第3章 面向目标的方法

回忆第1章中的案例1.3，小明思前想后地分析高铁和飞机的利弊。我们不禁要问，像高铁控制软件这样的复杂系统，既有整体目标又涉及很多独立成分，它们的需求是如何获取的？又如何确保获得的需求完整、正确呢？用一般性的需求获取手段，似乎只能获取到一些片段的、可能杂乱无章的需求条目。如何组织这些需求条目，使之形成系统化的、具有整体性的需求模型，以便让像小明这样的系统干系人能从需求模型中的系统功能点出发来判断系统的全局安全性？

面向目标的方法（全称：面向目标的需求分析方法）就是将“目标”看作系统需求的源头和依据，以系统目标为主要线索，探究系统干系人“为什么”要构建软件系统和“为什么”引入特定的需求策略，按照目标分解关系、精化关系、操作化关系等来梳理需求条目，构造软件系统的层次化需求模型，展示软件系统目标达成的完整逻辑思路，并据此构建系统需求规格说明。除此之外，系统目标模型还不仅描述系统需要“做什么”，也分析“为什么这样做”，并按照这个逻辑将不同抽象层次的需求关联起来，提供清晰的层次结构，显式地展示用户的高层需求将如何得到满足，从而有效地指导系统设计决策。

本章将介绍面向目标的方法的基本概念、目标建模的基本要素及基于目标的需求分析过程。

3.1 概　　述

目标的原始含义是射击、攻击或寻求的对象，在其他场景中引申为对活动预期结果的主观设想，是头脑中形成的一种主观意识形态，如活动的预期目标。很多实际软件系统项目都源自关于改变现实社会现状的一个主观预期，希望引入待开发的软件系统来实现这个预期。面向目标的方法就是在这个意义上提出的，它采用目标这个概念来类比软件系统开发的需求。在这种方法中，目标是一个贯穿全局的概念，本节将从不同侧面和不同层次讨论目标这个概念。

3.1.1 目标的分类

面向目标的方法将需求抽象/类比为目标。需求可以分为功能性需求和非功能性需求，因此目标按照其所描述的内容，也可以分为功能性目标和非功能性目标。功能性目标描述对系统行为的需求，表达系统要实现的服务，如“启动列车”“控制列车速度”“保持安全距离”等。功能性目标又可以按照其效果的时序特征分为达成型/停止型目标和维持型/避免型目标。其中，达成型/停止型目标指系统要完成特定行为，最后达成/结束给定的条件状态；维

持型/避免型目标指系统要完成特定行为，从而维持或避免给定条件或者避免某状态。

这种分类实际上确定了目标的语义，表 3.1 给出了目标类型及其语义，并给出了对应的需求示例。

表 3.1　目标类型及其语义

目标类型	含　义	举　例
达成型目标	要求系统最终满足某性质	“启动列车”是一个达成型目标，系统完成“启动”这个行为后，列车要处于“运行”状态
停止型目标	要求系统最终不再满足某性质	“紧急刹车”是一个停止型目标，系统完成这个行为后，列车不再处于“运行”状态
维持型目标	要求系统始终满足某性质	“控制列车速度”可以是一个维持型目标，指需要列车的速度维持在一个确定的范围内
避免型目标	要求系统从不满足某性质	“避免碰撞”则是一个避免型目标，指需要防止列车碰到障碍物

一般情况下，非功能性目标是对整个系统全部功能的约束，例如较高的安全性通常指整个系统都满足安全性约束。非功能性目标可以表达对系统服务质量的约束，如良好的保密性、较高的安全性、较好的易用性等；也可以表达对开发过程质量的期望，如良好的可审核性和可测试性、较高的灵活性等。

功能性目标是独立存在的，但非功能性目标一般只是对系统功能的约束，无法独立存在。有些非功能性目标会随着系统需求的不断精化，逐渐聚焦于具体的系统功能模块，例如系统的登录模块应是安全的。还有一些非功能性目标会在目标精化/操作化的过程中实例化为功能性目标，例如信息的安全性最后会实例化为某些信息加密后传输或者加密存储等功能。关于某些非功能性目标/需求的详细论述请参阅第 7、8、10 章中的相关内容。

3.1.2　目标的层次

面向目标的方法将软件系统需求建模为层次化的目标树，目标的抽象层次取决于其所描述内容，一般分为高层目标和低层目标。高层目标通常是策略性的、粗粒度的、作用于企业或组织范围的抽象目标，主要描述企业或组织的高层业务需求（例如：运送更多旅客，提供随时随地订票的服务）。低层目标通常更靠近技术，粒度较细，是设计层面的具体目标，多数是关于软件系统功能点的需求（例如：发出加速指令，3 次密码输入错误则锁定账户）。

采用面向目标的方法进行需求分析，就是将不同抽象层次的目标组织起来，用“与/或”树的结构表示出目标间的分解和精化关系，形成目标模型的主框架。与/或树的叶节点要控制在多大的粒度上没有明确约定，某个目标是否作为叶节点依赖于其是否需要继续分解或精化。主观判断的依据是：该目标是否能明确分配给系统的某个相关主体，由它负责达成。

3.1.3　目标的作用

目标模型在需求的获取、审查和验证，甚至在系统设计的决策中都有重要的作用，其主要作用有以下几点。

（1）**指导需求获取过程**：为需求获取提供明确的线索和思路。面向目标的方法关注目标间的精化关系，目标的精化为需求获取提供了一种自然的分层机制，目标精化过程支持发

现具体需求。

(2) **保证需求完备性**：为需求完备性提供充分和精确的判定依据。如果根据已知的领域性质(E)和规格说明(S)能够证明目标集合(G[①])中的所有目标均可满足，则该规格说明对于给定的目标集合来说是完备的。即，如果

$$E, S \models G$$

则可以推断出 S 相对于 G 是完备的。

(3) **避免引入无关需求**：为需求的相关性提供充分和精确的判定依据。如果根据已知的领域性质(E)和规格说明(S)能够证明目标集合(G)中的至少一个目标是可满足的，则该规格说明对于给定的目标集合来说是相关的。即，如果

$$\forall s \in S, \quad \exists g \in G, \text{使得 } E, s \models g$$

则可以推断出 S 中不包含相对于 G 的无关需求。

(4) **需求解释与追踪**：提供从高层策略目标到具体实现技术的追踪链。当某个具体需求的原因不明时，可以通过跟踪链进行回溯，找到引入该需求的高层目标，两者间的目标精化路径可以解释该需求引入的原因。

(5) **候选设计方案分析**：支持候选解决方案的对比分析。目标的“或”分解可以表达达成目标的不同方式，经过多级“与/或”分解可以得到满足高层目标的不同组合。冲突的目标可以通过选择不同候选方案来消除。对比分析不同候选方案的优缺点，可以支持设计决策。本书第 4 章有设计决策的详细介绍。

(6) **系统的全局分析视角**：支持需求和组织业务场景相关联。具体体现为：最低层目标到系统相关主体的分配关系，表明目标的满足是由系统及其相关主体共同参与完成的。例如，铁路运输系统的安全目标是由火车司机、轨道管理系统、车站管理系统、通信设备、乘客等共同参与完成的，ATM 系统保持用户合法性的目标是由 ATM 控制软件、感应器、效应器、用户等共同协作完成的。

(7) **非功能性需求分析**：非功能性需求是需求分析的难点。面向目标的方法用软目标表达非功能性需求，支持软目标的分解，实现非功能需求的操作化，以及不同非功能需求间的冲突决策。本书第 8 章有关于软目标的详细介绍。

(8) **支持需求的演化**：高层目标的相对稳定性有利于支持需求的演化。目标分解树成为自然的需求演化管理模型，即越高层的目标越稳定。不同版本的系统通常具有相同的高层目标。

3.2 目标建模元素

在面向目标的方法中，需求建模的第一步是建立系统目标模型，然后以目标模型为核心，结合对象模型、主体模型、障碍模型、操作模型和行为模型，形成多个相互补充的视图，建立对软件系统需求的全面描述，完整表达系统需要什么、为什么需要，谁负责完成等。具体地，其需求分析框架涵盖以下视图。

- 意图视图：描述系统的功能性和非功能性目标的目标模型。

① 对应第 1 章中的需求三元式，这里的目标是具体化的需求。

- 结构视图：描述系统所操纵的概念对象、它们的结构及其相互关系的对象模型。
- 责任视图：描述构成系统的主体及其与系统目标间的责任关系的主体模型。
- 功能视图：描述系统为实现其功能性目标而执行的操作和提供的服务的操作模型。
- 行为视图：描述主体在特定交互场景中的预期行为的行为模型。

图 3.1 以列车控制系统的部分需求模型为例，具体给出了该需求分析框架中各类视图的建模示意以及它们之间的关系。本节将基于这个需求分析框架，逐一介绍面向目标的方法所涉及概念的含义及其图形化表示方式。

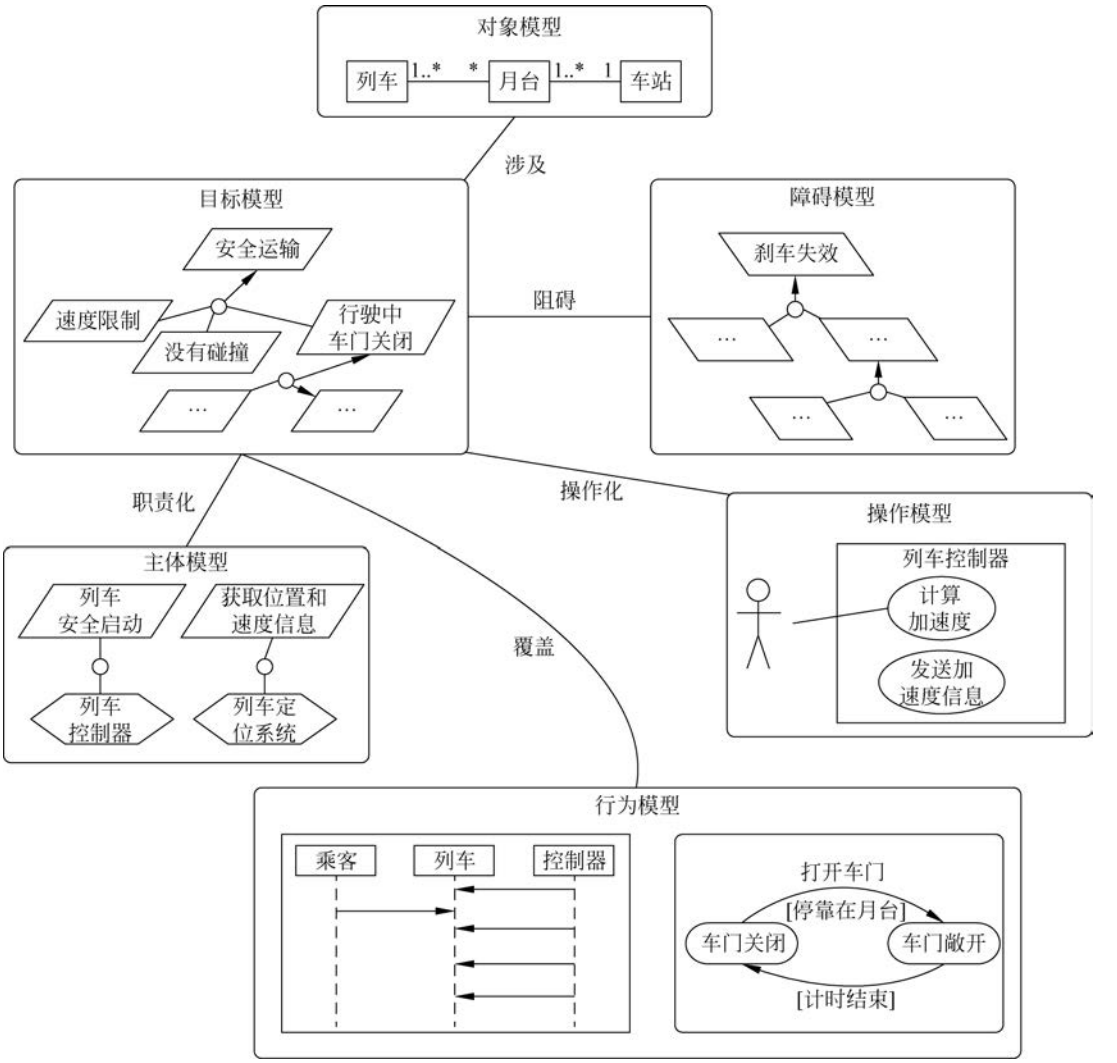

图 3.1　面向目标的方法中需求分析框架总览

3.2.1　目标的表示

目标模型中，每个目标都由一组属性来刻画，其中，目标名与目标定义是任何目标都必须有的，是必要属性，其他属性则是可选的，可以根据目前掌握的信息有选择地给出。主要有如下属性。

- **目标名**：每个目标都有唯一的名称，以保证该目标在整个模型（包含不同视图模型）中能被唯一标识。
- **目标定义**：在系统相关概念和术语的基础上，用自然语言准确表述的目标定义。
- **行为类型**：目标所描述的系统行为可以按照其时序特性分为达成型、停止型、维持型和避免型四种，其含义见表 3.1。
- **目标分类**：描述目标所属的类别，包括功能性目标和非功能性目标。其中非功能性目标又可以根据所约束的质量特征进一步划分，例如分为安全性目标、性能目标、可用性目标等。
- **目标来源**：记录目标的来源，包括提出该目标的干系人、背景研究初步文档中的特定部分、领域标准和法规等。
- **目标优先级**：定性地给出目标重要性的分级，以便在进行冲突/竞争目标的决策选择时可以进行比较。例如，在列车控制系统等安全关键系统中，与列车的性能等目标相比，安全性目标的优先级往往是最高的。
- **形式规格说明**：用时序逻辑对目标的内容进行形式化描述，以支持对目标模型的充分性、一致性和完整性的验证。这样的验证对安全关键系统尤为重要。

目标模型常常用图形化形式给出。图 3.2 给出了“保持安全距离”这个目标的图形化表示，其中，目标名称由带目标名标记的平行四边形表示（▱）。如果声明了目标类型，则将目标类型作为目标名称的前缀。在这个例子中，目标类型为维持型，关键字为保持（Maintain），名称为“保持安全距离”。

图 3.2　列车系统中的一个目标描述示例

3.2.2　目标的精化

精化关系是目标模型中的重要关系，描述了如何将一个高层的、抽象的、粗略的目标精化为一组低层的、具体的和精化的子目标。通过精化获得的子目标间可以是“与”关系，也可以是“或”关系。

“与精化”关系用带圆点的箭头表示，其中圆点上方的箭头指向父目标，而圆点下方与每个子目标相连。子目标之间是“与”关系时，所有子目标被满足，父目标才满足。图 3.3 中，从目标“安全运输”到“灾难疏散”“保持车门关闭”“避免[列车相撞]”三个子目标的精化是“与”关系，即为了使“安全运输”这个目标得到满足，其三个子目标“灾难疏散”“保持车门关闭”和“避免[列车相撞]”必须同时满足。

当目标存在多种可选的精化方案，这些精化方案之间构成“或”关系时，用“或精化”关系表示，表示为一个独立的带圆点箭头。子目标之间是“或”关系时，只要有一个子目标被满足，父目标就被满足。图 3.3 中，目标“避免[列车相撞]”有两种可选的精化方案：一是避免同一区域出现多辆列车，二是保持列车之间的安全距离。

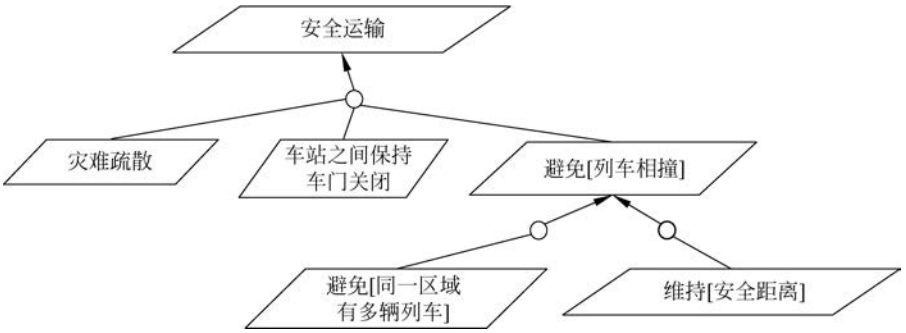

图 3.3　目标精化示例

3.2.3　其他概念

如图 3.1 所示，在面向目标的方法中，除了目标概念，还包含障碍、主体、情景、对象、操作等概念，它们也具有相应的分析模型（如对象模型、操作模型等），这些概念的基本含义及其与目标模型的关系如下。

（1）障碍（Obstacle）。

障碍是使系统目标无法满足的条件。针对特定的领域性质（E），障碍（O）可使得目标（G）无法被满足，即：$E, O \models \neg G$。其中，障碍与其所影响的目标之间存在阻碍关系。障碍在基于目标的系统风险分析中具有重要作用，与目标的精化分析类似，通过对障碍进行与/或精化分析可以创建障碍图，用于识别细粒度的且能够被有效处理的障碍。根据障碍所阻碍的目标类型，障碍也分为不同类型。例如，阻碍安全性（Safety）目标的障碍是灾害（Hazard），阻碍安保性（Security）目标的障碍是威胁（Threat），等等。考虑到障碍起到的是对目标取“反”的结果，其图形化表示为反平行四边形（⏥）。

（2）主体（Agent）。

主体是能够自主执行操作的系统组件，可以是人、软件或硬件设备等。在面向目标的方法中，当叶子目标被操作化为具体的操作后，需要指定该操作的执行者，即主体。主体与目标之间是职责关系，即接受委派后的主体有责任在适当的时候执行适当的操作，来满足特定的目标。主体通过能力（Capability）和执行（Perform）两个原语与操作目标相关联，前者表示主体有能力执行的操作，后者表示主体负责执行的操作。主体有一个特殊的属性 Load，表示该主体当前的负载率，用于对主体进行负载分析，防止主体出现任务过载的情况。主体的图形化表示为六角形（⬡）。

（3）情景（Scenario）。

情景是由相应主体控制的、领域相容的状态迁移序列。这里的领域相容性（Domain-consistency）是指当一个操作的领域前置条件满足且该操作相关对象的领域条件满足时，运用该操作所导致的后置条件将满足领域后置条件。情景用于描述主体的特定交互行为。关

于情景分析的内容将在本书第 6 章详细介绍。

(4) 对象(Object)。

对象是系统需求中所关注的事物。对象的实例可以从一个状态演化到另一个状态。对象实例在某一时刻的状态定义为：该实例的所有属性在该时刻的取值。对象包含三种类型：实体(Entity)是自主的对象；关系(Relationship)是将对象联系起来的特殊对象；事件(Event)是瞬时存在的对象。通过构建对象模型，能够结构化地对系统相关概念进行描述。目标、障碍、操作、主体、行为等概念都需要基于特定对象进行定义，其与对象模型之间为涉及(concern)关系。

(5) 操作(Operation)。

操作是系统状态之间的二元关系。操作通过 Input、Output 原语与对象相关联，能够改变对象的状态，进而实现系统状态之间的迁移。例如，将“开门”这一操作应用于初始状态为“关闭”的“列车车门”对象上，能够将其状态变为“打开”。操作可以表示为输入系统状态集合和输出系统状态集合之间的二元关系，即：操作⊆输入状态集合×输出状态集合。对目标模型进行精化分析所得到的叶子目标需要被操作化为特定的操作。具体地，需要根据叶子目标中所期望的系统状态来找到适当的操作进行满足，即叶子目标期望状态集合⊆操作输出状态集合。

(6) 领域属性(Domain property)。

领域属性是对环境中对象和操作客观性质的描述，不依赖软件系统而存在，也不会因为软件系统的行为而发生改变。领域属性包括物理法则政策法规以及环境实体可能会对系统施加的约束等，如列车移动时列车的速度不能为零。领域属性用对象不变式和操作的前/后置条件来表达，如：列车移动 ↔列车速度 ≠ 0。其图形化表示为五边形(⬠)。

3.3 基于目标的需求分析

面向目标的方法以目标模型构建为核心，结合其他几种模型，能够系统地从不同角度进行需求分析。粗略地说，其需求分析过程就是首先抽取企业或组织期望满足的目标，然后通过引入目标实现策略，逐步分析得出如何通过软件系统实现这些目标，最后形成软件需求规格说明。在抽取和定义目标的过程中，还要考虑哪些因素可能会妨碍目标的实现，并进行障碍分析，挖掘系统的安全需求。在需求分析过程中，目标的采集应尽可能照顾到多方来源，并保证目标的精确性。将每个目标与相关的干系人联系起来，以涵盖不同的视角，也便于消解冲突。

具体地，面向目标的需求获取、分析与模型构建包括以下主要步骤：

(1) 获取顶层目标，并通过反复迭代的目标精化过程构建目标模型；

(2) 针对所构建的目标模型进行障碍分析，构建障碍模型；

(3) 确定目标所关注的、需要对其施加操作的实体对象，构建对象模型；

(4) 确定系统的相关主体，进行各种候选方案的主体职责分配，构建主体模型；

(5) 基于目标模型和主体模型，构建所有可选的软件系统需求规格说明，并从中选择最优方案，将最优方案的目标模型中的叶子目标操作化为具体的系统功能约束，构建操作模型。

上述步骤通常按顺序执行，但没有严格的顺序要求，例如，步骤(2)至步骤(4)可以并发执行。这些步骤一般都可回溯，例如，步骤(2)可能会导出新的安全目标，这就需要回溯到步骤(1)对新目标进行精化分析。

下面详细介绍上述分析步骤。

3.3.1 目标建模

构建完整且无歧义的目标模型是面向目标方法的核心环节。目标模型构建一般采用自顶向下和自底向上相结合的方式。

首先，通过对系统场景的调研，获取初始目标集合。该目标集合应该包括企业或组织的战略目标、领域相关目标以及由于当前系统存在的问题所引出的目标。

接下来，针对初始目标集合中的每一个目标，通过反复询问“为什么要”和“如何做”这两个问题，尽量找全各自的父目标和子目标。例如，针对图 3.3 中“避免[列车相撞]”这一目标，分别询问“为什么要”和“如何做”问题，可以向上得到其父目标“安全运输”，向下发现两个可相互替代的子目标“避免[同一区域有多辆列车]”和“维持[安全距离]”。

针对现有目标集合中的某个目标及其父目标，围绕该目标的精化过程如下。

(1) 仅满足当前目标是否能够保证其父目标被满足？如果不能，则需要找出与当前目标同时满足才能保证其父目标被满足的其他兄弟目标。例如，“避免[列车相撞]”的目的是为了保证“安全运输”，但“避免[列车相撞]”只是实现“安全运输”的必要条件，而非充分条件。通过询问满足“安全运输”的充要条件，识别出“避免[列车相撞]”的两个兄弟目标，即“灾难疏散”和“车站之间保持车门关闭”，这三个目标与保证“安全运输”之间构成“与精化”关系。

(2) 是否存在其他可替代的方式来满足该父目标？例如，通过问“为什么要”获取“维持[安全距离]”的父目标“避免[列车相撞]”后，再考虑其他可以满足“避免[列车相撞]”的途径，即得到另一个子目标“避免[同一区域有多辆列车]”，这两个目标与“避免[列车相撞]”间构成“或精化”关系。

目标精化是一个迭代过程，通过“为什么要”和“如何做”这两个启发式问题进行目标分析，能够将特定目标精化为更细粒度的子目标。最终得到的叶子目标的粒度没有固定的标准，但可以给叶子目标的粒度赋予一个可操作的定义，即 “能分配给一个具体的主体，该主体通过执行特定的操作可以满足这一目标”。

不同目标有不同的精化方式，在一定的抽象层次上，能够发掘出一些通用的目标精化策略。将这些精化策略封装为精化模式，可以实现对目标精化知识的有效复用。具体地，常用的目标精化模式有以下三种。

(1) **里程碑驱动的精化模式**：找出目标条件实现过程中必备的中间条件(即里程碑)，按里程碑满足的先后次序划分子目标。

(2) **案例驱动的精化模式**：通过引入案例相关的特定条件来指导目标精化。典型的案例驱动精化模式有以下三种。

- 案例分解模式：当目标条件的实现需要分情况讨论时(如正常案例和异常案例)，则根据不同的案例情况对父目标进行精化。
- 守卫引入模式：目标条件的达成须保证特定条件始终被满足，引入该条件作为守卫条件进行目标精化。

- 分治模式：对于维持型目标，通过对其维持的目标条件进行分解，可以实现对该目标的精化。

(3) **可实现驱动的精化模式**：目标通过迭代的精化，最终应能够被指派给特定主体来实现。因此，当目标不能够被主体实现时，则应当将其目标条件进一步精化为可被实现的条件。

通常，精化模式包含模式名称、应用条件、目标精化树、应用实例和可能的模式变体。图 3.4 展示了里程碑驱动精化模式的目标精化树(图 3.4(a))及相应的应用实例(图 3.4(b))。模式的应用就是将模式中的核心概念实例化为问题领域中的相应概念。在图 3.4(b)所示的例子中，要达成“乘客下车”的目标，必须满足的一个里程碑条件是“列车停车”。

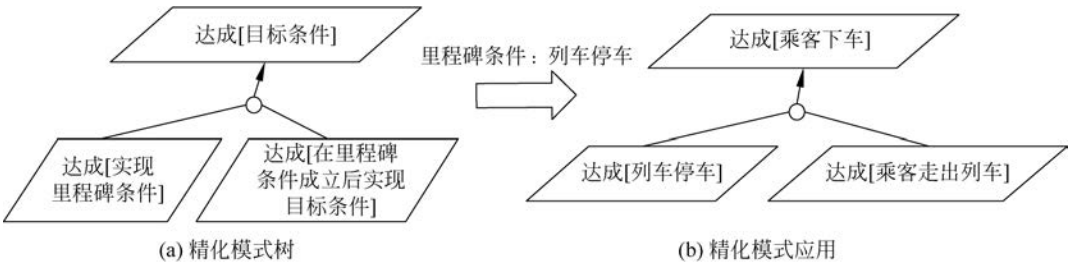

图 3.4　里程碑驱动的精化模式

采用精化模式进行目标分解的规则包括：

- 如果一个目标能够与精化模式的父目标相匹配，则可以对该目标进行精化分解；
- 如果一个目标能够与多组精化模式的父目标相匹配，则基于这些目标模式的精化就构成了对该目标的一组“或精化”关系；
- 如果一组目标能够与精化模式的子目标相匹配，则可以自底向上识别出这组目标的父目标；
- 如果一个目标及其子目标能够与精化模式中的父目标和部分子目标相匹配，则应当基于该模式补全相应的子目标。

3.3.2　目标障碍分析

目标障碍分析用于发现系统有哪些可能的异常行为会阻碍系统目标的满足，即在目标层次上分析系统的异常行为，并将其作为需求描述的组成部分。在面向目标的方法中，目标通常被定义为一组期望行为的集合，而每个行为都对应于一个状态迁移序列，在相应的主体控制下完成预期的操作。相反地，障碍则定义了一系列不希望发生的行为，是系统要避免的场景。障碍的逆运算成为系统目标满足的前提条件。

假定 G 为目标，Dom 为领域性质集合，当且仅当以下条件满足时，称命题 O 为目标 G 在领域 Dom 中的障碍。

Ⅰ. $\{O, \text{Dom}\} \models \neg G$（阻碍关系，obstruction）

Ⅱ. $\{O, \text{Dom}\} \not\models \text{false}$（领域相容性，domain-consistency）

Ⅲ. O 在当前场景中可以被满足(障碍的可行性，obstacle feasibility)

上述条件Ⅰ表明，障碍规约和领域性质所构成的模型不能满足目标 G。条件Ⅱ表明，障碍规约必须与领域性质在逻辑上相容。也就是说，分析与领域性质不相容的障碍没有意义。

条件Ⅲ表明，障碍必须是可满足的，即存在一个行为能够使该障碍存在的前置条件成立。例如，对于列车控制系统中的目标“列车收到刹车命令后制动”，与其相关的领域性质是“如果列车驾驶员响应刹车命令，则列车会进行制动”。针对以上的目标和领域性质，如果出现“列车员未响应命令”这一情况，则会导致列车在收到刹车命令后无法制动（满足条件Ⅰ）；此外，该情况与领域性质不存在冲突（满足条件Ⅱ）；最后，该情况有可能在特定场景下被特定行为满足，例如列车员睡着了或在与其他人交谈（满足条件Ⅲ）。因此，“列车员未响应命令”是妨碍上述目标满足的障碍。

在障碍分析过程中，需要尽可能地找全每个目标的障碍，以保证需求的完备性，尤其是与安全性有关的目标。当且仅当以下条件满足时，目标 G 的障碍集 $O_1,\cdots,O_n$ 称作是完备的：

$$\{\neg O_1,\cdots,\neg O_n,\mathrm{Dom}\}\models G\text{（领域完备性，domain-completeness）}$$

即障碍集中的所有障碍均不发生时，目标将被满足。需要注意的是，障碍分析与系统领域性质分析是相辅相成的。

一方面，障碍分析的完备性依赖于对领域性质的理解程度。对领域性质了解越深入，则越容易识别出潜在的障碍。例如，基于领域性质“**如果**（**IF**）列车驾驶员响应刹车命令，**则**（**Then**）列车会进行制动”进行障碍分析，对 **IF** 引导的命题取反，可以提示“列车驾驶员未响应命令”这一障碍。

另一方面，障碍分析有助于对领域性质的理解和验证。例如，针对列车制动目标的障碍分析，可提示并识别出另一个障碍“列车刹车系统失灵”，从而加深对已知领域性质的理解，提示修改上述领域性质为“**如果**（**IF**）列车驾驶员响应刹车命令，**并且**（**AND**）刹车系统正常，**则**（**Then**）列车会进行制动”。

障碍之间也存在“与/或精化”关系，其语义与目标精化关系一致。如图 3.5 的右部所示，“列车员未响应命令”和“列车刹车系统失灵”两个障碍是对“列车收到刹车命令后没有制动”这一障碍的“或精化”，即发生两个障碍中任意一个都可能引起它们的父障碍，并最终阻

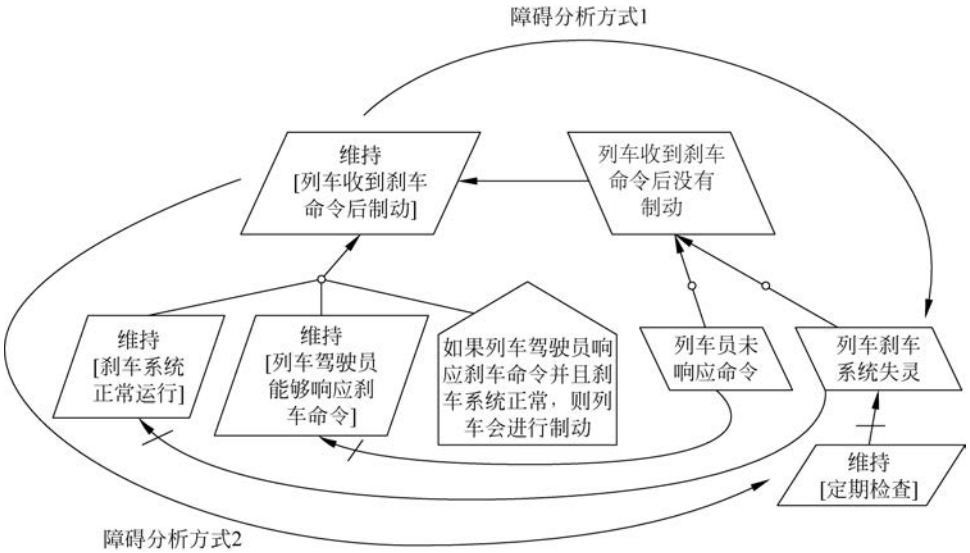

图 3.5　障碍模型及其分析方式示例

碍系统目标的实现。对障碍进行精化分析的目的，是尽可能准确地判断障碍的影响，以便选择和采取适当的应对措施。例如，图 3.5 中针对“列车刹车系统失灵”问题，引入“定期检查刹车系统”作为对该障碍的应对措施，该措施最终将整合到系统目标模型中。

障碍精化虽然和目标精化在形式上有相似之处，但目标描述的是系统希望出现的情况，而障碍则描述系统不希望出现的情况。目标精化分析更符合人们常规的思维方式，其分析难度要低于障碍精化分析。如图 3.5 所示，第一种障碍分析方式需要对根障碍进行精化分析，得到具体的子障碍，分析人员需要具有较强的反向思维能力，难度相对较大；而第二种分析方式先将系统目标精化分解为更细粒度的子目标，然后再分析每个子目标的障碍，难度相对较低。

从分析过程的角度看，障碍分析与风险分析的过程类似，包含以下步骤：

(1) 针对特定的系统目标，识别相应的障碍；

(2) 评估障碍产生的可能性以及障碍所造成损失的严重程度；

(3) 分析和应用适当的障碍消解方法，并相应地修正目标模型。

本节接下来将详细说明上述每一个步骤。

1. 障碍识别

图 3.6 展示了障碍识别的分析流程。在确定待分析的目标后，首先对目标取反，获得直接阻碍该目标实现的障碍，将该障碍作为根障碍，迭代地进行障碍精化，直到能准确地对障碍的产生概率和损失严重程度进行评估。

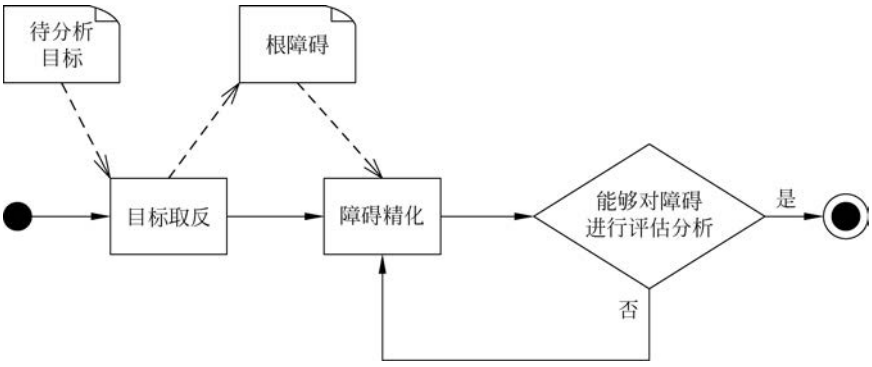

图 3.6　障碍识别流程

考虑到实际需求分析中难以给出目标的形式化表示，因此目标取反所得到的障碍往往也只是以自然语言的形式表示。相应地，障碍精化分析主要利用形如“如果目标规约具有 X 性质，则考虑以下类型的障碍”的启发式规则。具体地，对“监视或控制某对象”这类目标，要保证它的可满足性，可能有以下几种类型的障碍。

(1) 信息不可用：主体无法得到关于该对象状态的信息。

(2) 信息不及时：主体得到关于该对象状态的信息过迟。

(3) 信息错误：主体保存的关于该对象状态的信息与实际状态不符。信息错误障碍又可进一步精化为以下几种。

① 信息提供错误：来自另一个主体的信息是错误的。

② 信息被破坏：来自另一个主体的对象状态信息被第三方破坏。

③ 信息过时：来自另一个主体的对象状态信息是过时的。

④ 信息丢失：来自另一个主体的对象状态信息在使用时不可用。

⑤ 推断错误：主体根据现有信息做出了错误的推断。

⑥ 信息混淆：主体将对象状态信息与其他信息混淆。信息混淆障碍又可进一步精化为以下几种。

- 实例混淆：将某个对象实例与同类对象的另一个实例混淆；
- 值混淆：将同一对象某个属性的不同值混淆；
- 单位混淆：将同一对象属性值的单位混淆。

依赖启发式规则的障碍识别与精化分析需要较强的领域知识背景。通过对障碍进行半形式化表示，能够有效支持障碍精化分析。具体地，使用逻辑运算符半形式化地进行障碍描述，例如，“**not**(**if** 列车驾驶员响应刹车命令 **and** 刹车系统正常，**then** 列车会进行制动)”。利用重言式可以更加便捷地进行障碍精化分析，如下：

- $\text{not}(A \wedge B) \Leftrightarrow \text{not } A \vee \text{not } B$
- $\text{not}(A \vee B) \Leftrightarrow \text{not } A \wedge \text{not } B$
- $\text{not}(\text{if } A \text{ then } B) \Leftrightarrow A \wedge \text{not } B$
- $\text{not}(A \text{ iff } B) \Leftrightarrow (A \wedge \text{not } B) \vee (\text{not } A \wedge B)$

根据这些重言式，如果障碍描述的模式与某重言式左部相匹配，则可以根据该重言式对该障碍进行与或精化。例如，“**not**(**if** 列车驾驶员响应刹车命令 **and** 刹车系统正常，**then** 列车会进行制动)”是形如“**not**(**if**(A **and** B)**then** C)”的障碍。利用重言式可以获得其等价的逻辑表示“A **and** B **and not** C”，即“列车驾驶员响应刹车命令 **and** 刹车系统正常 **and not** 列车会进行制动”。

2. 障碍评估

在识别出障碍并进行充分的精化分析后，针对障碍树的每个叶子障碍，需要评估其发生概率及其造成影响的严重程度。障碍评估一般使用风险分析技术，并需要领域专家提供必要的领域知识，特别是对障碍发生概率的评估。

在完成对每个叶子障碍的评估后，可以基于障碍树结构，自底向上地完成对根障碍发生概率的评估。如果一个障碍被“与精化”为多个子障碍，则该障碍的发生概率为其所有子障碍发生概率中的最小值；而如果一个障碍被“或精化”为多个子障碍，则该障碍的发生概率为其所有子障碍发生概率中的最大值。

(1) AND-refinement $(O, \text{SO}_1, \cdots, \text{SO}_n)$

$$\text{Likelihood}(O) = \min(\text{Likelihood}(\text{SO}_1), \cdots, \text{Likelihood}(\text{SO}_n))$$

(2) OR-refinement $(O, \text{SO}_1, \cdots, \text{SO}_n)$

$$\text{Likelihood}(O) = \max(\text{Likelihood}(\text{SO}_1), \cdots, \text{Likelihood}(\text{SO}_n))$$

3. 障碍消解

针对每个障碍，需要找到适当的障碍消解方法。与设计决策一样，首先需要确定候选的消解方法集，然后从中选择一个最适当的加以应用。消解方法的选择基于风险和性价比分析，根据障碍发生的可能性和后果的严重程度来决定。障碍消解方法的确定有三种不同的策略：障碍清除、障碍减轻和障碍容忍。

(1) 障碍清除：主要是使妨碍目标实现的条件不能发生，或使其与领域不相容。具体如下：

- 目标替换：采用不同的目标分解策略，使得被妨碍的目标不再成为需求。
- 主体替换：采用不同的职责分配策略，使得被妨碍的场景不可能发生。
- 障碍防护：通过在需求中增加新目标来显式避免该障碍条件发生。
- 目标改良：如果障碍是由过于粗略和理想化的目标导致的，则对需求目标进行改良，适当放宽目标的内容约束来避免潜在的障碍。
- 领域变形：通过改变领域性质，使得障碍条件在该领域中不再成立。

(2) 障碍减轻：障碍减轻与障碍清除的区别在于前者只是降低障碍发生的频率，而后者则使其不可能发生。对操作主体采取适当的管理措施属于障碍减轻，使主体出现异常行为和不负责任行为的可能性降低。

(3) 障碍容忍：当障碍既不能被完全避免，消解的代价又过大时，一般采取障碍容忍的策略，需要明确障碍发生时的补救方法。第一种方法是目标复原，即在目标模型中增加新目标“当障碍条件为真时，被阻碍的目标在未来某一时刻会恢复满足”；第二种方法是障碍降解，即设法降低障碍带来的后果的严重程度；第三种方法是障碍忽略，当明确目标的障碍对需求满足没有太大影响时，不采取任何措施。

3.3.3 对象识别

对象是领域特定概念的实例的集合。每个对象实例都有一个内置的、不可变更的标识，使得它能与其他对象实例区分开。系统所涉及的对象实例一般应该是可枚举的。以下是四种常见的对象类型。

实体(Entity)是能够独立存在于系统中的被动对象，不能控制其他对象实例的行为。例如，列车和站台的实例可被建模为实体对象。

关联(Association)是依赖于其他对象而存在的一种概念对象，其所依赖的两个对象分别扮演这个关联中不同的角色。例如，“停靠”是依赖于列车实体和站台实体的关联对象。

主体(Agent)是能够独立存在于系统中的主动对象，能够控制系统中的其他实体，改变其状态。例如，列车控制系统能够控制列车实体的速度属性。

事件(Event)是瞬时实体，存在于特定的系统状态中。例如，列车启动这一事件仅能够出现在列车从停止状态开始启动的瞬间。

目标模型与对象模型之间存在“涉及(Concern)”关系，即目标规格说明会涉及和/或使用对象模型中的对象。通过对自然语言描述的目标规格说明进行分析，能够从中抽取相应的对象元素。名词或名词短语可能是实体或主体，连接主语与宾语的谓语可能是关联，等等，可作为识别概念对象的启发式规则。例如，从目标“**避免**[多辆列车行驶在同一区域]”中可以获取实体对象“列车”和“区域”及关联关系“行驶在”，如图 3.7 所示。

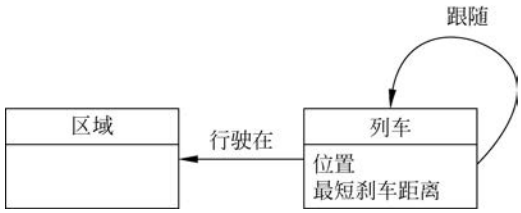

图 3.7 基于目标模型构建对象模型

如果已有目标的形式规格说明，则可以更加直观地从中抽取相应的概念对象。例如，“**维持**[安全距离]”要求随行的两辆列车之间的距离要大于后车的最短刹车距离(Worst Case Stop Distance，WCSD)，其形式规格说明如下：

$$\forall tr_1, tr_2 : Train, Following(tr_2, tr_1) \Rightarrow$$
$$tr_2.Position - tr_1.Position > tr_2.WCSD$$

从中可以获取列车对象之间的“跟随”关联，以及列车属性“位置”和“最短刹车距离”。

3.3.4 主体职责分配

要让顶层目标最终得到满足，需要将叶子目标分派给特定的主体来完成。通过职责分配可以识别和界定系统与环境的边界。职责分配的过程如下。

(1) 构建主体能力模型：在进行主体职责分配前，需要首先在对象模型基础上对主体的能力进行建模。具体地，每个主体能够控制(Control)或监控(Monitor)一个或多个对象的属性。例如，列车驾驶员能够控制列车的行驶状态和速度，速度传感器能够监控列车的速度，等等。

(2) 找出目标的候选职责分配：目标是对系统相关对象及其属性的期望，因此对叶子目标进行职责分配时，可以通过分析主体的能力确定叶子目标的候选职责分配。如图 3.8 所示，“速度传感器”所监控的对象，与叶子目标“测量列车运行速度”所涉及的对象相匹配，因此可以将实现该目标的职责分配给该主体。同理，叶子目标“控制列车速度”可以分配给“列车驾驶员”或“速度控制器”来 负责。

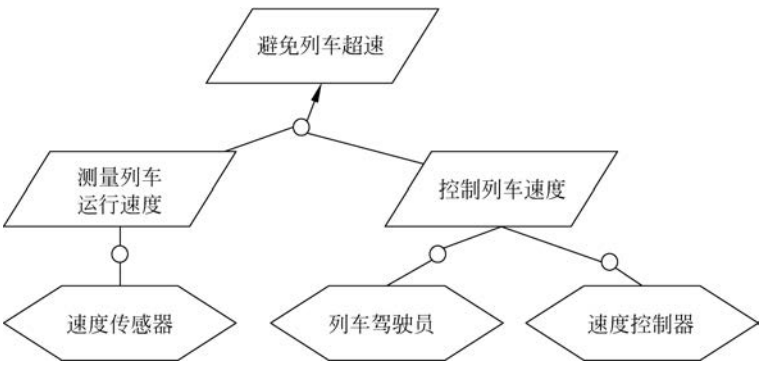

图 3.8 主体职责分配模型

3.3.5 最优方案选择和目标操作化

根据目标模型和主体模型，参照候选方案对质量目标的贡献，选择最优的职责分配方案。针对这个最优方案，对那些通过精化分析得到的系统叶子目标，需要规定实现这些目标的操作。

目标是系统需求干系人预期的系统状态，而操作可以视为实现系统状态迁移的二元关系，即从当前状态迁移至目标状态，如下所示。

$$Op \subseteq CurrentState \times TargetState$$

具体地，可以通过描述领域前/后置条件(Domain Pre-/Post-condition)来定义操作完成的状态迁移。如图 3.9 所示，对开门(OpenDoors)操作而言，其领域前置条件是列车车门状态为关闭，领域后置条件是列车车门状态为开启。操作及其领域前后置条件反映了与该操作相关的领域性质。当操作实现特定目标时，还需要根据该目标的内容识别出必要前/后置

条件(Required Pre-/Post-condition)。此外,一个操作可以操作化多个叶子目标,如果不同叶子目标要求操作满足不同的必要前置条件,则需要在操作规格说明中逐条写明,如图 3.9 所示。

```
Operation OpenDoors
    Def Software operation controlling the opening of all doors of a train;
    Input tr: TrainInfo;
    Output tr: TrainInfo/DoorsState;
    DomPre tr.DoorsState ='closed';
    DomPost tr.DoorsState ='open';
    ReqPre for DoorsClosedWhileMoving: tr.Speed = 0;
    ReqPre for SafeEntry&Exit: tr.Position is a platform position;
```

图 3.9　开门操作规格说明

目标的操作化关系与精化关系类似,也可以包含与/或操作化。如图 3.10 所示,“列车在行进中车门保持关闭”被与操作化为两个操作:一方面,需要保证车门不能在行车过程中打开,即开门操作的必要前置条件为列车静止;另一方面,需要保证列车启动时车门是关闭的,即列车启动操作的必要前置条件为车门关闭。

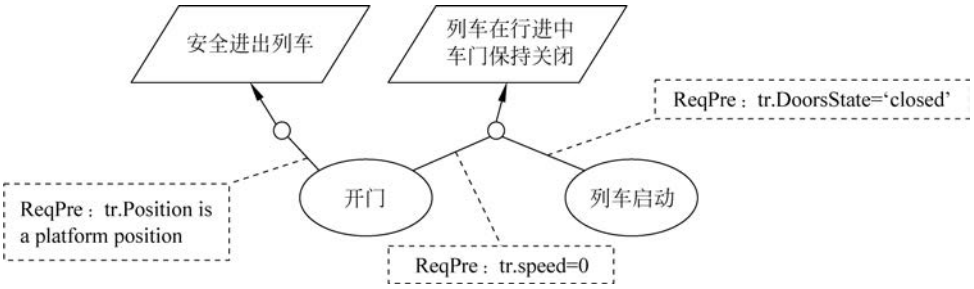

图 3.10　目标操作化模型

3.4　工业界应用

Respect-IT 公司开发的 CASE 工具 Objectiver 能够支持完整的目标分析方法[①]。它不仅能够支持系统目标的建模与分析,并且能够由分析结果自动生成软件文档,从而降低人力成本。Objectiver 的界面如图 3.11 所示。

面向目标的方法已经应用于航空、医疗、出版等不同领域。其中,在欧盟 SAFEE(Security of Aircraft in the Future European Environment)项目中的应用体现了面向目标的方法的实用价值。该项目联合 31 家欧洲企业、研究所和大学开发航空安保系统,项目规模近 3600 万欧元。在该案例中,面向目标的方法的应用步骤如下。

(1) 对系统干系人进行访谈,对航空规章制度进行分析,构建以安全目标为根的目标模型,并以此为基础进行目标操作化和主体职责分配。

(2) 围绕安全目标进行障碍分析,识别出系统的脆弱性和潜在威胁。

(3) 针对所识别的威胁,提出相应的安全措施,并构建新的需求目标模型。

① 详情参见本章参考文献[9]。

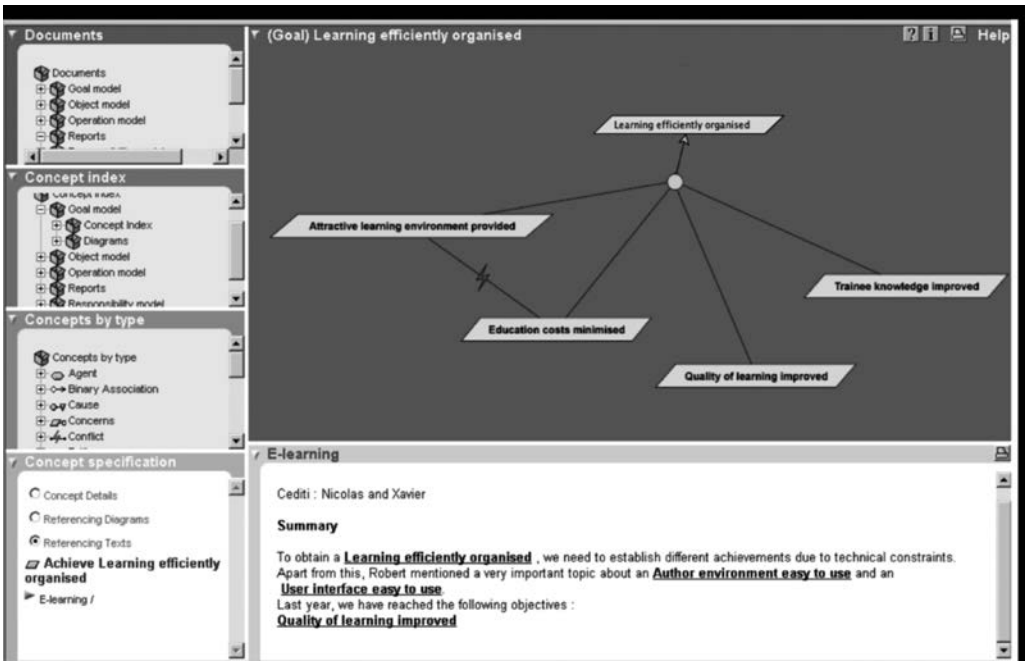

图 3.11　Objectiver 软件界面

SAFEE 项目最终产出了长达 200 页的需求文档，涉及 1400 余个相关概念，包括 25 个主体、100 余个对象、500 余个目标、300 余个威胁、150 余个需求和 300 余个期望[①]。

3.5　小结与讨论

面向目标的方法将“目标”看作软件需求的源头和依据，以目标为需求获取的基本线索，引导系统干系人按目标的精化关系，逐步构建系统目标与/或树。该方法显式地记录需求的层次，建立高层的抽象需求和低层的具体需求之间的追踪链，不仅描述需要做什么，还清楚地阐述为什么需要这样做。对于复杂系统的需求分析，基于目标精化关系所构建的目标树，能够有效且自然地对系统干系人多层次的关注点进行整合和建模。面向目标的方法也得益于目标之间的与/或精化关系，能够在早期需求分析阶段表示不同的可替代需求方案，并据此获取最优方案。

按描述的具体内容，目标可以分为功能性目标和非功能性目标。功能性目标描述要实现的服务，是干系人期望发生的所有场景的集合，要求所表达的内容是清晰一致的。非功能性目标描述对服务质量的偏好，如良好的保密性、较高的安全性、较强的准确性、较好的易用性等。按描述的抽象层次，目标可以分为高层目标和低层目标。高层目标通常是战略性的、粗粒度的、作用于组织范围的抽象目标；低层目标通常是技术性的、细粒度的和作用于系统设计层面的具体目标。

面向目标的需求分析包括以下步骤：(1)基于目标精化分析，获取系统目标结构，构建

① 详情参见本章参考文献[10]。

目标树；(2)分析并确定目标所关注的对象；(3)分析会对期望实现的目标产生阻碍的系统例外行为，识别并处理潜在的系统风险；(4)确定系统相关的主体及其具备的能力(能够完成的行为)，在此基础上确定主体职责分配的候选方案，并从中选出最优方案；(5)将目标操作化为能够保证目标被满足的操作，并将操作分配给相应的责任主体。

另外，通过与 UML 模型的深度整合，面向目标的需求分析方法在软件项目开发中的实用性得到有效提升。虽然面向目标的方法可以对半形式化和形式化的目标进行推理，但在实际项目中，该方法仍主要针对非形式化的需求描述展开，一方面是因为形式化表示的门槛较高；另一方面是因为形式化方法在非安全关键项目中的投资回报率不高，非形式化的分析已经能够较好地满足项目的需求分析任务。

3.6 思 考 题

1. 某高校学生评教系统的需求模型中包含目标“在学期结束之前，教师不能查看学生评教的结果”，请通过思考“为什么”的问题，构建上述目标的父目标。

2. 请思考并阐述需要对目标进行分类(功能性目标和非功能性目标)的原因。

3. 某电影售票系统的需求模型中包含目标“观众选择合适的电影票”，请选择合适的目标精化模式对该目标进行精化分析，并解释选择该精化模式的原因。

4. 拒绝服务是一种常见的网络攻击手法，其目的在于使目标计算机的网络或系统资源耗尽，服务暂时中断或停止，导致合法用户不能够访问正常网络服务。请根据第 3 题所构建的目标模型，针对拒绝服务攻击进行目标障碍分析建模。

5. 请根据障碍消解的三种策略(障碍清除、障碍减轻和障碍容忍)，思考并阐述两种针对拒绝服务攻击的防御措施。

6. 请说明面向目标的需求分析方法的优点，并基于一个具体的实例来佐证你的观点。

7. 请根据以下智慧医疗场景构建目标模型，并在此基础上识别出潜在的障碍。“独居老人在家中佩戴能够测量脉搏、体温、血压等的可穿戴检测设备。这些设备定期将检测数据传输到智慧医疗系统中。系统记录并保存每位独居老人的检测数据，一旦某项检测数据出现异常，将及时通知附近的社区医疗中心。”

8. 请思考当需求发生变化时如何有效地演化目标模型。以第 7 题为例，对于患慢性病、需要长期服药的独居老人，智慧医疗系统还需要监控其服药的情况。请根据该新增需求对原有的模型进行演化。

参 考 文 献

[1] Dardenne A, Lamsweerde van A, Fickas S. Goal-directed requirements acquisition[J]. Science of Computer Programming, 1993, 20(1-2): 4-50.

[2] Darimont R, Lamsweerde van A. Formal refinement patterns for goal-driven requirements elaboration [J]. ACM SIGSOFT Software Engineering Notes, 1996, Vol. 21, No. 6: 179-190.

[3] Lamsweerde van A. Requirements engineering[M]. Hoboken, NJ: John Wiley & Sons, Ltd., 2011.

[4] Dalpiaz F, Franch X, Horkoff J. iStar 2.0 language guide[EB/OL]. (2016-06-16)[2022-12-30]. https://doi.org/10.48550/arXiv.1605.07767.

[5] Jureta I J, Borgida A, Ernst N A, et al. Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling [C]//IEEE 18th International Requirements Engineering Conference, 2010: 115-124.

[6] Jureta I, Mylopoulos J, Faulkner S. Revisiting the core ontology and problem in requirements engineering[C]//IEEE 16th International Requirements Engineering Conference, 2008: 71-80.

[7] Horkoff J, Aydemir F B, Cardoso E, et al. Goal-oriented requirements engineering: a systematic literature map[C]//IEEE 24th International Requirements Engineering Conference, 2016: 106-115.

[8] Horkoff J, Aydemir F B, Cardoso E, et al. Goal-oriented requirements engineering: an extended systematic mapping study[J]. Requirements Engineering, 2019, 24: 133-160.

[9] Objectiver[EB/OL]. [2023-04-30]. http://objectiver.com.

[10] SAFEE[EB/OL]. [2023-04-30]. http://objectiver.com/index.php?id=88.