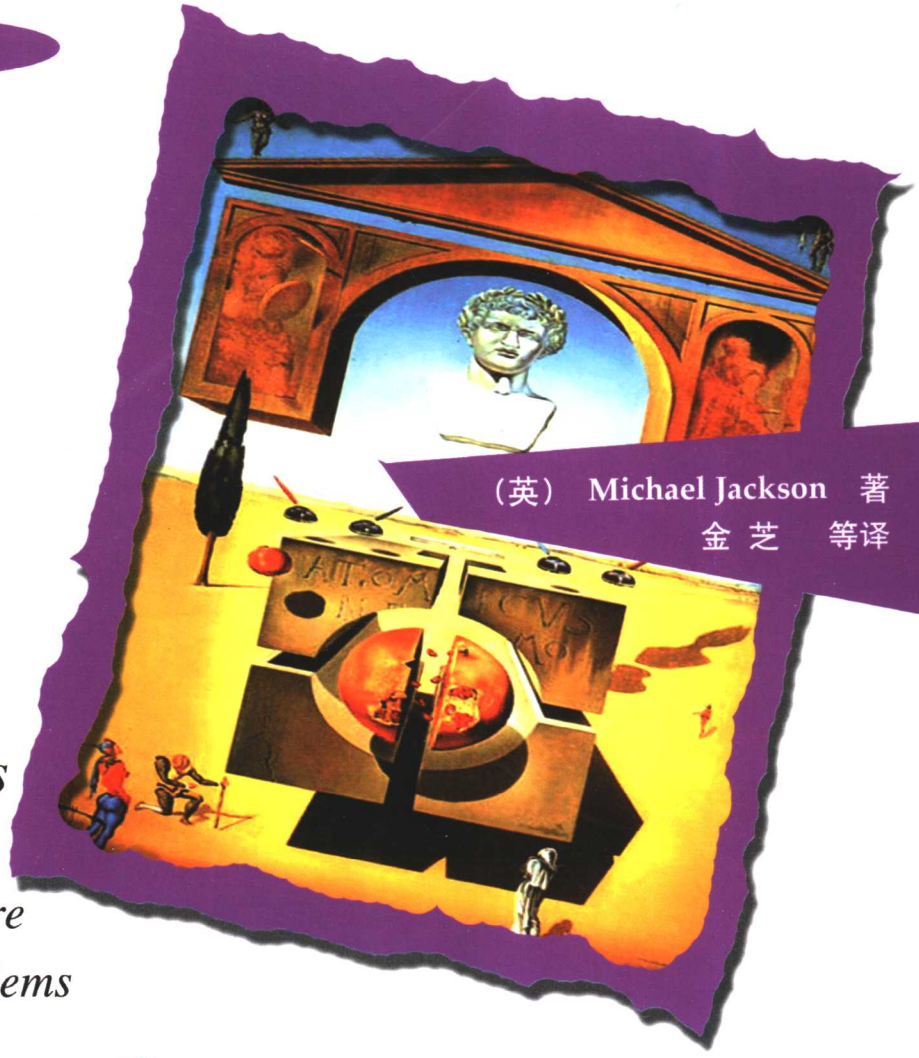
PEARSON Addison Wesley

HZ BOOKS

软件开发问题框架

现实世界问题的结构化分析

分析系列



（英） Michael Jackson 著
金 芝 等译

Problem Frames
Analyzing and
Structuring Software
Development Problems

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

软件开发问题框架

现实世界问题的结构化分析

分析系列

（英） Michael Jackson 著
金 芝 等译

Problem Frames
Analyzing and
Structuring Software
Development Problems

机械工业出版社
China Machine Press

本书分析了许多现实世界中的实例问题，讲述了如何在实际中识别和结构化问题。既给出了大问题也给出了小问题，展现了问题类的层次性本质，并讨论了每个问题的不同方面。

本书适用于系统分析、系统规格说明以及软件和需求工程领域的教师、学生和从业者，以及对软件开发的概念和智能工具感兴趣的任何人。

Michael Jackson: Problem Frames: Analyzing and Structuring Software Development Problems (ISBN 0-201-59627-X).

本书版权登记号：图字：01-2002-3594

图书在版编目（CIP）数据

软件开发问题框架：现实世界问题的结构化分析 /（英）杰克逊（Jackson, M.）著；金芝译. -北京：机械工业出版社，2005.2

（软件工程技术丛书　分析系列）

书名原文：Problem Frames: Analyzing and Structuring Software Development Problems

ISBN 7-111-15705-2

Ⅰ.软 … Ⅱ.① 杰 … ② 金 … Ⅲ.软件开发－结构分析 Ⅳ.TP311.52

中国版本图书馆CIP数据核字（2004）第128205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：迟振春

北京瑞德印刷有限公司印刷 · 新华书店北京发行所发行

2005年2月第1版第1次印刷

787mm × 1092mm 1/16 · 20.25印张

印数：0 001-3 000册

定价：45.00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

译者序

什么是需求？软件需求的真正含义是什么？我们为什么需要软件？需要软件来做什么？这些是研究需求工程的人们一直在试图回答的问题。从一般教科书和许多研究论文上，我们常常看到两种观点：一种观点认为，需求是用来求解问题或实现目标的条件或能力，这是站在用户角度上的观点；另一种观点认为，需求是系统或系统的成分所拥有的条件或能力，以满足一个合同、标准、规格说明或其他形式的文档，所有需求的集合形成系统或系统成分的后续开发的基础，这是站在软件开发者角度上的观点。应该说，这两种观点都反映了某个层次上软件需求的特征。

但是，对需求的认识仅仅停留在这个程度上就够了吗？我想大部分人会有否定的回答。因为如果把它们作为需求的定义，当我们面对现实世界，准备去捕获需求、提取需求并描述需求的时候，还是无所适从。我们常常会问：现实世界中有如此多样的事情要做，有各种各样的问题要解决，哪些是软件所能解决的问题？哪些是在捕获软件需求的时候所必须关注的问题？换句话说，软件需求的真正含义指的是什么？

强调对软件将要作用的环境进行刻画，将需求的含义指称到环境的描述上，是本书作者Michael Jackson一贯的观点，这个观点的形成可以追溯到1997年他发表在*Annals of Software Engineering*上的一篇文章中，在这篇文章中，他首次提出了软件要作用的环境、软件需求和目标软件之间的关系，即：

环境，软件 ⊢ 需求

这个观点目前正在引起国际需求工程界越来越广泛的重视，它以一个全新的视角来审视需求的内涵。软件可以作用在什么样的环境上？软件作用到环境上能够使环境发生什么样的改变？作用的特征是什么？作者认为，软件可以作用到的现实世界中的问题是软件需求的真正内涵，对它们进行结构化分析，是需求分析的根本出发点。而对上述问题的回答将对软件需求的捕获和分析起到指导性作用。本书正是试图给出在实际问题中如何去解决这些问题的。它从关注并定位现实世界的问题出发，按现实世界问题的分解、问题框架的关注点和风格、各种问题框架的变体和特殊关注点、问题的组合和成长式软件开发过程等，揭示了软件所能解决的基本的现实世界问题模式。

需要说明的是，译者在开始翻译本书时也是第一次接触这个全新的思想，虽然在翻译过程中阅读了作者在过去几年中所撰写的许多其他相关材料，也和作者就一些问题进行过深入讨论，但限于对这个问题理解的深度，加上时间仓促，难免有翻译不当之处，恳请读者指正。

2004年春于北京

前　言

软件开发问题

软件开发问题中的中心目标，是为具有某种现实用途的计算机系统创建软件。本书讨论的就是如何分析和结构化此类问题。

这些问题可以在许多不同的上下文中，以许多不同的形式出现。如果正在进行软件开发，你可能需要构建一个系统，用作银行账号和借贷信息的资源库和存取机制。你可能需要开发一个电话交换系统来转接当地的电话。你可能需要创建一个工具，来书写和编辑文本和图；或者创建一个控制设备来维护汽车的行驶速度；或者创建一个编译器将Java程序转换为字节码。几乎人类社会和物理世界的任何部分都可以作为软件开发问题的原始材料和上下文。

因为计算机具有这么多用途，并且在求解许多问题中发挥着中心作用，所以软件开发的实践比已经成熟的工程学科更不严谨。开发人员和开发项目存在着极大的不同，因此我们通常应该以一种在其他工程学科中很少需要的方式，从描述和结构化问题开始软件开发工作。因为在其他工程学科中，要解决的问题的变化要小得多。设计跑车的汽车工程师不需要问汽车是否必须能够载15个人，是否能够在水下开，是否能够载重10吨，或者是否能够以每小时100英里的速度向后退。“跑车”这个术语既明确了问题，也明确了这些问题和其他许多问题可接受的解决方案。

但是作为一个软件开发者，你很少会求解立即可识别的和容易理解的问题，通常必须从提问开始：这是哪种问题？这个问题准确地说是什么？它的用途到底是什么？计算机为了这个用途必须具有什么行为和特性？常常，软件开发看上去是从零开始的。

问题框架

当你分析问题的时候，要看它是什么样的问题，并且标识出要解决的关注点和困难，Java编译器中的关注点将完全不同于汽车制动系统中的关注点，你要考虑不同的事情，必须做出不同种类的描述，并且组合起来成为各种不同形式的论点。问题分析将使你从识别问题的层次迈向做出求解问题所需要的描述的层次。

但是大多数现实的问题只用像这样的两个层次来处理会太大、太复杂，还需要另一个层次：将问题结构化为相互作用的子问题的集合。如果结构化成功，子问题将比开始时的问题要小并且简单，并且它们之间的交互将是清楚的和可理解的。然后你能够独立地分析每个子

问题（它们本身也是简单问题），并且做出所需的描述。

本书的中心思想是在问题分析和结构化中使用问题框架。它们将通过定义不同的简单问题类来帮助你。当你结构化一个大的、现实的问题时，可以这样选择子问题，使得每个子问题都是由某个问题框架定义的简单问题。然后，当你分析子问题的时候，可以按照适合于它的问题框架来看它引出了什么关注点。这个框架展现了要解决问题所必须做的事情。

问题框架通过捕获它所涉及的部分世界的特征和相互的连接，以及可能出现的关注点和困难，来定义问题。所以，问题框架帮助你聚焦于问题，而不是陷入到设想解决方案上。通过强调计算机以外的世界、所需要的效果，以及客户最终判定是否实现这些效果的现实世界中的事情和事件之间的关系，问题框架做到了上面这一点。

问题框架吸取了设计模式中的许多精髓。设计模式深入其中去考察计算机及其软件，而问题框架则向外考察发现问题的世界。但是它们都识别和描述了一些不断再现的情景，提供了一个术语体系，在这个体系中，你所获得的经验和知识的每次增加，都可以在一个更大的架构下找到合适的位置，并且可以更有效地被共享和访问。在面向对象设计中，如果熟悉设计模式，你可以说“我们这里需要修饰器模式的一个实例”；同样，在问题分解中，如果熟悉问题框架，你可以说“这部分问题是一个工件问题”。如果能发现一个问题属于某个已知的问题框架，就可以利用与这个框架相关的经验。

问题框架的思想首先是在我的*Software Requirements & Specifications*一书中发表的，在该书中，只在概要中把它列为几个相关主题之一进行了简要介绍。本书在这个基础上进行了更新，讨论和说明了许多基本和组合的问题框架，以及许多风格和变体，并且考察了它们引出的一些关注点。另外，还考察和说明了如何在将现实问题分解为子问题时使用问题框架。

关注的视角

一些人认为，这种看待软件开发问题的角度太关注、太狭窄了，他们指出，计算机系统以及软件开发本身几乎总是存在于一个复杂和可变的社会、政治、道德和经济环境中。当你发现对一个系统的需求时，很可能就被牵扯进冲突的项目相关人员群体的社会协商过程中；当你做一个关于系统功能性的决定时，可能会更偏向于某一方；当你扩大系统的范围时，常常就在给一个群体政治权利，而让另一个群体付出代价；当你分析系统的目的时，就在探索它在经济和组织方面的结果。

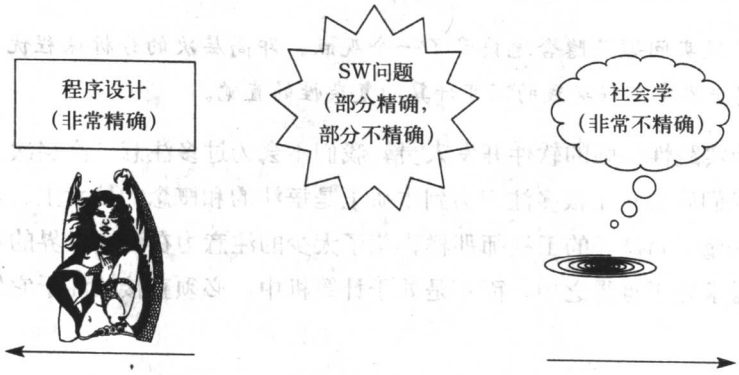
所以，从这个观点看，软件开发的主要关注点是社会的和政治的、组织的和经济的，按照问题来考虑这些关注点并不合理。这些关注点在许多开发中真的很重要，只不过本书中将忽略它们，我们不准备讨论怎样抽取需求，怎样产生业务案例，怎样管理项目、召集会议或

协商折中方案。我们将忽略这些事情，不是因为它们不重要，而仅仅是因为它们不是本书的主题。如果你想要理解某件事情，就不应该试图去理解每件事情。

相反，我们将试图理解在软件开发这个大背景中问题结构和分析中某些更关注的思想，我们的主题是系统应该带给世界的具体的、可观察的效果，实现这些效果的计算机行为，以及它们之间的联系。简言之，就是功能需求、软件规格说明以及从功能需求到软件规格说明的途径。

关注于问题

关注于软件开发中的问题并不是很容易的事情。原因之一在于，问题的一些方面会很精确，另一些方面则不那么精确，而所有这些都会引起你的注意。就像奥德修斯在从特洛伊回家的船上那样，你的航行途中前有锡拉女妖，后有大漩涡，你必须试图从中间穿过。



锡拉女妖和大漩涡

有些人认为我们对问题的视点太形式化、太狭窄，如果受这种论点所影响，你可能会偏离到右边，进入纯人类问题的世界——社会学和人种学的非精确世界，其中任何事情都不是完全确定的或完全准确的，那是一个重要的世界，但它忽略了软件和软件开发。

然而，如果你认为问题本身不如开发它们的软件解决方案那样有吸引力，你很可能会偏离到左边，朝向更精确的程序设计世界——变量、方法和对象类的世界，其中，布尔值或者是True或者是False，而绝不会处于两者之间。程序设计也很重要，但如果没有分析问题并且归结出程序必须做什么，则它将是没有用的。

在本书讨论的问题中，有许多精确的和不精确的成分，我们尽力平衡二者以理解和分析问题。

形式化性和非形式化性

如果你在锡拉女妖和大漩涡之间的正确航道上航行，并且成功地关注于问题，那么你的

描述必须有时是相当形式化的，而有时是相当非形式化的。形式化当然很容易观察到。一些软件开发者认为，当按照某种规则而不是凭一时的想象画图的时候，他们就是绝对形式化的。一些人认为，如果不依赖于严格的数学上的正确性证明，则开发无疑是非形式化的。

软件开发问题是关于现实世界的，系统必须在现实世界中起作用。所以，在问题分析中必须要做的大部分事情的形式化程度，取决于要处理的现实世界问题的形式化程度。能像问题允许的和任务要求的那样精确和形式化当然很好：在能够准确的时候为什么要模糊呢？但是物理世界的很多事情是不能被如实地甚至用纯形式的术语适当描述的，认识到这一点也非常重要。

在*Engineering and the Mind's Eye*这本关于机械和结构工程的好书中，Eugene Ferguson抱怨，许多工程灾难的发生，正是因为现代工程师将太多的注意力放在结构的计算和形式化的分析上，而很少注意到其结构也是物理现实世界的一部分。他认为：

> 工程教育的现实问题是隐含地接受了一个观点，即高层次的分析课程优于鼓励学生发展对现实世界中工程实践的不可计算的复杂性的直觉。

或许，作为实践性很强的软件开发人员，我们不会为过多注意“高层次分析课程”而感到内疚，但是我们确实花了很多注意力到本质上是语法的和概念的技术上。这让我们就像其教育过程受到Ferguson批评的工程师那样，花了太少的注意力在现实世界的不可计算的复杂性上。问题和需求处于世界之中，而不是处于计算机中。必须直接关注于它们，并且谨慎地描述它们。

真的有问题吗

有些人之所以不从问题出发来思考软件开发，确实另有原因。他们按共生的关系来看待每个系统及其环境，相互间不断地进化和调整。安装和使用一个系统（甚至提出要安装和使用一个系统）导致环境中的变化，而这种变化反过来影响这个系统在其以后的版本中的进化。“问题”从它被提出的那一刻起就被连续地修改，问题描述最好是问题在其将很快消失的某一刻的快照，没有像“需求”那样的东西，就是有的话，也是没有用的。

这经常被用作支持增量式部署和进化式开发的一个很好的论据，并且还可能是支持企图预计系统的效果、计算它将带来的环境中的变化并在系统开发中考虑这些变化的一个很好的论据。仅仅设计一座桥梁，想要应付眼前和能预想到的交通需求是不够的；必须能够处理这个桥梁本身引起的更多不确定的需求增加。

但是对于拒绝从问题和需求出发来思考而言，这种观点绝不是什么站不住脚的论据。当一个计算机系统（或者对一个系统增量的或者进化的改进）安装和运行时，它将肯定有一些

由软件所确定的特定的行为，这种特定的行为将在问题世界中产生效果和影响，而这些效果和影响可能是想要的或不想要的。要捕获需求，并且结构化和分析问题，就是尽你所能决定什么结果是想要的，并且创建将能带来这些结果的软件。就是说明年所需要的不同结果和不同软件，绝不能成为今年软件出现问题的理由。

对需求的不确定性和变化性的正确应对措施，不是放弃这个任务，而是要在更高的通用性层次上进行工作。有一种技术是将一些功能性编码为机器在执行时要解释的显式描述，然后用户可以通过修改这个描述来修改系统的行为。这种增加通用性层次的技术能够归于本书中讨论的问题分析方法中：它使用拥有*描述领域*的变体框架。这当然还可以归于设计模式的常用实践，其中它是用带有诸如*反射*、*元数据*或*类型对象*等名字的模式来表示的。

什么方法

问题框架的使用并不意味着一种特定的开发方法。一个原因是，不同的框架要求不同的方法，并使用不同的概念和表示法。想用一种灵丹妙药解决诸多软件开发困难的想法现在已经很少人相信了。但可悲的是它仍然存在。令人吃惊的是，实际上是在宣称对每种问题都可用的灵丹妙药仍然有人在设计和销售，并且似乎很流行。如果一个方法对每种问题都能提供帮助，那么不必期望它对某种特定的问题有太多的帮助。

问题框架并不提倡某一种特定的开发方法的另一个原因是，大多数现在被广泛应用的方法都强烈地面向解决方案。你需要面向解决方案的方法来帮助得出解决方案，但是它们在结构化和分析问题中所能给予的帮助很少，并且常常是一种纯粹的障碍。

问题框架以及相关的思想可以作为所有行动的前端；或者为你怎样扩展或修改实践提供建议；或者也许只是澄清它。如果采用了这些思想，就要循序渐进地去实践，而不要一下子彻底改变。你或许会发现其中的许多概念只是为已经有的思想和直觉起了一个名字，而为它们取名会使你在需要时更有意识地想到和使用它们。对于一些对你而言比较新的概念，它们会为一些你曾经感到困难但一直没有弄清的地方带来一线曙光。

本书的结构

本书的关键思想是将复杂和现实的问题分解为适合已知问题框架的简单子问题所组成的结构，因此其中混杂有大大小小的问题。一些比较小的问题没有实际意义，特别是用来示例最基本的问题框架的那些问题。但不要轻视那些没有实际意义的问题，它们以最直接的形式逐步地展现了问题类的本质，这使得当这些问题在更大问题下出现时，更容易被识别。

本书中的问题并不是按规模和复杂性的升序出现的，那样虽然合理简洁，但却会使本书

不太好阅读。较大的和较小的问题交织在一起，并且每个问题的不同方面都在它们看起来相关并提供了当前论点的一个好的示例的地方被讨论。

一些主题（比如描述性语言和表示法）贯穿了全书，而且，这些主题的各个方面都在它们出现的时候被引入。

这种非形式化的结构由两个附录来补充，一个附录总结了所使用的描述性语言和表示法；另一个附录提供了术语表，还有一个参考文献的统一列表。

本书适合你吗

希望本书对系统分析、系统规格说明以及软件和需求工程领域的教师、学生和从业者，以及对软件开发的概念和智能工具感兴趣的任何人都有所裨益。本书不是教科书，但它可以用作一些软件开发课程的补充教材。

有问题吗？

什么是锡拉女妖和大漩涡？

锡拉女妖是希腊神话中的女妖，生活在西西里和意大利之间的墨西拿海峡上的岩洞中，她将过往的船只拖进岩洞中并吃掉；这就像原本用来解决问题的精力被程序设计技术过程消耗掉。大漩涡是墨西拿海峡对面的致命漩涡，它将过往船只拖到海底；这就像问题被政治和社会因素弄得完全失控。奥德修斯和亚尔古英雄都成功地从这两个危险之间穿过。

所提到的讨论问题框架的两本书是什么？

Michael Jackson, *Software Requirements & Specifications: A Lexicon of Practice, Principles, and Prejudices*, Addison-Wesley, 1995。

Ben Kovitz, *Practical Software Requirements: a Manual of Content and Style*, Manning, 1998。

Ferguson关于机械工程和结构工程的书是什么？

Eugene S Ferguson, *Engineering and the Mind's Eye*, MIT Press, 1992。

在哪里可以找到将问题分类和结构化的其他思想？

此类文献在实践者的领域并没有很多，但在系统需求领域有一些很有趣的研究，系统需求与问题框架非常相关并且有一些与问题框架的思想非常紧密并行的思想。

Requirements Apprentice提供了需求的分类体系和需求模板（requirement clichés）的库，模板被划分成三个域：环境、需要和系统，系统可以是信息系统、跟踪系统或显

示系统，Requirements Apprentice本身是一个工具，帮助你搜索模板库，并为特定问题使用这些模板，可以在下文中读到关于它的描述:

Howard B Reubenstein and Richard C Waters, *The Requirements Apprentice: Automated Assistance for Requirements Acquisition*, IEEE Transactions on Software Engineering, Volume 17, Number 3, pages 226-240, March 1991。

对象系统模型和信息系统模型捕获了需求工程问题的抽象，对象系统模型的最高层分类包括诸如资源分配、工件处理和金融对象交换等。这些工作在下文中描述:

NAM Maiden and AG Sutcliffe, *Analogical Retrieval in Reuse-Oriented Requirements Engineering*, Software Engineering Journal, Volume 11, Number 5, pages 281-292, September 1996。

KAOS方法按照目标来看待需求，要构造的系统的目标被分解和精化，并分配给主体。一个主体可以是由硬件和软件组成的构件，或者是一个人，或者是物理问题环境的另一部分。目标被分成信息、安全等种类，目标之间的冲突被标识和解决；目标满足的障碍被检测和克服。KAOS在下文中描述:

A van Lamsweerde, R Darimont and Ph Massonet, *Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt* in Proceedings of RE95, the Second IEEE International Symposium on Requirements Engineering, pages 194-203, May 1995。

R Darimont and A van Lamsweerde, *Formal Refinement Patterns for Goal-Driven Requirements Elaboration* in Proceedings of FSE4, Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering, pages 179-190, October 1996。

Axel van Lamsweerde and Emmanuel Letier, *Integrating Obstacles in Goal-Directed Requirements Engineering* in Proceedings of the 20th International Conference on Software Engineering, April 1998。

致 谢

许多人对将问题框架的思想引入到开发阶段做出了贡献。

从我开始写这本书以来，Daniel Jackson就一直给予我帮助、鼓励和建议。就像以前一样，他认真仔细的评论使我避免了许多错误，他的耐心和大度让我受益匪浅。

Ben Kovitz在写作*Practical Software Requirements*一书时，从我早期的一本书*Software Requirements & Specifications*中偶然发现问题框架的思想，他将我加入到电子邮件联系人中，他对术语提出了极好的建议，并且对特定的和通用的软件开发问题提供了许多颇具启发性的认识。

Barry Cawthorne和我一同研究问题框架及其他相关主题，他提供了关于学术管理界的一个详细的实例，和我一同讨论，并指出我的早期思想的一些令人费解和错误之处。

Steve Ferg在关于问题框架的电子邮件联系中提供了有用的思想，并对本书的草稿做了非常有益的评价。

Richard Botting指出了sibling以前定义中的一个错误，现在我已经更正了。

Pamela Zave和我在AT&T Research公司一起工作了10年，主要工作是电信服务的分析和结构化，虽然我们没有明确地讨论过问题框架，但本书肯定有许多我在我们的长期合作中学到的东西。

许多其他人对问题框架的兴趣也鼓励了我，特别是：Dines Bjørner, Michael Caspersen, Hugo Fierz, David Garlan, Anthony Hall, Ralph Johnson, Neil Maiden, Lars Matthiassen, Jay Misra, Mary Shaw, Alistair Sutcliffe, Axel van Lamsweerde, Roel Wieringa, Jeannette Wing, Jim Woodcock，Amiram Yehudai。

感谢他们，他们让本书变得更好，其中仍然存在的瑕疵都是由于我的疏忽。

目　录

第1章 关注于问题

直接关注于问题而不是径直进入对解决方案的设计非常重要。计算机以及运行于其上的软件是解决方案；而问题则处于计算机以外的世界中。尽管你可能有很好的意图，但如果你按目前通常的方式来使用像“系统”和“模型”这样的词，并且试图进行无缝开发的话，会很容易将问题与问题的解决方案搞混。本书是关于问题的结构化和分析的，而不是关于解决方案的结构化和分析的。

1.1 解决方案之前的问题

这本书是关于问题的，我们将通过对所讨论的每个问题给出一个粗略纲要，来介绍这些问题。下面是第一个问题的粗略纲要：

病人监护问题

医院的ICU（监护室）需要一个病人监护程序，每个病人由一个模拟设备来监护，模拟设备用来测量诸如脉搏、体温、血压以及表面阻力等参数。这个程序按周期（为每个病人设置的）来读取这些参数，并将它们存储到数据库中。对每个病人来说，医生还指明了每个参数的安全范围。如果某个参数超出了病人的安全范围，或者模拟设备失效，则通知护士工作站。

每个人都会同意在开始开发解决方案之前，应该从关注问题本身开始。开发一个解决方案意味着考虑如同下面这些问题：

- 应该用什么SQL语句来写数据库？
- 应该怎样调度才能使每个病人都按所需要的频度得到监护？
- 系统中应该有些什么样的对象类？
- 病人列表应该保留在Java向量中吗？

而关注于问题意味着考虑如同下面这些问题：

- 所有要监护的病人都要被监护，还是只是其中一些要被监护？
- 是对不同的病人有不同的致命参数，还是所有的病人有相同的致命参数？
- 是由医生指明周期以及范围，还是其他什么人来指明？