第5章 问题驱动的方法

问题驱动的方法主要指问题框架方法，它是一种面向问题域的方法，它认为需求存在于现实世界中，存在于一组可识别的现实世界实体上，是关于现实世界实体产生的现象之间的一组期望关系。这组关系在没有引入待开发软件之前并不成立，待开发的软件就是用来促使这些关系能够成立。因此，问题驱动的方法认为需要从现实世界的实体及这种期望的关系出发，推断出待开发软件的需求规格说明。

本章介绍问题驱动的方法，含问题结构化分析的相关概念和技术，包括方法概述、问题框架的描述、基于框架关注点的分析过程，以及一些综合的问题关注点。

5.1 概　　述

软件需求工程首先需要关注待开发软件要解决的问题，而不是直接进行解决方案的设计，这是需求工程界公认的原则。问题驱动的方法遵循从问题出发这个原则，倡导从现实世界中寻找软件需求的源头，并从源头出发推断对软件能力的要求。

这种需求识别的理念适合于什么场景呢？图5.1示意性地给出了人机物融合应用场景中的问题定位视角，即软件要解决的问题来自包含人类社会和物理社会在内的现实世界，软

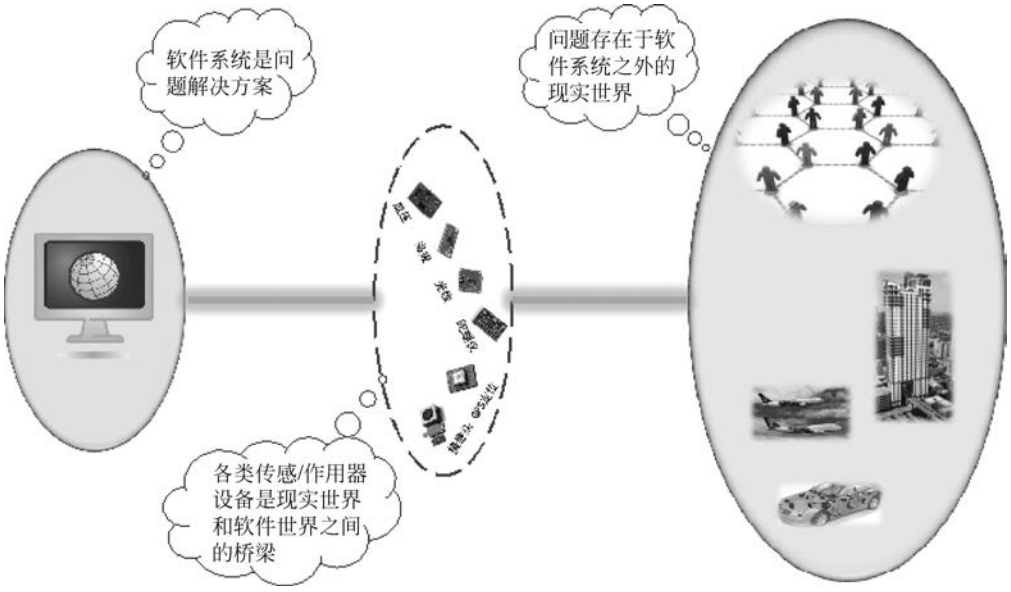


图5.1　问题和问题解决方案

件加上与其相关联的物理设备或人等现实世界的实体，构成一个人机物融合系统。这个系统是问题的一种解决方案，它通过由各类传感设备等组成的信息采集通道，接收来自现实世界的信息，然后根据其设计好的计算解决方案，产生问题求解过程中实施决策和控制的信息，并通过由各种激活器等组成的信息输出通道实施到现实世界中，从而带来期望的效果。因此，问题驱动的方法关于需求定位的思路特别适合于这类支持人机物融合应用场景的软件。

问题驱动的方法主要解决如何定位问题以确定软件问题的边界、如何表达问题以准确刻画需求、如何分解问题以控制问题的复杂度，以及如何根据问题框架（或模式）推断软件设计规格说明，等等。第 1 章开头给出的案例 1.2 是关于“智能家居系统”的场景描述，从这个场景出发，可以进一步了解最终用户理想中的智能家居（如案例 5.1 所示）。本节将结合这个案例回答前三个问题，最后一个问题将在 5.3 节中具体讨论。

案例 5.1　理想中的智能家居

你想要什么样的智能家居呢？我们可以想到很多让生活更方便的方式。例如：

早上起来，能知道室内温度和湿度、室外温度和空气质量等，以决定当天的衣着和行程。走进厨房，咖啡机正做着自己喜欢的口味的咖啡。面包机也打开了。当然，如果有厨房机器人的话，机器人会把早餐准备好、放到餐桌上。

上班时，能随时观察家中老人的生活起居。家中老人也能得到实时监护，包括定时测量血压、体温、脉搏等。这些身体指标都是由家庭医生专门指定的，如果指标超出了医生给定的安全范围，监护人能及时得到通知。如果监控设备失效，监护人也能收到提示。

还能随时查看当日、当月、当季或指定时间段的运动健康表，以查阅个人身体变化情况，保持身体健康。

家中有人时，室温尽可能保持在让人舒适的温度，如 26℃。另外，还需要从整体上考虑节能及居家安全和隐私保护的需求。

对家居智能化的期望给出了待开发系统面临的现实世界问题。在案例描述的智能家居场景中，软件开发的问题就是如何管理和调度如空调、温度感应器、医学监护仪等居家设备，协调它们的运行和能力，从而为生活起居带来舒适和便利。其软件开发任务就是设计并开发一个能实施设备管理和调度并协调它们运行的软件。这类软件将通过各种交互设备直接与现实世界进行交互并起作用，软件及其调度的设备实体构成满足最终用户需求的系统。

5.1.1　问题的定位

采用问题驱动方法，首先需要定位问题，关注点是“问题在哪里”。前面已经提到，软件要解决的问题处于现实世界中，问题涉及的元素需要到现实世界中去找，更直接地说，就是“待开发的软件将来会和现实世界的什么实体发生什么联系，并期望得到什么效果”。例如，上述的案例 5.1 中，那些需要管理、调度和协调运行的设备，以及需要服务的人和参与服务的人等，可能都是待开发软件未来需要交互的对象。这些现实世界的实体成为待开发软件

会涉及的现实世界对象，它们形成了待开发软件的边界，通过这些交互对象关联到的现实世界部分成为问题的边界。

图 5.2 示意性地给出了案例 5.1 中软件及其系统的问题边界的片段。其核心部分是待开发的智能家居系统软件，围绕着它的是一组软件能协调管理和调度或部分协调管理的现实世界实体。也存在一些现实世界实体，软件可能无法与它们建立直接的联系，需要如图 5.2 所示的传感设备和控制设备等为其架起桥梁。

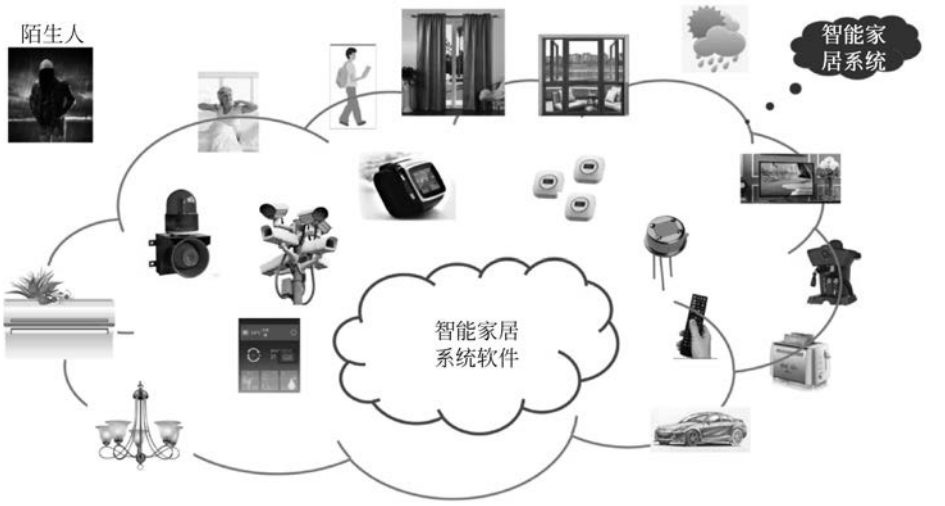


图 5.2　智能家居系统的问题边界(片段)

待开发软件及其与这些现实世界实体的直接或间接交互，形成了待开发软件的上下文，可以用上下文图来表示，它形象地展示了这个软件将处于的交互环境，同时还展现了软件与交互对象之间的接口，即软件如何与其环境实体连接。在问题驱动的方法中，软件称为机器领域，而其环境实体称为问题领域，机器领域和问题领域统称为领域。对待开发软件而言，有一类特殊的环境实体，如其他软件，它们也属于待开发软件的问题领域，但又区别于诸如天气、人和设备等物理实体，它们有一个专门的名字，叫设计领域，而其他的问题领域叫给定领域。

领域之间可以共享一组现象，形成交互关系，又称为领域间的接口。当其中一个领域激发了需要关注的现象时，另一个领域通过这个接口能同时共享这个现象。两个领域之间共享哪些现象，需要根据应用需求和问题领域的特性来决定。

上下文图涉及如下三类领域，其中，每个上下文图有且仅有一个机器领域。

- 机器领域：指待开发的软件，当前的软件开发任务就是设计这个软件，使得能使用它来实现问题领域中的现象之间的一组期望关系，以满足系统需求。
- 给定领域：指现实世界中的实体或物理设备，它们的属性不是通过假设或规定来获得，而是由自然规律或物理特性决定。问题分析的目标，就是根据其固有的特性，设计机器领域的行为，以建立与它们的共享现象，来影响给定领域产生期望的行为。例如，机器领域和可接收控制信号的空调，共享“开信号/关信号”等现象，当机器领域发起“开信号”，则空调将接受这个信号，引起空调处于开状态。

- 设计领域：这类领域是人工设计的领域，但对当前的待开发软件而言，它是已经存在的，可能是以前设计的软件。它们的属性不需要像给定领域那样按照物理法则或自然规律去获得，可以是约定好的信息表示。例如，待开发系统需要和银行系统之间有支付接口，银行系统对待开发系统而言是设计领域，当前机器领域和它共享涉及支付、账号等的现象。随着软件设计的不断细化，当前机器领域也可能会分离出一些组件（如模型、信息存储等），并作为问题领域显式地表达在细化的机器领域上下文图中，成为细化层次上的设计领域。

上下文图的图元包括长方形框，用于表示领域。其中，带双竖线的长方形框表示机器领域，带单竖线的长方形框表示设计领域，不带竖线的长方形框表示给定领域。两个长方形框之间的实线连线表示两个领域间共享现象的接口。图 5.3 是智能家居系统的部分上下文图。其中，智能家居系统软件是机器领域，它将与咖啡机、面包机、家务机器人、门、窗帘、气象系统、警务系统等直接交互，通过医用腕表监视老人的身体情况，通过手机、红外感应器、显示器等感知、通知用户或接受用户指令，通过摄像头拍摄的图片发现是否有陌生人，通过温度/湿度/气敏传感器感知室外或室内空气。这里，医用腕表、手机、红外感应器、显示器、摄像头等起到连接两个领域的作用，又被称为连接领域。

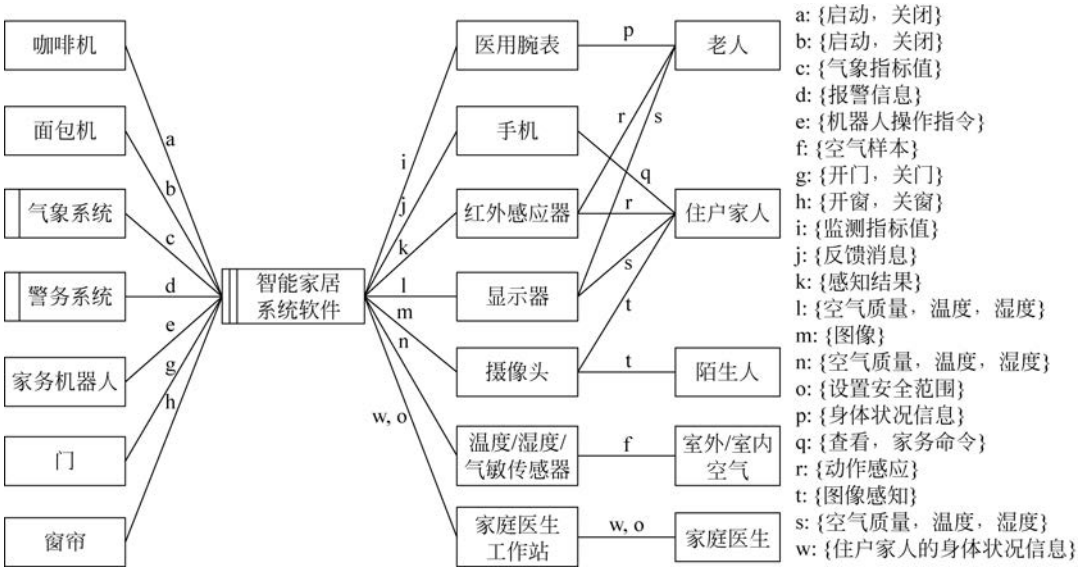


图 5.3　智能家居系统的上下文图（片段）

5.1.2　问题的描述

上下文图主要描述软件开发问题涉及的现实世界的范围，但它只给出了待开发软件所涉及的领域，并没有刻画问题本身（即需求），没有明确到底需要机器领域（软件）做什么。要表达软件需要做什么，需要把上下文图扩展为问题图。图 5.4 给出了智能家居系统软件的一个子问题（即老人监护问题）的问题图。

对照图 5.3 和图 5.4 可以发现，问题图在以下三个方面扩展了上下文图。

第一，形式地表达了领域之间的接口上的共享现象，表示形式为<领域名>！<现象集合>，其含义是现象集合中的现象均由名为<领域名>的领域控制。

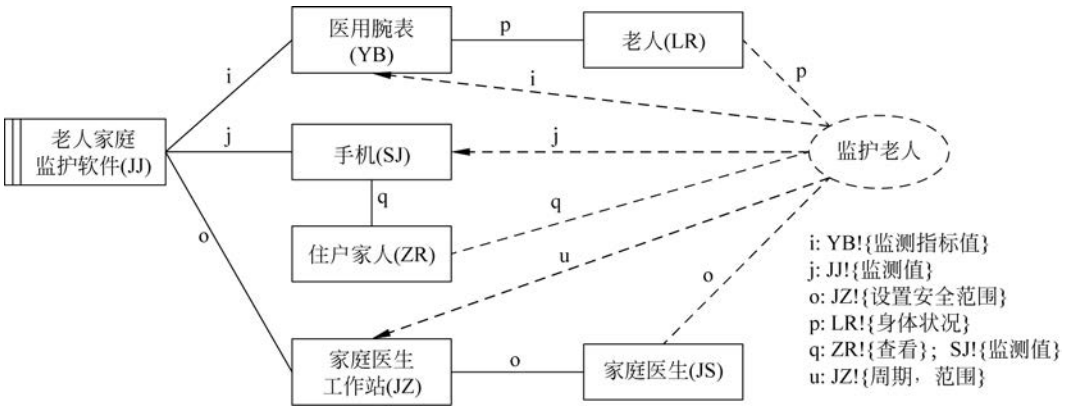


图 5.4　老人监护问题的问题图

第二，增加了需求领域(用虚线椭圆表示)。

第三，增加了从需求领域到问题领域的需求引用(用从需求领域到问题领域的虚线连线表示)。

需求领域会给出希求式陈述，表示在引入机器领域后希望在需求领域所引用的问题领域上看到的效果，这就是对机器领域的需求。在问题图中，带箭头的虚线连线表示约束性引用，指需求领域不是仅仅引用该领域现象，而是表达对该领域现象的期望。例如，图 5.4 中，需求引用“i”要能在医用腕表上获得监测指标值，需求引用“j”表示要能通过手机获得反馈监测值，需求引用“u”表示要能在家庭医生工作站上获得监测的周期和安全范围。

因此，问题图提供了问题分析的依据，提示进行问题分析时要关注的事情，以及必须描述和推断的事情。问题分析主要关注以下几个要点。

- 给出机器领域必须满足的需求：需求是用户希望在引入机器领域之后将在问题领域中发生的事情，或将建立的问题领域现象间的关系，是希求式描述。
- 描述领域特性：领域特性是关于问题领域的事实或是其现象间的关联，不管是否引入机器领域，这些事实和关联都是存在的，是陈述式描述。
- 推断机器领域规格说明：规格说明描述期望在机器领域和问题领域接口上发生的事情，即产生的共享现象及其之间的联系，也是希求式描述。

其中，陈述式领域特性是问题分析的核心，是连接机器领域规格说明和现实世界需求的桥梁。

5.1.3　问题的分解

现实世界问题有时会比较复杂，在软件工程中，分解一直是应对问题复杂性的策略，是控制规模和复杂性的关键。例如，自顶向下的功能分解将功能按层次进行划分，在任意一层上，每个功能都分解为下一层中的多个功能。用例分解则将软件系统看成是要支持一组用例，每个用例表示软件系统与一个或多个参与者之间的交互，也就是将软件系统看成在限定场景中提供离散服务的机制，用例间越独立，则分解越有效。

问题驱动的方法采用问题分解来控制问题复杂度，就是要将大而复杂的问题分解成一组较小且较简单的子问题，其策略是问题投影，目的是从类型未知的复杂问题中识别出类型

已知的子问题。所谓类型已知的问题，指那些已经有规范解决方案的问题，或者知道如何能导出解决方案的问题。当类型未知的问题可以由一组类型已知的问题组合而成时，就能通过组合这些子问题的解决方案，形成整个问题的解决方案。问题类型将在 5.2 节介绍。

根据投影原则，子问题应具有如下特征。

- 完整性：每个子问题都是完整的，有自己的问题图，即包含一个机器领域、一个或多个问题领域，以及一个需求领域。在分析当前子问题时，假设其他子问题的解决方案已经存在，这样可以防止由于要考虑与其他子问题的交互而混淆子问题间的界限，假设其他子问题已经解决是有效的分离关注点的手段。
- 并行性：问题分解不能将问题看作层次结构，子问题应该是并行的。问题划分不能出现如“产品由部件组成，部件由子部件组成，子部件由零件组成”这样的情况。问题划分应该更像是颜色分离，有红色分离、绿色分离和蓝色分离，这三种分离对整幅画来说是叠置的。子问题是整个问题在某个维度上的投影，即：子问题需求是整个需求的投影；子问题机器领域是整个机器领域的投影；子问题的问题领域是整个问题领域的投影；子问题的接口也是整个接口的投影。
- 并发性：由于子问题间的并行性，必须注意子问题间的关系及它们的交互方式，可以理解为，对于两个或多个子问题的机器领域，如果它们的问题领域间有共享现象，则这两个或多个子问题的机器领域的并发执行会引发共享现象的发生。
- 可组合性：子问题的交互非常重要，多个子问题通过交互组合成更复杂的问题。

5.2 问题框架

软件要解决的问题都相同吗？回答显然是“不”。但不同的问题可能会包含相似的子问题，相似的子问题可以归为同一类型，从而可以用相同的关注点去进行问题分析。如果所有的子问题都分析好了，则通过问题的组合就形成对整个问题的分析。

问题驱动的方法通过问题所处的领域、领域特征、接口特征和需求特征来定义问题类，一些可定义的问题类称为问题框架，针对每种问题框架给出分析关注点，从而形成问题驱动方法的分析关注点。本节首先介绍现象及由现象确定的领域特征，然后介绍基本问题框架，最后讨论基本问题框架的几种变体。

5.2.1 现象和领域特征

在明确什么是现象之前，需要区分两个表示原语，即个体和关系。这里，个体指某种可以被命名并且可以明确地与其他个体区分开的东西。问题驱动的方法区分以下三类个体。

- 事件：指在某个特定的时间点上发生或出现的个体，每个时间点都是不可分的且瞬时的，即事件没有中间结构，事件的发生也不需要花时间。因此，“在事件之前”和“在事件之后”是有意义的，但“在事件期间”就没有意义。
- 实体：指持久存在的个体，它可以随时间改变它的特性和/或状态。一些实体在其状态发生变化时还会触发事件。具体的实体可以属于某个实体类，例如，在智能家居系统的老人监护问题中，某个特指的老人就是一个实体，它属于被称为“被监护人”的实体类。

- 值：是无形的个体，它存在于时间和空间之外，是不会发生改变的。一般是用符号表示的数值和字符。例如，智能家居系统的老人监护问题涉及“监测周期”这个值，它表达每两次监测间的最大间隔。

关系指个体之间的某种关联，分为以下三类。

- 状态：状态指实体与一个表示其状态的值之间的关系，可以随时间而变化。例如，在老人监护问题中，“第 123 号模拟设备处于失效状态”表示为 “状态(模拟设备123,失效)”。一个状态可以成立(为真)或不成立(为假)。
- 真值：真值指不随时间发生变化的关系。基本上用于表达数学上的事实，如“等于(|ABCDE|,5)”和“大于或等于(5,3)”。
- 角色：角色指事件和参与事件的个体之间的关系。角色关系有两个作用。其一是区分不同的参与者；其二，将角色的控制与事件的控制分开。

5.1 节曾提到领域间通过共享现象进行交互。在个体和关系的基础上，可以讨论现象的含义。有如下两类现象。

- 因果现象：包括事件、角色和状态等关系。它们由某个领域直接引起，或者被某个领域直接控制，并且能引起其他现象，表现出现象间的因果性。
- 符号现象：包括值、真值和仅与值相关的状态。它们用于符号化其他现象及其之间的关系。

在区分了现象的类型后，可以根据以下几种现象类型对问题领域进行类型刻画。

- 因果型：因果型领域通常是物理的领域，其特征是在它的现象之间存在可预测的因果关系。例如，智能家居系统中的空调是一个因果型领域，其因果性体现为：如果收到“开脉冲”事件，它就会处于“开”状态。
- 顺从型：顺从型领域[①]通常由人或者其他具有自主性的外部系统组成，它是物理存在的，但其内部的因果性对机器领域而言是个黑盒子，与因果型领域相比，缺乏可预测性。例如，在大多数情况下，机器领域可以给人发送指令，让他去做某件事，但他是否会去做或者会按要求去做，都不是可预测的。
- 词法型：词法型领域是数据的物理表示，它涉及的是符号现象，但一般同时涉及因果现象，其因果性表现为数据的写入和读取。例如，数据被写入后就可以被读取，这表现为因果性。

5.2.2 基本问题框架

问题驱动的方法提出：将软件能解决的问题用问题框架进行抽象，并让这些问题框架向上匹配需求，向下关联问题分析关注点，从而指导系统化的需求分析。以下五种典型的基本问题框架分别对应特定的软件问题类型。

- 需求式行为框架：存在着物理世界的某个部分，需要有规律地控制其行为。这个软件问题是：创建一个机器领域来施加所要求的控制。
- 命令式行为框架：存在着物理世界的某个部分，需要按照操作者发出的命令来控制

① 在本章参考文献[1]中，Michael Jackson 给出的英文名称是 biddable domain，要求这类领域遵循机器领域发出的命令，否则有可能产生输入错误而导致机器领域失效，即期望这类领域是顺从的。

其行为。这个软件问题是：创建一个机器领域来接受操作者的命令，并相应地施加所要求的控制。

- 信息显示框架：存在着物理世界的某个部分，关于其状态和行为的信息，需要连续地显示出来。这个软件问题是：创建一个机器领域，不断从物理世界的这个部分获得这些信息，并按规定的形式呈现出来。
- 简单工件框架：需要一个工具，让用户能在上面创建并编辑确定种类的文本、图像或其他结构，供复制、打印、分析或按其他方式使用。这个软件问题是：创建一个机器领域来充当这个工具。
- 变换框架：存在可读的输入文件，其内容需要被转换为另一种确定形式的输出文件，输入/输出文件都有规定的格式，转换也需要遵循给定的规则。这个软件问题是：创建一个机器领域，它能根据输入产生所需要的输出。

问题框架图用来表示问题框架，它在三个方面扩展了问题图。

第一，领域的名字对应于其问题框架的类型，即机器领域分别具体化为控制机器、信息机器、编辑机器和变换机器等，问题领域具体化为受控领域、操作者、现实世界、显示、工件、用户，以及输入、输出等。

第二，每个问题领域附带标记，表示它所属的领域类型，有时一个问题领域具有多个领域类型的特征，可以附带多个领域类型标记。

第三，领域间接口上的共享现象被实例化，表示为形如

<领域名>! <现象集合>[<现象类型>]

的具体现象。其含义是：现象集合中的现象均由名为<领域名>的领域控制，这些现象所属的类型为<现象类型>。下面具体介绍这五种基本问题框架。

1. 需求式行为问题框架

图 5.5 是一个具体的需求式行为问题框架，表示智能家居系统中的空调定时控制问题，即通过一个机器领域(空调控制器)来控制空调(受控领域)，定时设置为早上 8 点关闭，晚上 6 点打开。其中，受控问题领域“空调”右下角的 C 表明这个问题领域是因果领域。C1、C2、C3 分别代表空调控制器和空调之间的三组因果现象。

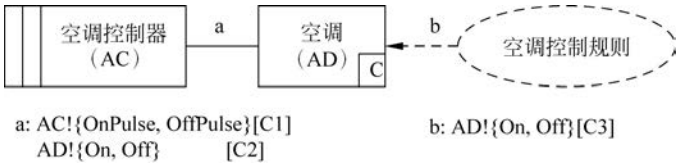


图 5.5 空调定时控制问题的框架图

图 5.6 是需求式行为问题的抽象问题框架图。其中，因果领域和因果现象的表示方式与图 5.5 一致，需要说明以下几点。

- C1、C2 是控制机器和受控领域间的共享因果现象，C1 由控制机器控制，C2 由受控领域控制。控制机器通过现象 C1 来控制受控领域的行为，受控领域用现象 C2 作为反馈。
- 需求由引用受控领域的因果现象 C3 来表达。问题领域和机器领域所共享的现象(C1 和 C2)称为规格说明现象，现象 C3 称为需求现象。

- 需要定义受控因果领域的领域特性，从而建立需求现象 C3 和规格说明现象 C1、C2 之间的联系。

图 5.6　需求式行为问题框架图

2. 命令式行为问题框架

命令式行为问题与需求式行为问题一样，存在一个问题领域需要控制。不同的是，命令式行为问题还有一个操作者，机器领域要根据操作者的命令来控制受控领域，使它产生所需要的行为。例如，图 5.7 表示按照操作者命令进行空调控制的问题框架图。其中机器领域（即空调控制器），根据空调操作者的命令来控制空调。空调操作者是顺从的领域（用 B 作为其领域类型标识），除了因果现象集 C1、C2、C3 外，图中还有空调操作者向空调控制器发出的命令集（Open，Close），被命名为事件集 E4。

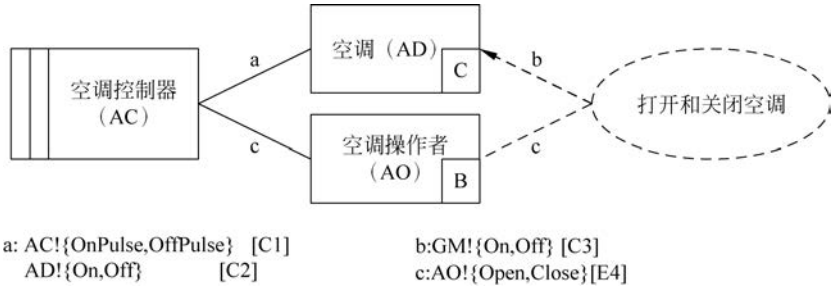


图 5.7　不定时空调控制问题的框架图

图 5.8 为抽象的命令式行为问题的问题框架图。其中，控制机器、受控领域以及它们之间的共享现象都与需求式行为问题一样，但这里多了一个顺从的操作者领域。这个操作者向控制机器发出命令，这些命令被命名为事件集 E4。E4 由操作者和控制机器所共享，并由操作者所控制。

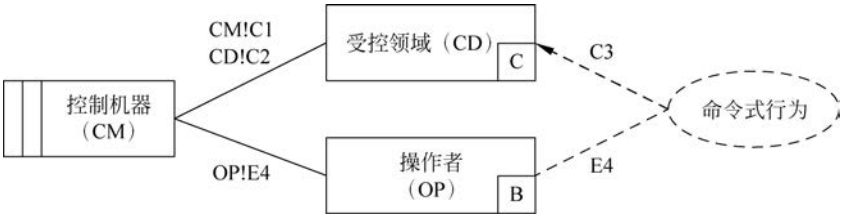


图 5.8　命令式行为问题的框架图

命令式行为问题框架图中的需求称为命令式行为，它描述了所要求行为的通用规则，以及为了响应操作者的命令，受控领域应该如何被控制。对操作者领域来说，其需求现象和规格说明现象不存在差异。因此，事件集 E4 既是操作者领域的需求现象，也是规格说明现象，即与控制机器共享。

操作者领域是自主的，可以在没有任何外界刺激的情况下自主地引发 E4 事件，该问题框架中没有会影响操作者领域的行为。操作者领域控制的 E4 命令是独立的，但 C1 现象则

不同，它必须依赖 E4 事件。操作者领域通过 E4 发送命令给机器领域，而机器领域通过 C1 控制受控领域。问题分析的任务就是决定怎样以及在什么时候，机器领域应该（或不应该）引发 C1 现象作为对 E4 命令的响应。同需求式行为框架一样，命令式行为框架需要知道受控因果领域的领域特性，才能建立需求现象 C3 和规格说明现象 C1、C2 之间的联系。

3. 信息显示问题框架

图 5.9 是一个具体的信息显示问题框架。作为智能家居系统的一部分，环境监测子系统实时监测室内环境，包括室内温度、湿度等，并将监测结果显示在指定的屏幕上。其中，用于监测的系统（即环境监视机器）是信息显示机器，它控制传感器（C1）（包括温度传感器、湿度传感器），感知室内的温湿度指标（C3），获得温湿度值，并根据这些值计算穿衣指数等，最后在指定的显示器上（E2）显示出来。显示器和传感器、空气都是因果领域，其中显示器接受显示命令，并将温度值、湿度值和穿衣指数（符号现象 Y4）显示出来。

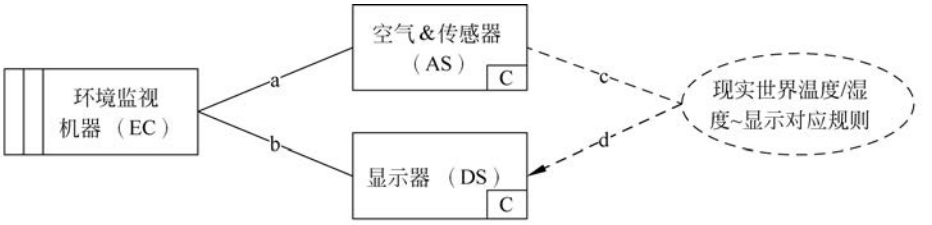


图 5.9　环境监测机器问题的框架图

图 5.10 是信息显示问题的抽象问题框架图。其中，从中获得信息的部分称为现实世界，显示界面指要在这里显示相应的信息。需求称为“现实世界～显示”对应规则，它规定了显示领域的符号现象 Y4 和现实世界的因果现象 C3 之间的对应关系。

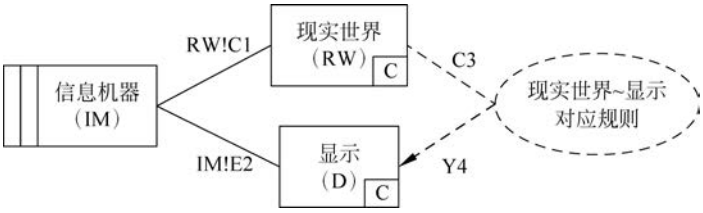


图 5.10　信息显示问题框架图

现实世界领域和显示领域都是因果领域。现实世界领域是主动且完全自治的，它引起事件和状态的瞬间变化，并控制它与机器领域接口上的共享现象，需求没有对它施加任何限制，也没有任何会影响现实世界领域行为的约束，它拥有自身的因果关系。

机器领域根据 C1 现象判断现实世界领域的需求现象 C3，然后机器领域引发事件 E2，引起显示领域的符号值的改变，导致所需的需求现象 Y4，从而满足需求约束。C1 和 C3 之间的联系需要由现实世界领域的因果特性来建立。

4. 简单工件问题框架

工件是指由工具或机器制造出来的制品。软件工具可以用来创建或编辑文本或图形等对象。例如，用字处理软件编辑文档，用画图软件绘制图形等，这里文档和图形都是工件。

案例 5.2　参 数 编 辑

智能家居系统中的控制功能需要设置一些参数，可以维护一个控制参数表，包含要控制的设备、控制参数、时间等内容。例如，空调在 6 点下班之前 10 分钟打开并维持室内温度在 26 度。需要设计控制参数编辑器，用来创建和编辑这个控制参数表。

案例 5.2 中的控制参数表就是工件，需要一个编辑器，用来创建和编辑这个工件。该编辑器的问题框架图如图 5.11 所示。其中，机器领域称为控制参数编辑器(CE)，需求称为“正确的编辑操作”。用户通过编辑命令(E3)来创建或者编辑控制参数列表(Y2，Y4)。

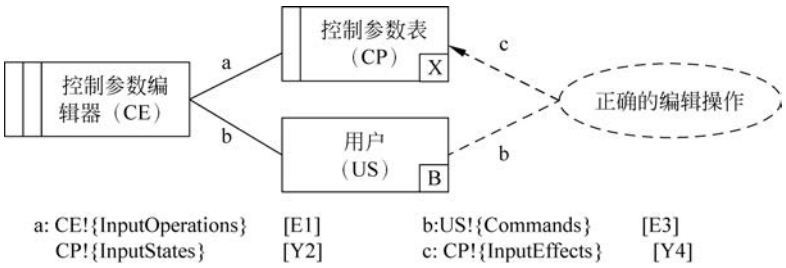


图 5.11　控制参数编辑问题框架图

图 5.12 是简单工件问题的抽象框架图。机器领域称为编辑工具。有一个用户领域(一般是人)，它是一个顺从的领域(用 B 表示)。这个用户是自治的，会在没有任何外界刺激的情况下引发 E3 事件。工件领域是一个词法领域(用 X 表示)，它与机器领域有共享接口，在这个接口上，机器领域控制对工件进行操作的事件现象 E1，这些操作引起工件领域中的符号值和状态的改变；在同一个接口上，工件领域允许机器领域检测工件的当前符号值和状态，即符号现象 Y2。

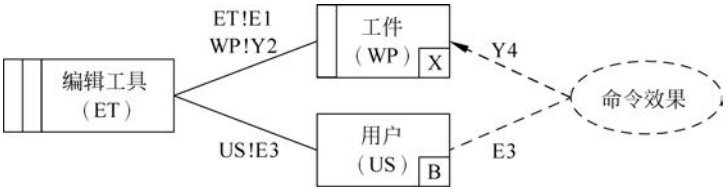


图 5.12　简单工件问题框架图

用户与编辑工具共享事件现象 E3。这个事件现象由用户控制，是用户发给编辑机器的命令。需求称为命令效果，它规定用户发送给编辑机器的命令 E3 应该对这个工件的符号现象 Y4 产生什么效果。现象 Y2 和 Y4 都是工件领域的符号现象，Y2 与 Y4 可以完全不同，也可以部分相同。

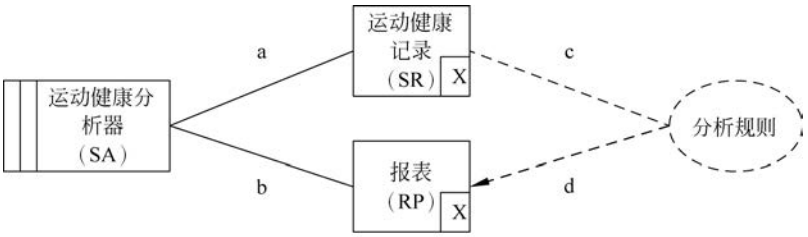
5. 变换问题框架

案例 5.3　运动健康分析问题

智能家居系统中有运动健康监测子系统，这个系统实时监测每个人的运动健康情况，例如，给出每个人的体重、行走公里数、消耗卡路里数、吸收卡路里数等。需要一个运

动健康分析器产生统计报表，显示出每个人的健康数据与运动数据间的关系，如体重和卡路里数之间的关系。

案例 5.3 描述了运动健康分析器问题，它是一个变换问题，其问题框架图如图 5.13 所示。运动健康记录和报表都是词法领域(用 X 表示)，在标记为 a 的领域接口上，运动健康分析器读取运动健康记录领域中的文件及其中的内容，由文件行和每行的字符组成(Y1)。文件包含了通过检测器搜集到的个人健康信息和运动信息，如日期、体重、行走公里数等。在标记为 b 的领域接口上，运动健康分析器确定报表领域的行和字符(Y2)。需求是“分析规则”，给出运动健康记录的结构以及每条记录如何关联到报表的数据上。



a: SR!{FileDir , File, Line, Char} [Y1]　　c: SR!{Date,Duration,Kilometers}[Y3]
b: SA!{ReportLine, Char}　　[Y2]　　d: RP!{LineData} [Y4]

图 5.13　运动健康分析器问题框架图

图 5.14 为变换问题的抽象问题框架图。输入领域和输出领域都是词法的。输入领域是给定的，不能被改变；输出领域由机器领域产生。需求称为“输入/输出”(I/O)关系，它规定了输入领域的符号现象 Y3 和输出领域的符号现象 Y4 之间的关系。机器领域被称为变换机器，它访问输入领域的符号现象 Y1，确定输出领域的符号现象 Y2。Y1 可以与 Y3 相同，也可以不同，Y2 和 Y4 之间也是如此。

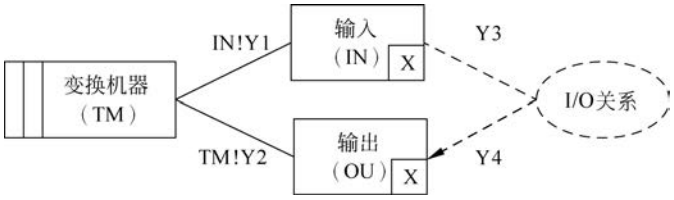


图 5.14　变换问题框架图

5.2.3　基本问题框架变体

5.2.2 节给出了五种基本问题框架，可以将它们看作软件问题的基本模式，这些模式提示了该类软件问题的分析关注点。在实际软件开发时，如果存在与已有问题模式匹配的软件问题，则可以通过该问题模式的关注点的引导，对实际软件开发问题进行分析。

可以通过分析现实问题，提炼并确定尽可能多的问题框架，规约其问题关注点，从而获得更多的问题框架，也可以在已有的问题模式上引入一些变化，产生问题模式的变体。下面介绍基本问题框架的一些常见变体。

(1) 描述变体。

引入描述领域来描述某方面的需求,或者描述可能出现在问题上下文中的某个其他领域,形成问题框架的描述变体。任何基本框架都可以引入描述领域。例如,需求式行为框架引入描述领域,用于显式地表示受控领域的需求式行为,如果需求式行为发生变化,则直接替换该描述领域的内容,控制领域则根据描述领域中给出的行为描述来施加控制。变换框架引入描述领域,用于描述输入领域所包含的符号的含义,变换机器领域则根据该描述领域的符号定义对输入领域实施变换。

(2) 操作者变体。

命令式行为框架就是需求式行为框架的操作者变体,它在需求式行为框架中引入操作者,要求控制机器保证受控领域按照操作者的指令来行动。信息显示框架中引入操作者,构成命令式信息显示框架,这种信息显示框架不像基本信息显示框架那样,在需求中就固定了要显示的信息,而是允许操作者选择要显示的信息,可以认为是机器回答操作者查询的问题。其主要关注点是确定机器能够回答的查询集合。可能的查询受限为:现实世界和机器的接口上直接存在的共享现象;可从领域特性中导出的推理结果(可能带有明确的模型);查询命令语法可以表达的含义。

(3) 连接变体。

基本问题框架假设机器领域与受控领域直接关联。但很多情况下,这个关联需要通过连接领域完成。如果连接领域完全可靠,则可以直接忽略不予考虑。但如果连接领域不可靠,或者可能存在不确定因素的话,就需要专门考虑连接领域,因此出现基本问题框架的连接变体。在这种情况下,机器领域并不直接和问题领域连接,而是通过一个连接领域关联到问题领域,这个问题领域成为机器的远程问题领域。加入了连接领域后,除了需要分析问题领域的特性,还要分析连接领域会如何影响问题领域,这些影响将反映在它与机器领域的共享现象上。

(4) 控制变体。

对事件的控制有三个方面:事件类型的控制(发生哪类事件);事件出现的控制(什么时候发生);事件角色的控制(需要哪些个体参与)。当共享事件的控制发生改变时,则成为相应的控制变体。一般而言,行为框架和信息显示框架没有控制变体,因为它们对共享事件的控制是确定的。但变换框架和工件框架则有可能存在控制变体,如被控的变换框架,其变换方式由输入/输出领域控制。

5.3 框架关注点

基于问题框架的需求分析,通过对问题的分析推导出机器领域的行为描述,产生规格说明。每种问题框架都有特定的框架关注点,这些框架关注点明确了在分析这类需求问题时需要确定的描述,并指导如何将针对不同关注点的描述组合在一起,形成正确的规格说明。本节考察 5.2.2 节中五个基本问题框架的框架关注点。

5.3.1 需求式行为问题关注点

图 5.5 的定时空调控制问题是要找到机器领域的行为规格说明,使它能让问题领域(即

“空调”)表现出控制规则所规定的行为。解决这个问题的框架关注点,就是如何将需求、规格说明和领域特性的描述适当地搭配起来,形成对这个问题的解释。

其关注点如图 5.15 所示,用数字标识的顺序表示应该如何去解释这个问题,好让用户理解所构造的规格说明能满足需求。即:从指定的机器领域行为(1)出发,与给定的问题领域特性(2)组合在一起,则能实现所需要的行为(3)。

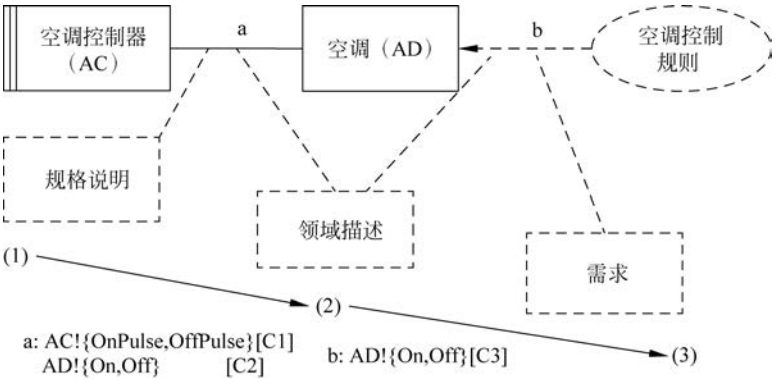


图 5.15 需求式行为问题框架图的关注点

数字标识的逆序还可以启发需求工程师如何获得机器领域规格说明,即从捕获需求(3)开始,研究空调的领域特性,识别机器领域控制问题领域的因果特性(2),然后得出施加控制的机器领域的行为(1)。

5.3.2 命令式行为问题关注点

命令式行为问题框架比需求式行为问题框架多了一个操作者,操作者是完全自主的,可以自由地发布命令。这带来了一定的复杂性,不能假设操作者一定会采用完全正确的方式发布命令,因此在问题分析时,除了要根据需求和问题领域特性归结出机器领域的规格说明之外,还必须分析操作者命令的合理性和可行性。也就是说,需要让机器领域能够对操作者的命令进行选择。例如,在出现下列两种情况之一时需要忽略命令:(a)命令不合理,它们在当前命令上下文中没有意义;(b)命令不可行,它们在受控领域的当前状态下不适当或者不允许。另外,当操作者连续发出的命令对机器领域来说频率太快的时候,还需要考虑不要遗漏有意义的命令。总之,除了需求式行为框架的关注点之外,对命令式行为问题的分析必须考虑命令的合理性和可行性。

图 5.16 展示了不定时空调控制问题的框架关注点。其中,需求(1)规定合理的命令,需求(5)声明如果命令可行的话应该对受控领域带来的作用效果。问题领域特性描述(4)规定,机器领域在共享接口上产生的现象会怎样影响领域状态和行为。规格说明(2 和 3)表明,机器领域对可能出现的命令如何反映,包括如下命令:不合理的命令(2),在问题领域的当前状态下不可行的命令(3),以及既合理又可行,因此要被服从的命令(4)。必须说明所有影响导致的领域状态和行为确实是所需要的(5)。

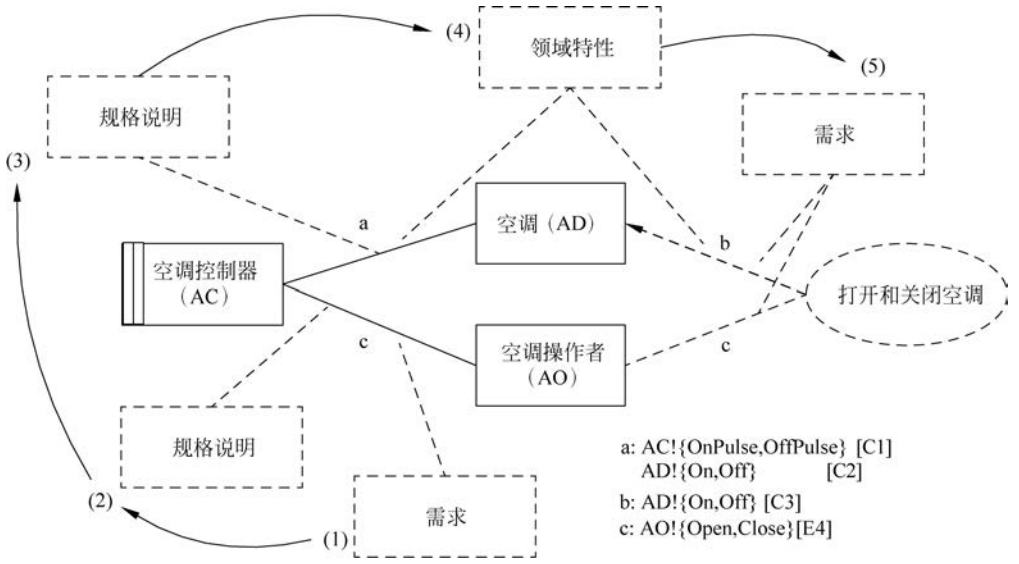


图 5.16　命令式行为问题框架关注点

5.3.3　信息显示问题关注点

信息显示问题需要考虑的是，机器领域和现实世界之间的接口所共享的现象与需求现象之间可能是脱节的，例如图 5.10 中的 C3 现象和 C1 现象之间不能建立联系。以图 5.9 中的环境监测子问题为例，需要机器领域报告空气的不同方面（温度、湿度及穿衣指数等）的情况，但是它只能直接读取两个传感器的读数，并不知道真实的情况。要确定建立了它们之间的联系，必须通过领域特性来解释，即依赖于空气 & 传感器领域的特性。

图 5.17 中，为了解决适合于信息显示问题的框架关注点，必须拥有需求描述、领域特性和机器规格说明。其中，需求(1)涉及现实世界的状态，并规定需要被显示的信息(6)。物理

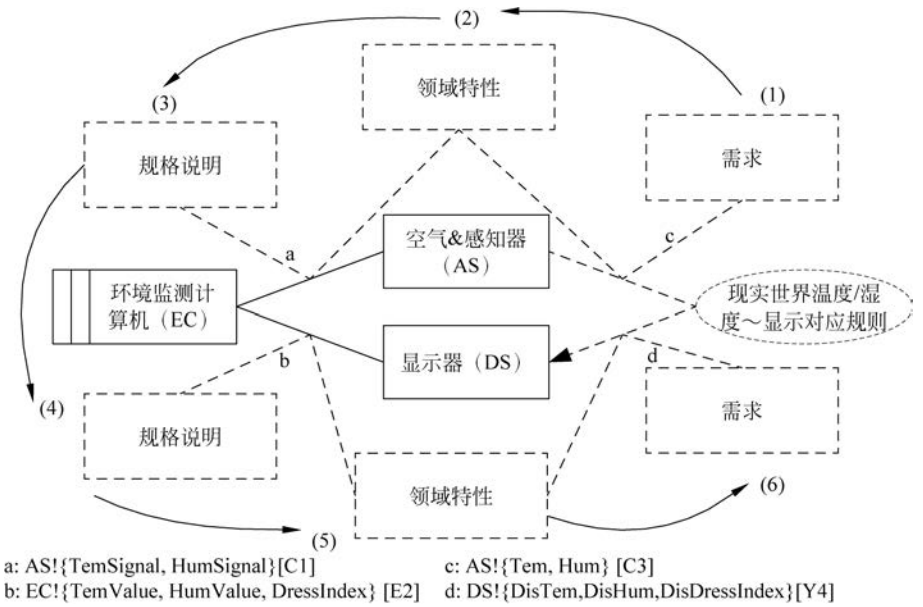


图 5.17　信息显示框架的框架关注点

世界的领域特性描述(2)指出需求(1)中涉及的现象应该如何引发,从而产生能被机器领域直接监测到的现象(3)。机器领域必须像在其规格说明(4)中描述的那样响应这些现象,并且显示领域(5)的领域特性将保证将所需要的信息输出(6)。

信息显示问题中的显示领域比较容易处理,它是设计出来的,如显示器。物理世界领域(如空气 & 传感器领域)相比而言要难一些。问题分析的任务是建立需求现象(要显示的各种信息)和机器接口现象(传感器读数)之间的联系。机器领域的行为必须体现从空气变化引起的传感器读数变化中推断出空气的温度、湿度的规则,以及穿衣指数与空气温度、湿度的关系。

5.3.4 简单工件问题关注点

工件问题中的用户相当于命令式行为问题中的操作者,他们有相同的关注点。机器领域也必须考虑用户命令的上下文,并且拒绝其中的一些命令。图 5.18 是图 5.12 中的控制参数编辑问题的框架关注点,与命令式行为框架的框架关注点有些类似,只是在工件领域的特征上有不同,工件领域支持用户操作和操作效果之间的因果性,但主要是在词法层面,通常是设计的领域。例如,控制参数表领域有数据结构,能够保存值和值之间的关系,有一些能引起这个数据结构中的值和关系发生变化的操作。

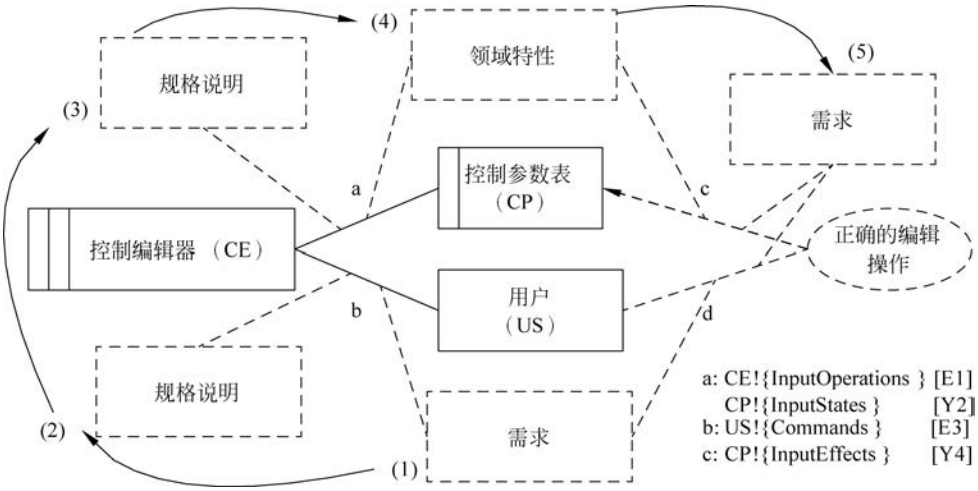


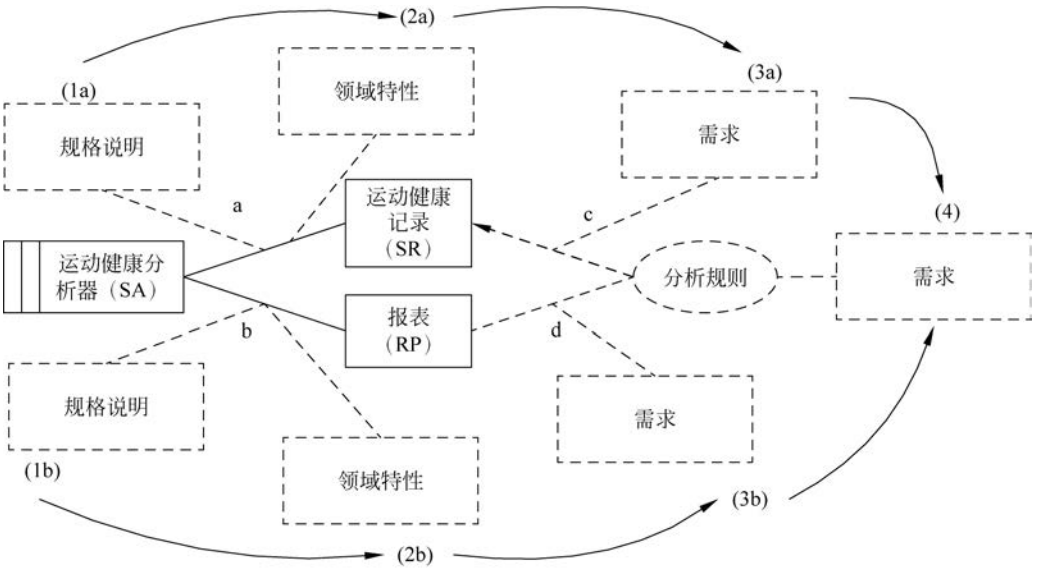
图 5.18 简单工件框架的关注点

另一个方面是次序问题,需要用规定的次序去考虑领域特性和需求。例如,要从控制参数表领域开始,进而考虑用户命令;或者先描述用户命令及它们的作用,然后描述控制参数表领域和输入操作。

5.3.5 变换问题关注点

回忆图 5.14 中的运动健康分析器问题框架图,它是一个变换问题,其输入和输出领域都是词法的。在任何情况下,领域的需求现象与机器接口上的规格说明现象都不同。领域的需求现象和规格说明现象之间的关系是重要的关注点。另一方面,需要理解词法领域输入和输出的纯符号视角和这些符号现象被变换机器存取的因果视角之间的关系。机器领域必须遍历输入和输出领域以满足需求。

图 5.19 是运动健康分析器的关注点，按两条路径(a)和(b)进行，双重遍历嵌入其中的检索和存储符号的操作，必须正确地对应于这两个领域的结构和内容，保证需求中给定的输入/输出关系得到满足。



a: SR!{FileDir , File, Line, Char} [Y1] b: SA!{ReportLine, Char} [Y2]
c: SR!{Date,Duration,Kilometers } [Y3] d: RP!{LineData} [Y4]

图 5.19 变换框架的关注点

5.4 综合关注点

5.3 节给出了五类基础问题框架的需求关注点，除此之外，还有一些独立于问题的关注点，它们不依赖或不局限于特定的问题框架，是软件开发需要考虑的综合关注点。本节简单分析几种典型的综合关注点。

1. 溢出关注点

在系统实际运行过程中，每个领域都在连续地共享现象，在下一个共享事件出现之前，领域能否完成对当前事件的反应，这就是溢出关注点。也就是说，当两个领域在接口上存在速度不匹配的情况时，溢出关注点就会产生。即如果在共享连续现象的两个领域之间，一个领域相对于另一个领域来说反应太快或太慢，则存在产生溢出关注点的可能性。

例如，图 5.12 中的控制参数编辑问题，如果用户输入命令的速度比控制参数编辑器能够处理的速度更快，在用户领域与控制编辑器之间则存在溢出关注点；图 5.9 中的环境监测问题，监测机器可能会以快于显示器的速度来产生信息输出事件，因此在监测机器与显示器之间也可能存在溢出关注点。前一个例子是用户的动作太快，即问题领域相对机器领域来说太快；后一个例子则是由机器领域引起的，是机器领域相对于问题领域太快。

当机器领域太快，可以通过在机器领域中引入延迟而让它的速度慢下来。或者，在机器领域和问题领域之间引入新的共享现象，来判定问题领域是否准备好了下一步的交互。例如，环境监测问题中的显示器可以使用流控制的形式，即机器领域只有在显示器发出准备接

收的信号时才发送数据。

当问题领域对于机器领域来说太快的时候，解决溢出关注点有以下多种策略。

第一种策略是简单阻止，指在机器领域还没有准备好的时候阻止共享事件。这种策略在受控领域是顺从的领域时是合适的，人可以调整自己的动作以避免死锁，如等待或者做别的事情。当问题领域是因果领域的时候，简单阻止就行不通了，因为简单阻止意味着要阻止由因果领域控制的事件，但因果领域的共享事件是根据领域本身的因果性产生的，是不可抗拒的。

第二种策略是忽略，指机器领域忽略那些它没有准备好参与的事件。也就是说，机器领域在还没有准备好的时候，简单地忽略在这期间出现的事件，就相当于假设没有出现这些事件。采取这种策略需要满足下面两个条件：(1)共享事件是由顺从的领域发出的命令；(2)机器领域需要给出明确的提示，表明命令被忽略。

当受控领域是因果领域时，采取这种策略需要考虑忽略这个因果领域共享的事件会带来什么后果。例如图 5.7 中的不定时空调问题，假设空调操作者首先发出 Open 命令，然后是 Close 和 Open 命令，命令的发出非常快以至于机器领域来不及对 Open 命令做出反应，而不得不忽略它。空调操作者知道 Open 命令会被忽略吗？如果不知道，操作者可能会觉得莫名其妙。而如果这个因果领域是另外一个问题的机器领域，并且这个事件被声明为前提条件，即它引发的事件对另一个问题领域具有决定性作用，在这种情形下，采用忽略策略可能会带来无可挽回的错误。

第三种策略是缓冲，即将机器领域没有准备好参与的事件缓存起来，等机器领域准备好了再参与。对缓冲策略而言，当问题领域是顺从的领域时，可能会导致如下情形发生，即一个以前被认为是被动的领域，如事件反应式或状态反应式，表现出其主动式，从而让用户或操作者莫名其妙。特别是在一个命令由于当前的条件不允许执行而被缓冲起来的时候，会出现这样的情形。例如，在一个飞行器控制问题中，“……机器领域发出‘关闭舱门’的命令，这个‘关闭’命令是飞行员在测试飞机状况的时候扳动控制面板上的关闭开关而产生的，而当时由于地勤人员正在对这个门进行维护，机械禁令起了作用，门没有被关闭。两个小时后，维护工作完成，机械禁令解除，机器领域激活这个命令，把舱门关闭。”这个关闭舱门的动作可能出人意料。这里的“阻止”就是一种缓冲，而不是禁止。错就错在容许命令的发生和机器领域响应的行为之间存在延迟。在机器领域执行这个命令的时候，操作者可能早就忘记了这个命令还在等待队列中。

2. 初始化关注点

初始化关注点指机器领域设置其问题上下文的初始状态。初始化关注点分为初始化机器领域和初始化问题领域，一般而言，机器领域可以在任何时间被启动或重新被启动，也就是说，只要操作员或用户愿意，他们就可以在任何时候启动或重启机器领域，而机器领域一旦被启动，就按照规格说明中描述的行为开始运行。需要考虑的是启动或重启机器领域在什么情况下才能发生，没有特别原因，通常假设任何时候都可以发生。

但来自物理世界的问题领域就不那么简单了，它们可能是连续变化的。它们一般在机器领域出现之前就已经存在，它们的状态不由机器领域来设定，而是由其自身的因果性决定。那么，哪个状态可以作为起点呢？这个需要视情况而定。行为问题中的问题领域通常从受机器领域控制的状态中选择一个作为初始状态，但信息显示问题不能这样处理，因为在信息显示问题中，来自物理世界的问题领域是自治的，不受机器领域控制，这种情况下，需要

给机器领域施加约束，将它变得与物理世界的问题领域同步。另外，当机器领域与设计领域存在共享现象时，通常可以初始化这个设计领域。

3. 可靠性关注点

软件开发的目的是构建机器领域，该机器领域利用问题领域描述中的领域特性来满足所指定的需求。

行为式问题中，机器领域利用受控领域的特性，来保证所需要的领域行为。例如，空调控制器依赖空调装置的特性来打开和关闭空调。

信息显示问题中，机器利用现实世界的特性，从它能够直接访问的传感器的状态现象中推断出需求现象。例如，显示机器依赖空气传感器的特性，从传感器中获得空气的温度和湿度。

变换问题中，机器领域利用输入领域的词法特性来分析输入，并将其转换为输出。例如，运动文件分析器依赖目录结构和邮件文件语法，正确地分析消息。

但是问题领域是物理世界的一部分，物理世界中有各种各样的实体，不能假设它们总是可靠的，例如，空调可能坏了，空气传感器也可能坏了。可靠性关注点涉及如何应对并处理这种可能性。当然，可靠性关注点针对问题领域，解决可靠性关注点是为了保证系统能应对某些失效带来的风险，当然，应对措施的实施需要付出代价，这是否值得呢？要回答这个问题，需要对失效的可能性与失效所导致损失的严重程度进行分析。安全关键系统中，失效的后果会非常严重，因此必须要保证系统能应对所有失效，甚至包括不太可能发生的失效，哪怕要付出很大代价。例如图 5.4 中的老人监护问题，即使假设模拟设备是可靠的，也必须诊断并报告可能的失效，因为监测失败会使被监护人的生命处于危险之中。在非安全关键系统中，不会考虑付出太大的代价来提升系统安全性，因为任何策略的引入都是有成本的。

解决可靠性关注点的一种方式是进行问题分解。采用这个策略需要定义一个更精确的问题领域描述，其中包含了所有可能的失效，然后引入一个失效检测子问题去检测这个可能失效的问题领域。这个子问题的需求首先包含检测，也就是说，识别某些失效是否已经出现；它还可能包含诊断，例如，诊断已经检测到的特定错误的原因；它有时还包含修复，例如对词法领域，可以通过修复领域来消除错误。在增加了这个检测子问题之后，原来的问题就可以不考虑问题领域的失效了。

4. 身份关注点

当一个问题领域的多个实例可以独立存在的时候，身份关注点用于识别属于相同问题领域的不同实例。例如，当机器领域与一个问题领域的多个实例之间都存在共享现象，而且这些实例是独立存在的，就需要考虑身份关注点。解决身份关注点的关键在于共享现象的接口，需要区分机器领域是一次只连接到一个实例，还是可以同时连接到多个实例。这两种情况下，这个关注点都要决定，对所共享的现象，机器领域要与哪个实例共享。可以引入一个设计领域来保证正确地识别实例的身份。

5. 完整性关注点

顾名思义，这是为了保证问题需求、问题领域和机器描述是完整的，保证要创建的软件将做用户需要的所有事情。当然，对这个问题谁都不敢打包票，人们基本上做不到仅根据当前的情况就考虑周全未来的需求。需求分析的任务之一就是帮助需求干系人识别出他们所需要的东西。

5.5 小结与讨论

本章介绍的问题驱动方法，通过识别与现实世界的交互去定位问题、进行问题描述和问题分析，它通过问题分解来控制复杂度，根据问题框架（或模式）推断软件设计规格说明。适合于人机交互系统、嵌入式系统、信息物理融合系统等的需求识别、建模与分析。

针对问题驱动方法，有如下三个研究方向值得关注。

- **自动问题分解**：问题驱动方法通过问题分解来控制问题复杂度，它提出了问题投影的概念，对问题投影得到的子问题的性质进行了规约。目前的问题投影机制还是人工的，随着系统规模的增大，需要有系统化的方法支持自动问题投影，例如定义问题投影的维度（根据什么进行投影），问题投影的完整性（如何保证投影得到的子问题一定是完整的子问题），等等。
- **自动问题组合**：问题驱动的方法以问题为驱动，结合现实世界上下文建立机器领域的行为规约。但现实的问题总是由一组以复杂的方式进行交互的子问题组成的，这就引出了组合关注点，基于交互的组合成为采用简单问题框架解决复杂问题的瓶颈。实际上，问题组合和问题分解是解决问题复杂性的两个方面。问题分解从复杂问题出发，识别出可以有效规约的子问题；问题组合是从可以有效规约的问题出发，建立基于交互的聚合，可能产生涌现现象，构建相对复杂的问题规约。
- **自动问题渐变**：在基本的问题框架中并没有考虑远程领域，但在实际的软件项目中，特别是一些复杂问题，都存在着远程领域。例如本章的智能家居例子中，就有一些需要考虑的远程领域。用户需求往往存在于远离机器领域的现实世界中，从用户需求向机器领域靠近，获取机器领域规约的过程称为问题渐变，需要系统化的方法予以支撑（如何进行自动的问题渐变）。

除此之外，问题驱动方法还有其他一些关注点需要深入研究，如非功能需求的描述。需要寻找更多的问题框架以适配更多的现实问题。为了推动其走向成熟，还需要在工业界推广应用，相关工具的研制也是必要的。

5.6 思 考 题

1. 请介绍经典的基本问题框架，并给出其直观含义。
2. 根据问题驱动的方法的原理，需求工程师应该如何去识别问题？
3. 什么是问题驱动的方法中的问题分解手段？
4. 请使用问题驱动的方法对下述问题进行建模。

胰岛素系统是一种胰岛素注射系统。传感器监测患者的血糖数据，并将数据发送到控制器。在接收到传感器数据后，控制器向泵发送喷射命令，泵在接收到命令后设置喷射量。针头注射相应量的胰岛素。

5. 灯光控制系统有两种控制方式，一种是按照操作者的指令进行控制，另一种是定时控制。请画出其问题图，再进行问题分解，最后识别出其子问题所属的问题框架。
6. 请对如下系统进行问题建模。

构建一个智能植物养护系统，它可以实现以下功能：

（1）当人靠近时（通过红外线传感器感知），门会自动打开。

（2）当通过温度传感器感知到温度低于设定值的时候，新风系统换气速度变慢；当通过二氧化碳检测仪检测到二氧化碳浓度过低时，换气速度加快。

（3）当在一天某个时段内光照强度（通过光照传感器感知）达不到要求，就会启动光照系统。

（4）当系统内的湿度（通过湿度传感器感知）低于设定值，灌溉系统就会启动，定期自动施肥的功能也由该系统来完成。此外，还有一个展示系统来展示系统内的相关参数，如温度、湿度和二氧化碳浓度。

7. 请对如下停车场管理系统进行问题建模。

停车场管理系统需要监控车位。在入口处检测到新车进入时，大门开启，车位数量减1；而在车离开停车场时，费用支付之后，大门开启，车离开，车位数量加1；剩余车位数量将进行实时显示。

参 考 文 献

[1] Jackson M. Problem Frames：Analyzing and Structuring Software Development Problems[M]. Reading，MA：Addison-Wesley，2001.

[2] Jin Z. Environment Modeling-Based Requirements Engineering for Software Intensive Systems[M]. San Francisco，CA：Morgan Kaufmann，2018.

[3] Hall Jon G，Rapanotti L，Jackson M. Problem Oriented Software Engineering：Solving the Package Router Control Problem[J]. IEEE Trans. Software Eng.，2008，34(2)：226-241.

[4] Jin Z，Liu L. Towards Automatic Problem Decomposition：An Ontology-based Approach[C]//2nd International Workshop on Advances and Applications of Problem Frames (IWAAPF 2006)，2006：41-48.

[5] 陈小红，尹斌，金芝. 基于问题框架方法的需求建模：一个本体制导的方法[J]. 软件学报，2011，22(2)：177-195.

[6] 陈小红. 结合情景与问题框架的需求捕获和问题投影方法[D]. 北京：中国科学院大学，2010.

[7] Jin Z，Chen X，Zowghi D. Performing Projection in Problem Frames Using Scenarios[C]//APSEC2009：249-256.

[8] Yuan Z，Chen X，Liu J，et al. Simplifying the Formal Verification of Safety Requirements in Zone Controllers Through Problem Frames and Constraint-Based Projection[J]. IEEE Trans. Intell. Transp. Syst，2018，19(11)：3517-3528.

[9] Li Z，Hall Jon G，Rapanotti L. On the Systematic Transformation of Requirements to Specification[J]. Requirements Engineering Journal，19(4)：397-419.

[10] 李智，金芝. 从用户需求到软件规约：一种问题变换的方法[J]. 软件学报，2013，24(5)：961-976.

[11] Yin B，Jin Z. Extending the Problem Frames Approach for Capturing Non-functional Requirements[C]//ACIS-ICIS 2012：432-437.