软件开发 面向对象 建模技术 KBOMT方法 (13)

计算机科学 1995 Vol. 22 №. 3

53-5

基于知识的面向对象建模技术：KBOMT 方法

李师贤 周晓聪

（中山大学计算机系 广州 510275）

TP311-52

摘 要 Integrated the knowledge representation method and inference technology, the KBOMT method are proposed in this paper, as the extension of the OMT method. The KBOMT method tries to use the formal language to describe the process and result of system analysis.

关键词 Problem description, Object model, Dynamic model, Functional model.

一、引 言

人们对结构化开发方法研究和使用了多年，发现软件需求分析面临的最主要的困难是[1]：

·系统分析员可能工作在陌生的应用领域，但又要在短时间内理解所开发系统的整个问题空间。

·分析员与用户之间没有有效的交互手段，难以建立起对系统一致的理解。

·用户需求的不断变化更使分析员感到迷惑，很难建立起对系统稳定的理解。

为了克服这些困难，人们开始将注意力转向面向对象的需求分析。从一般系统论角度看，系统结构是系统内部各个组成部分相互作用的表现，系统功能是系统与外部环境相互作用的表现。人作为一个特定系统外部环境的一部分，认识系统总是从认识系统功能开始，这是基于功能分解的结构化分析方法容易被人们理解和掌握的主要原因。软件需求分析过程其实就是分析员认识系统的过程。

基于这种观点，我们认为，面向对象的需求分析过程也应该从认识系统功能开始，而且要深入到对系统结构的分析，这是面向对象需求分析的核心，即强调对系统结构的理解而不仅仅是功能分解。不同的系统结构可实现同样功能，需求分析的目的就是从现有的系统结构出发，构造出更适合计算机实现的系统结构。更重要的是，同样的系统结构可实现不同的功能，结构比功能具有更好的稳定性，从而面向对象分析方法就能更好地处理变化的需求。

总之，需求分析方法采取从分解系统功能到理解系统结构，再回到如何实现系统功能的思路就更符合人们认识系统的过程，整个分析过程就显得更自然清晰，容易被用户和分析人员理解掌握，用户和分析人员之间的沟通也可变得容易。在从功能到结构再到实现的过程中，应用领域中的一般知识会显得十分重要，可以辅助分析员形成好的系统结构，因此在需求分析方法中应提供对这些知识的描述。KBOMT 就是基于这样的观点提出的一种面向对象的需求分析方法。

二、面向对象的建模技术(OMT 方法)

KBOMT 是在研究 James Rumbaugh 等人[2]提出的面向对象建模技术(OMT 方法)的基础上提出来的。OMT 方法从问题陈述开始，理解问题陈述中的客观世界，将其本质抽象成模型表示，建立系统的三种模型，即对象模型，描述系统中的对象及其关系；动态模型，描述系统中对象的交互行为；功能模型，描述系统中数据的变换过程。

OMT 方法体现了面向对象的系统开发方法的基本特点：

·强调对系统结构的理解，而不是系统功能的分解。在 OMT 方法中对象模型最重要，动态模型次之，最后是功能模型。

·使用面向对象的基本思想构造的系统模型与客观系统的结构十分类似，容易使用该模型与用户通信。

·在分析阶段得出系统模型后，系统设计的任务主要是细化模型。分析和设计可使用统一的表示

李师贤 教授，当前研究方向包括软件工程、计算机语言的形式语义等。周晓聪 硕士研究生，研究方向是计算机软件工程和人工智能。

方法，省略了类似结构化方法中从数据流图到模块调用层次图的转换过程，而且这种模型用面向对象的程序设计语言来实现也显得十分自然。

但OMT方法还存在以下几个方面的不足：

·对问题陈述论述得不足。在KBOMT方法中，我们在研究分析人员可能对系统做的调查及其步骤的基础上定义了一种问题描述语言。

·三种模型的一致性难以检测和维护，而且对系统约束的描述能力也不足。KBOMT中结合框架型的知识表示方法，并在模型中引进规则和对象中的不变式，操作中的前后置断后，加强对系统约束的描述，同时也为检测模型的一致性提供了更多的支持。

·建模过程描述得不很清晰，从建模的结果即各种图形表示也不能体现建模的过程，从而增加了分析人员掌握该方法的难度。

·OMT方法使用的图形表示在分析大系统时不容易画得条理清晰。在KBOMT方法中使用形式化语言来描述三种模型，并强调使用层次结构来组织这些描述，在大系统分析中也可以做到条理分明，并且这种描述语言能够体现KBOMT中建模的过程和思路。

三、问题陈述

面向对象的分析方法要从用户对问题的描述入手。系统分析人员在调查分析现有的系统时，应该引导用户从时间、地点、操作条件、操作对象、操作、操作结果来描述系统的功能。对一个大系统，可用分层的方法来组织这些描述，通过高层次用户，如管理人员对系统的总体概述得到问题的高层次和概括的描述，然后针对具体的操作利用底层用户的陈述进行细化，从而得到对系统的一个树状形式的描述。在这种树状描述中，我们可以得到系统不同层次的抽象，从高层往底层看，每一层都只描述系统能干什么，而不关心系统的功能具体是怎样实现的；而从低层往高层看，低层就反应了高层是怎样完成其功能的。至于应该组织多少层，则取决于系统的大小与复杂度。

在完成从功能的角度描述系统之后，分析员应该针对上面获得的数据信息，即操作对象和操作结果的内容（后面我们称之为数据项），从以下几个方面详细分析这些数据项，得到从数据角度对系统的描述：

·每一数据项可能有哪些重要的属性。在上面的功能描述中，有些数据项可能是其他数据项所具有的性质，因而可得到一些数据项所具有的重要属性。

·每一数据项可能由哪些数据项组成。要注意区别数据项的组成部分和数据项的属性，一般来说，在目前要解决问题的角度下，数据项的组成部分还可以有自己的组成部分和属性，而我们却不再关心属性的组成部分和其拥有的性质。

·每一数据项是否是某一数据项的特例。

·每一数据项可能产生的最大规模。

·每一数据项可能产生的最大频度。

通过每一数据项的分析，分析员对系统的描述逐渐从功能角度过渡到以数据为中心。在与用户一起进行问题陈述的整个过程中应始终坚持使用应用领域的语言，并且自顶向下地结构化，使用户和编程人员既能抓住问题的核心，很快地理解现有系统的大致运转方式，又对系统有一定深度的了解。

为了更好地支持分析员与用户一起对系统从功能和数据的角度进行分析理解，我们定义了如下的形式化语言框架来表示这种问题陈述的过程及对系统描述的结果，从而使得形式化描述系统三种模型有了基础，也易于利用专家系统的思想支持从问题陈述向系统三种模型的自动转换：

```
SUBJECT/问题 subject_name/名称
  {STEP/阶段 | PART/部分|   // 一个问题可分为多个阶段或部分
    [TIME/时间: formated_time/格式化时间]
    [PLACE/地点: place_name/地名]
    [SOURCE/源数据:]
      {data_item_name/数据项名}          // 可有多个源数据
    // 前置条件描述源数据应满足的约束及处理启动应满足的时序条件
    {PRECOND}TION/前置条件:}
      {formated_condition/格式化条件}  // 可有多个前置条件
    {PROCESS/处理: subject_name/问题名称} //引出子问题可再进一步细化
    {RESULT/结果:}
      {data_item_name/数据项名}          // 可有多个结果
    {POSTCOND}TION/后置条件:}   // 后置条件描述结果数据应满足的约束
      {formated_condition/格式化条件}  // 可有多个后置条件
END/结束

DATA {ITEM/数据项 data_item_name/数据项名
  |SCALE/可能的规模: ONE/一个 | MANY/多个}
  |FREQUENCE/产生的频度: ONE/一次 | MANY/多次}
  |ATTR{IBUTE/属性:}   // 属性也是数据项
    |data_item_name/数据项名, data_type/类型}  // 可有多个属性
  |RESTR{ICTION/约束条件:}      // 本属性应满足的约束条件
    |formated_condition|      // 可有多个约束条件
  |COMPONENT/组成部分:}  // 组成部分也是数据项
    |data_item_name/数据项名{|: ONE | MANY}    // 可有多个组成部分
  |SPEC{IF}CAT{ION/特例:}  // 特例也是数据项
    |data_item_name/数据项名}  // 可有多个特例
END/结束
```

数据项的属性可以是问题部中所涉及的数据项，也可以是问题部中所不涉及的数据项，而是由系统分析员根据常识判断它在系统中的重要性而添加的，一个数据项既然确定为其他数据项的属性，本身就不应该再拥有属性，而数据项的组成部分和特例则可拥有自己的属性。

四、建立对象模型

在建立对象模型时常使用下面的概念来对应现

实系统的结构：

·对象(object)。是客观世界的某种实体，相当于一般系统论中的系统的概念，对象是普遍存在的。任何实体在我们考虑问题的一定范围内都可认为是对象，后面我们使用实体作为对象的等价名称。

·属性(attribute)。对象可拥有属性，属性描述对象存在时各方面的性质。属性本身也是一个对象，只是在我们考虑的问题范围内不再研究其本身所具有的性质，而用它来描述其它对象的性质，这时它成为该对象的属性。

·关联(association)。描述对象与对象之间的关系。在面向对象系统所提到的关联从属性对比的角度来说通常有如下三种：

①引用关系(reference)。是对象与对象之间存在的最为普遍的一种关系，它其实是对现实世界各种形形色色关系的概括。根据上面的描述语言，它必须从数据项与数据项之间的操作关系抽象出来，这是一种最难以形式化的关系。

②一般和特例的关系(generalization/specification)。是人们从某个角度对实体所具有的属性进行对比，发现许多实体的属性有许多共同之处，从而形成了实体之间这种一般和个别、普遍和特殊的关系。

③整体和部分的关系(composition)。是人们在认识复杂实体的过程中，采取分而治之的原则，将其分割成几个相对简单的部分而认识到的实体之间的关系。

从参与关联的实体的实例数量对比来说，关联可分为一对一关系、多对一关系和多对多关系。这种实体的实例数量，我们称之为该实体在该关联中的阶。

从参与关联的实体的数量来说，关联可分为二元关系和多元关系。二元关系研究的比较多，而多元关系一般都能化作二元关系。

为了提高模型的语义表达能力，对象模型中还使用下面一些概念：

·关联的受限。引入受限的关联是因为在发生关联的实体中往往存在特殊的属性，它虽然是某个实体的属性但与这种关联关系密切，按照这种属性查找相关联的实体实例往往变得更为容易。

·实体在关联中充当的角色。引入角色的概念是因为实体一般来说会与多个实体发生多种关联，因而在不同的关联中该实体可能充当不同的角色。

·实体在发生关联时的顺序。在一对多或多对多的关联中，多个实体实例参与关联，有时候这多个实体参与关联的顺序也很重要。

在OMT方法的基础上，我们引进规则[3-6]来简化模型和用于检测模型的一致性。具体说来，对象模型中可在下面几种情况使用规则：

·对象内部使用规则来说明对象的属性之间应该满足的约束。

·在关联中使用规则来说明参加关联的对象应该满足的约束。

·其它规则用来说明系统中出现的关联之间的关系。

使用规则可以隐含地说明系统的派生属性和派生关联，简化了对象模型。而且我们可用它来表达一般常识及应用领域的一般知识。对象内部使用的规则相当于某些面向对象程序设计语言所支持的类不变式，关联中使用的规则及说明关联之间关系的规则在一定程度上反映了动态模型中事件发生的条件和功能模型中对处理的约束，因此可以用来检测三种模型的一致性。

根据问题陈述中得到的系统描述，对象模型中的对象相当于其中的哪些不再充当其他数据项属性的数据项是容易确定的。确定了对象之后，对象之间的关联要通过分析功能描述来得到，最初可将其中的处理名作为源数据对象和结果数据对象的关联，然后对这些关联进行归类及一般化，去掉不必要的和重复的，再取一个更富有意义的名称。这样我们得到了系统最初的关联，通过建立系统的动态模型和功能模型，我们将对系统有更进一步的认识，可再对对象模型进行优化。对象模型中的规则主要来自对操作的约束及一般常识和应用领域中的一般知识。我们使用下面的形式化语言来描述对象模型，这种语言试图反映建立对象模型的一般过程。从下面的语言中也可得到关于对象模型更多的细节(见下页左上角)。

在这里我们不再进一步定义参数类型、条件表达式和值表达式，这并不影响对整个方法的说明。上面一共使用了三种规则形式：

MUST HOLD/必须满足：condition_exp/条件表达式]

IF/如果 condition_exp/条件表达式 THEN/则 value_exp/值表达式]

IF/如果 association_exp/关联表达式 THEN/则 association_exp]

MUST HOLD 规则相当类不变式，说明实体和关联中属性必须满足的约束条件。在实体和关联中

```
ENTITY/实体 entity_name/实体名 [IS_A | IS_PART_OF  entity_name,
  [ATTRIBUTE/属性 ]
    [attribute_name/属性名称  TYPE/类型]  // 可有多个属性
  [SERVICE/服务 ]
    [NAME/服务名称, service_name]
      [PARAMETERS/参数:]
        [parameter_name/参数名称: TYPE/类型]  // 可有多个参数
      [PRE-CONDITION/前置条件: condition_exp/条件表达式]
      [POST-CONDITION/后置条件: condition_exp/条件表达式]
                // 可有多个服务
    [RULE/规则:]  // 可有多条规则
      [MUST HOLD/必须满足: condition_exp/条件表达式[
      [IF/如果 condition_exp/条件表达式 THEN/则 value_exp/值表达式]
END/结束

ASSOCIATION/关联 association_name/关联名称
  [BETWEEN]
    [ENTITY entity_name: ][multiplicity/阶数[{ ORDERED/排序]
      [AS role_name/角色名]
      [QUALIFIER/限定词: qualifier/限定词[
            // 可有多个实体说明，注意这些实体之间并无 ASSOCIATION 中
            // 说明的关联，在这里它们完全是平等的，都与 AND 后引出的
            // 实体具有 ASSOCIATION 说明的关联.
    [AND]
      [ENTITY entity_name: ][multiplicity/阶数][ ORDERED/排序]
        [AS role_name/角色名]
        [QUALIFIER/限定词: qualifier/限定词]
        // AND 后只能引出一个实体.
              // 可有多个 AND 分句，两个 AND 分句表示了二元关系，而多个
                 AND 分句表示多元关系.
    [ATTRIBUTE/链属性:]
      [attribute_name/属性名称: TYPE/类型]  // 可有多个属性
    [RULE/规则:]  // 可有多条规则
      [MUST HOLD/必须满足: condition_exp/条件表达式]
      [IF/如果 condition_exp/条件表达式 THEN/则 value_exp/值表达式]
END/结束

RULE/规则 (全局规则)
  [IF/如果 association_exp/关联表达式 THEN/则 association_exp[
  ....      // 可有多条规则
END/结束
```

可使用的 IF THEN 规则用来说明实体和关联中的属性之间的推导关系。全局规则中使用的 IF THEN 规则是 Prolog 中规则的自然语句形式，用来说明关联与关联之间的推导关系，可用来表示应用领域的一般知识。规则的具体形式及怎样利用规则简化系统模型和检测模型的一致性有待于进一步研究。

五、建立动态模型

对象模型从横向(空间)上将系统所涉及的对象组织成类，并利用继承、聚集及引用等概念将这些类组织成一定的结构，描述对象类之间的关联；而动态模型使用事件和状态等概念在纵向(时间)上对系统中各个类的发展过程进行划分。

动态模型的建立从分析系统可能发生的外部事件开始。事件指两个实体发生相互作用时的一次交互过程，这种过程的延续时间可以忽略。系统可能发生的外部事件一般指外部设备的数据输入。在得到这些外部事件后，进一步分析系统中响应这些事件的实体。实体在响应这些事件的时候，又会触发一些事件，这些事件实际上是这些实体的响应动作与其相关联的实体发生相互作用的一种表现。一步一步地分析下去，就会得到整个系统中可能发生的事件及其响应实体。

以实体为中心来看，它所响应的可能的事件将其生存周期划分成实体的状态。实体的状态可以看成是实体某些属性的取值，这些取值随着时间的变化而变化，这种变化的动力在于事件的刺激，变化的方向又取决于实体当前所处的状态。实体的状态还可能是实体当前一直在做需占用一定的时间的操作，这种操作称为活动。这种活动结束后，实体还可能需要自动地做某种动作而进入另一个状态。这样实体响应事件执行活动可能自动调用两个动作：入口动作和出口动作。

分析出各实体的状态之后，可构造描述该实体动态变化的脚本，它包括实体当前状态及发生的事件，实体的响应动作，以及进入到下一状态整个过程的描述。

对事件和状态及事件引起状态转换的脚本的描述都可形成不同的抽象层次，就是说事件可能需要进一步细化，实体状态也可细分为子状态，从而建立动态模型的过程也成为一种逐步求精的过程。

根据上面的分析，我们使用下面的语言框架来描述系统的动态模型：

```
EVENT/事件 event_name/事件名 [IS_A event_name/事件名]
  [ATTRIBUTE/事件属性]
    [attribute_name/属性名称, TYPE/类型]  // 可有多个属性
  [TIME/发生时间: time_description/时间描述]
  [CONDITION/发生条件: format_condition/格式化条件]
  [RESPONSE[ON/响应]
    [ENTITY/响应实体: entity_name/响应实体名]
    [ACTION/响应动作: action_name/响应动作名]
END/结束

STATE OF ENTITY/实体状态 entity_name/实体名
  STATE/状态: state_name/状态名 [IS START | END STATE ]
    [SUB-STATE OF: state_name/父状态名]
    [FEATURE/特征: feature_description/特征描述]
    [ENTRY ACTION/入口动作: action_name/动作名]
    [ACTIVITY/活动: activity_name/活动名]
    [EXIT ACTION/出口动作: action_name/动作名]
          // 可有多个状态，一般按时间顺序排列.
END/结束

SCENARIO OF ENTITY/实体脚本 entity_name/实体名
  [CURRENT STATE/当前状态: state_name/状态名称]
    [EVENT/事件: event_name/事件名]
    [ACTION/响应动作: action_name/响应动作名]
    [NEXT STATE/下一状态: state_name/状态名]
          // 多个这样从当前状态到下一状态过程的描述构成脚本
END/结束
```

六、建立功能模型

问题陈述中对功能的描述是为了得到系统结构，正如在这之后开始建立对象模型和动态模型。建立功能模型的时候，要从已经得到的系统结构，考察系统结构与系统功能之间的对应关系。这时我们根据问题描述中对系统功能的分解，研究对象模型中得到的实体及其关联同各个子功能之间的对应关系，这里借用传统的结构化方法使用的数据流图来描述这种对应。同结构化方法中使用数据流图对系统进行功能分解不同，这里是在理解系统结构的基础上，使用数据流图描述系统结构和系统功能之间的对应关系。

结合数据流图的表示方法，我们使用下面的语言框架来描述系统的功能模型：

```
PROCESS/处理 process_name/名称
  [STEP/阶段]   // 一个处理可有多个阶段/子处理
    [FLOW IN/入数据流]
      [data_flow_name/数据流名]   // 可有多个入数据流
    // 前置条件描述入数据流应满足的条件
    [PRECONDITION/前置条件]
      [formated_condition/格式化条件]   // 可有多个前置条件
    [SUB-PROCESS/子处理, process_name/问题名称] //引出子处理
    [FLOW OUT/出数据流]
      [data_flow_name/数据流名]   // 可有多个出数据流
    // 后置条件描述出数据流应满足的条件
    [POSTCONDITION/后置条件]
      [formated_condition/格式化条件]   // 可有多个后置条件
END/结束

DATA STORE/数据存贮 entity_name/实体名
  [ATTRIBUTE/属性]
    [attribute_name/属性名称: TYPE/类型]   // 可有多个属性
END/结束

ACTION ENTITY/活动实体 entity_name/实体名
  [ACTION/有关活动]
    [action_name/活动名称]   // 可有多个有关的活动
END/结束
```

充当数据存贮的实体是一种被动实体，活动实体则是一种主动实体，它通过生成或使用数据来驱动数据流图。活动实体应包括在数据流图的输入/输出中，它位于整个系统的边界，产生整个系统的入数据流或出数据流，是系统之外的实体（如用户，也可是计算机终端等其它外围设备）的抽象。

上面的描述语言与问题描述的语言有些类似，但它们有本质的不同。问题描述从认识系统的功能开始，来认识系统结构，是自顶向下的逐步求精过程，功能模型是从系统结构开始，看怎样实现系统功能，是自底向上的构造过程。这种类似体现了认识系统的螺旋上升过程，也体现需求分析是一个反复的过程，可以对对象模型和动态模型进一步优化。

七、结论

上面描述了基于知识的 OMT 方法的基本思想，我们认为 KBOMT 与 OMT 方法的不同之处在于以下几个方面：

• KBOMT 中结合知识表示方法和推理技术对实体和关联的约束进行更细致的描述，可以对模型进行简化，对三种模型的一致性检测也可提供更好的支持。

• 以一般系统论的基本理论为指导，使得建立三种模型的过程更为清晰，建立模型的目的也得到进一步明确。从认识系统的功能到分析系统的结构再到在这种结构的基础上考察如何实现系统功能的过程是 KBOMT 方法的核心。

• KBOMT 使用较形式化的语言作为描述模型的工具，它同 OMT 方法中使用的图形表示方法比较有如下特点：

①使用图形表示在分析小系统时有直观的优点，但当分析大系统时，过多的图表及图表上的图符和字符就会给人一种紊乱的感觉。使用形式化的语言描述则仍可做到条理清晰，层次分明。可结合这两种表示方法，在对系统比较粗的描述中，除了使用形式化语言外，可利用图形表示以取得直观效果。

②使用形式化的语言来描述三种模型可以使分析建模的过程更好地用计算机支持。在这种形式化语言的基础上可更好地构造 CASE 工具，并可利用专家系统的思想，支持从系统的问题陈述到系统三种模型的自动转换，从而在一定程度上支持系统原型的自动生成。

在进一步的研究中，我们将对从问题陈述到建立三种模型的过程作更详细的分析，力图给出一种形式化的方法来完成这种转换过程，并更好地体现知识在 KBOMT 中的作用，使基于面向对象开发方法的快速原型系统自动生成成为现实。

主要参考文献

[1] Peter Coad, Edward Yourdon, Ob,ect-Oriented Analysis, Yourdon Press, Prentice Hall, Englewood Cliffs, NJ, 1990

[2] James Rumbaugh et al., Object-Oriented Modelling and Design, Prentice Hall, Englewood Cliffs, NJ, 1991

[3] Donovan Hsieh, A logic to unify semantic-network knowledge system with object-oriented database models, JOOP, Vol. 6 No. 2, 1993

[4] Ian Graham, Migration using SOMA: A semantically rich method of object-oriented analysis, JOOP, Vol. 5 No. 9, 1993

[5] Janet M. Drake, et al., Document-driven analysis: description and formalization, JOOP, Vol. 5 No. 7, 1992

[6] James J. Odell, Specifying requirements using rules, JOOP, Vol. 6 No. 2, 1993

[7] Sangbum Lee, Doris L. Carver, Object-oriented analysis and specification: a knowledge base approach, JOOP, Vol. 3 No. 5, 1991

[8] Hyung-Sik Park, Abstract object types = abstract knowledge types + abstract data types + abstract connector types, JOOP, Vol. 4 No. 4, 1991

[9] Dennis de Champeaux, Penelope Faure, A comparative study of object-oriented analysis methods, JOOP, Vol. 5 No. 1, 1992

[10] 汪成为，郑小军，彭木昌，面向对象分析、设计和应用，国防工业出版社 1992