信息系统的面向对象分析

郭 江 （中国科学院计算中心，北京100080）

摘 要

First, the article introduces a new method of the Information System Analysis, i. e. Object-Oriented Analysis. Then it discusses the main principles and methods of the Object -Oriented Analysis. At last, the article compares the method of Object-Oriented Analysis with others.

一、分析的挑战

系统分析既使人振奋也使人沮丧，在所有类型的项目中有三个主要的难题困扰着系统分析员，即：问题空间的理解、人之间的沟通，以及需求不断变化。

1.问题空间

分析员面对的最大问题就是对应用领域的研究。在大多数情况下，我们并没有透彻地研究委托人的应用领域。一个分析员需要确定、处理和精确描述需求，让别人能读懂并理解他所认识到的需求。但是，理解问题空间才是分析的关键和难点。因为，没有对问题空间的透彻理解，对需求的思考就可能很模糊。尤其对于大而复杂的系统，正是基于这种理解，分析员才能将问题空间模型化并提出需求说明。分析是一个不断学习、深入研究应用领域，并且逐步了解其细微差别的过程。

2.人之间的沟通

一个分析员应该在分析过程中努力和其他相关的人进行交流。分析员和委托人交流，抽象出问题空间和需求。然后，分析员将自己的观点和同伴交换讨论。最后，分析员将他对问题空间的理解以及随后的需求反馈给委托人，证实他对需求的理解。分析员也需要帮助委托人摆脱那些在预算和计划限制中无法满足的需求。

软件工程实际上就是一个面向人的工程，软件工程中的各种方法都是为了解决人之间沟通的问题。而且，目前仍然是，有效的沟通——管理者、同伴、审核者、制定标准人员和委托人之间的沟通——是成功的信息系统分析的关键。具体而实用的软件工程方法必须有利于人之间的沟通，而不仅仅是一个易于实现计算的说明方法。

3.需求不断变化

需求总是处在不断的变化之中。委托人可能在特定的时间内给出一个不变化的需求。但是真正的需求和所需要的系统却是一直在不断地发展。许多因素影响着这个不断变化着的需求，如：顾客、竞争、调节者、支持者，以及技术人员等。正如Gerhard Fischer指出："我们不应该谴责需求的变化是草率思考的结果，而应接受这个现实"。

一个分析员应该努力组织他的说明（需求说明）和策略，使他的工作结果富有弹性。使需求能在一段时间内保持相对稳定。对共性的清晰记录是有助于稳定的，对数据和处理两者亦如此。

二、OOA的基本原则

下面讨论奠定面向对象分析(OOA)基础的主要原则。

1.抽象

抽象是为了更好地集中注意力而忽略那些和当前目的不相关的其它方面。在现实世界中我们处理的大多数事务——人、位置、对象和系统——本质上讲，都是相当复杂的，远远超出了我们一次所能处理的程度。因此，当我们使用抽象概念时，承认所考虑的事务是复杂的，我们不企图理解整个事务，而只是选择它的一个部分。这个技术是减少复杂程度的一个重要手段。

过程抽象是由需求分析员、设计者和程序员所广泛使用的一种抽象形式。它通常以"功能/子功能"抽象为特征。过程抽象指任何一个产生预先定义的结果的操作都可以被用户作为一个单个实体来对待，尽管这个操作实际上可能是由一系列低一层次的操作来实现的。

将处理分解成若干个子步骤是减少复杂程度的一个基本方法。但是，对于组织分析和说明来说，使用这种分解法有某种程度的任意性和易变化性。过程抽象不是OOA的一个基本抽象形式，但是它在OOA说明和描述单个对象时仍有用。

另一种更为强有力的抽象机制是数据抽象，是OOA的核心。这个原则形成了思考和说明的基本组织的基础。数据抽象是指按照应用于类型对象的操作来定义数据类型，同时加上使用这些操作而产生的对象限制。

在OOA中，一个分析员不仅要定义对象的属性，也要定义这些属性专用操作的处理，处理是获得属性的唯一途径。属性和它们的处理在OOA中被作为一个固有的整体来对待，在OOA中，属性、对象和处理是相当重要的。

2.信息隐藏

信息隐藏指在开发一个总体程序结构时，程序的每个组成部分应该隐藏一个单个的设计决策。模块之间接口的定义也应该尽可能少地暴露模块的内部工作方式。信息隐藏的力量和魅力就在于能减少复杂程度，在开发一个新系统时能帮助极大地减少重复性工作。如果分析员在分析过程中隐藏了需求分析中最易于变化的那些部分，那么需求的变化对总体努力结果的威胁就减小了。将易变性局部化是这种思想的本质。实现信息隐藏就是为了将一个对象的用户和它的作者分离开来。

3.继承性

继承性指从一个祖先接受性质或特征。这个原则形成清晰地表达共性的强有力的技术基础。继承性允许我们一次确定公共的属性、处理、以及特殊性，并且将这些属性和处理扩展到特殊的情况之中。OOA以需求分析的早期结论为出发点，用继承性来清晰地表达共性。

在使用继承性这个概念时，一个接收者可以从发送者接收属性和处理，并且接收者可以对这些性质进行增加或扩展。

4.组织方法

我们研究了各种软件的主要思想，特别是语义数据模型化和面向对象的程序设计语言，从中找出了可以用于组织和表示需求的一些关键原则和人们在理解现实世界不断利用的以下三个组织方法：

①特定对象及其属性的区别——例如，当他们区别一棵树和它的大小或和其它对象的空间关系。

②整体对象及其组成部分之间性质的区别——例如，当他们比较一个树和它的分枝时。

③不同对象类的形成以及它们之间的区别——例如，当他们形成所有树的类和所有石头的类，并对它们进行区别。

OOA的说明和方法是建立在这三个常用的组织方法之上的。在OOA中，它们分别是指：对象和属性、组装结构，以及分类结构。

三、分析的方法

什么是系统分析？Demarco提出了下面的定义："分析是在采取某种行动之前对问题

的研究”。对我们而言，分析就是对问题空间的研究，并且产生出一个人们可以观察到的行为的说明。即一个对需求的完整的、一致的、以及灵活的叙述。

分析意味着一个信息系统的需求的产生。即系统做什么才能满足用户的需要，而不是系统是怎样才能实现。需求包括功能操作和操作特征，如易使用性、可靠性、有效性、可维护性；需求也包括软件必须处理的接口、软件必须适应的环境、以及任何其它应用设计方面的限制。

一个需求文档有两个目的：一是它对用户的需求形式化；二是它建立起一个命令的清单。

下面讨论信息系统需求分析的四个主要方法：功能分解、数据流法、数据模型化法和面向对象的分析。前三个方法在系统开发领域已经讨论和实践了十几年，既有成功也有失败，而且在第四个方法(OOA)中的某些地方也有强大的生命力。因此，重要的是不放弃原有的、已建立的方法，而应努力将前三个方法中的最佳思想结合到一个更为广泛、更加全面的方法之中——OOA。我们不强调使用哪种方法更好，而是强调实用，强调在一个特定的情况下要使用多种方法的有机结合。

为了易于识别，下面的每个方法都用一个等式来表示。

1.功能分解

功能分解＝功能＋子功能＋功能化接口

功能分解的基本策略是：使用以前的经验作为指导，为新系统预测并选择适当的处理步骤及子步骤。将注意力集中在新系统需要处理什么这一焦点上，然后分析员去确定处理和功能接口。

功能分解需要人将问题空间映射到功能和子功能上。分析员必须在内部确定主要事务，并且给出系统提供的所需功能和子功能的文档。但是产生的文档却只能间接地映射回主要事务上面。因而，分析员必须理解主要事务，并且在处理前和委托人进行讨论加以确认。如果不能将功能化的结果清晰地映射回主要事务本身上，那么分析员和用户就难以估价需求是否是新系统所需要的准确描述。因此用这个方法既不能清晰地表达也不易验证分析员对问题空间的理解；还不能明确地揭示问题空间的细微差别。

尽管如此，功能分解还是很有用的。在最终确定数据和数据处理时，在为特定的对象定义某种处理的情况下，OOA仍要使用功能分解，换句话说，要将大的、复杂的处理分解成若干较小的处理模块。

由于功能/子功能分解方法对于处理结构（间接映射）和易变性（功能块的能力在不断变化）这两个方面都很困难，所以全局分析不应该建立在功能/子功能的分解上面，而是需要一个更加稳定的分析观点和说明的组织，这就是OOA方法。

2.数据流方法

数据流方法＝数据（及控制）流＋数据（及控制）转换＋数据（及控制）存贮＋连接器＋最小化确定＋字典。

数据流方法有时也称为“结构分析”或“实体关系图法”。这个方法中，分析员要将现实世界映射到数据流和处理上。这种映射需要分析员在考察现实世界时去跟踪数据流向，并且将数据流映射到随后的分析和说明之中。

现代通常使用的数据流方法是1982年提出的。这个方法主要是为了定义处理而首先定义事件，事件则由系统响应外部事务而产生。用户和系统间的实际交互也是通过信息来完成的，并且使用基于信息的交互模型来直接表达通讯。

在数据流方法中，为每个事件定义一个处理，然后给每个处理加上输入输出流。再将数据存贮放到需要用数据保持通讯的处理之间，并逐步导出高层处理，在最高层整个系统表示成一个单个的处理。对低层的处理来说，如果需要，还可以应用功能分解产生

更低层次的处理。

数据流方法存在的问题之一是：太多的事件使得处理的数目超过了人们的处理能力；当然，事件的分类有助于这个问题的解决。

另一个问题是数据字典的大小。如果存在五层的流图，可能就需要数百个数据流等式，而且复杂的接口也会加重数据字典的负担。因此，实际系统常导致数据字典的"大爆炸"。尽管CASE工具有助于摆脱语法问题，但仍难以克服语义问题。因此，沟通问题，就不能很好地解决。另外，由于接口的易变性，使得有复杂接口的系统的不断变化加重了语法和语义(更重要)的不一致性问题。

数据流法的第三个问题是过分强调功能化，致使对付变化的弹性减弱，而且不太重视数据存贮和数据结构。

但是数据流方法在现在的许多实际系统中还是很成功的。因为大多数系统可以认为涉及两个处理，一是获得数据并将之加入到一个存贮之中，二是从那个存贮中取出信息、传递数据，而且处理可以通过逐步求精加以实现，所以数据流方法在特定环境下仍有其优点。

3.信息模型化方法

信息模型化方法＝对象＋属性＋关系＋类型/子类型＋相关对象

过去的策略一般是：开发一个需求的属性表，将这些属性置于对象之中，增加关系，求精类型/子类型和相关的对象，然后进行形式化。新的策略与过去的基本相同，只是起始时在现实世界中寻找对象，并用属性来对这些对象进行描述。

对新的策略来说，信息模块直接从问题空间映射到模型中的对象上。这对映射是个很大的改进，但却需要更详细的映射。信息模型化总体上讲是一个局部的方法。

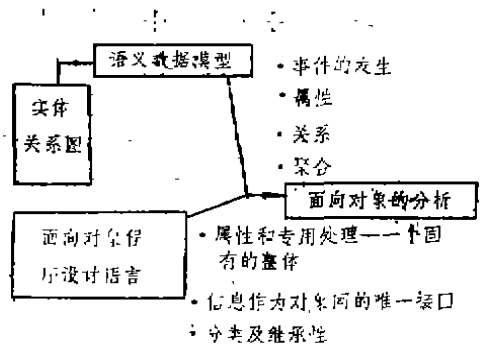
4.面向对象的分析(Object—Oriented Analysis)

面向对象＝对象（一些属性和专门处理的信息隐藏）一个问题空间中用一些实例来对一些事物的抽象）＋分类＋继承性＋用信息来通讯

"面向对象"是一个相当困难的主题，因为"对象"这个词来源于两个非常不同的研究领域：(1)从信息模型化方法中来，意味着某个现实世界的事物的一种表示，这个事物的一些实例。(2)从面向对象的程序设计语言中来，意味着由一个称为"类"的静态描述定义的、运行时进行的一些处理和值的实例。Ada是面向对象的吗？不！尽管键入的号数是很方便的，但不能代替继承性。信息模型化是面向对象的吗？不!它缺少处理、分类和继承等环节。

OOA是建立在信息模型化（实体关系图和语义数据模型）的最佳概念之上，也是建立在面向对象的程序设计语言的最佳概念之上。从信息模型产生出来了属性、关系，以及一个作为问题空间中某事的一些实例的表示；从面向对象的程序设计语言产生出来了属性隐藏和专用的处理，产生出来了作为一个固有整体的属性和处理。产生出来了分类结构的描述以及通过继承性而对共性的清晰表达。从问题空间到功能/子功能的映射或从问题空间到数据流和处理的映射不再是间接的了而变成了直接的。

下图将OOA进行了形式化：



OOA的形成基于下面方法的统一应用。①主要的组织方法(分析和说明的总体框架)②信息通讯(用户和系统之间的交互，系统

中实例之间的交互）③行为分类（确定由每个成份提供的处理的总体框架）。

为什么要使用OOA呢？有七个主要的理由：

（1）在组织的三个基本方法（对象和属性、分类结构、组装结构）的主要框架（主体结构）中定义和交流需求。

（2）基本上将注意力集中于对问题空间的理解。

（3）将属性和属性上的专用处理作为一个整体来对待。

（4）分析和确定对自包含划分的使用（对象之间的依赖性减少到最小）。

（5）通过对共性的清晰表达（继承性）来获得最佳的效果。

（6）为分析（建立什么）和设计（怎样建立）而使用了一种一致的、强有力的基本表达式。

（7）调整系统的家族以及实际中的权衡。

作为一个总的方法，OOA由五个主要步骤组成：（1）确定对象；（2）确定结构；（3）定义主题事务；（4）定义属性（及实例之间的联系）；（5）定义处理（及信息的联系）。

一旦建立起了模型，就可以在主题层、对象层、结构层、属性层和处理层这五个主要的层次上进行表示和检查。

（1）主题层：主题是控制一个读者一次考虑一个模型的多少的机制。

（2）对象层：对象是数据的一种抽象以及该数据上的专用处理；反映了系统保留现实世界中某事的有关信息并实现与之交互的能力。

（3）结构层：结构表示一个问题空间的复杂性。分类结构描述了类成员的组织，反映了一般到特殊的映射。组装结构表明了聚合映射，反映了整体和组成部分之间的关系。

（4）属性层：属性是一个数据元素用来描述一个对象或分类结构的实例。

（5）处理层：一个处理是指对收到的信息进行加工处理。基本策略集中于："事件发生"、"计算"和"监控"。

对OOA而言，为了能由分析顺利地过渡到设计，在分析和设计中使用了相同的基本表示，这种概念成为了OOA方法的基础。为了增加分析的稳定性，OOA方法使用了信息隐藏，将易变的部分隐藏起来；另外，对象的定义以及对象专用的处理使变化的影响更容易确定、约束、跟踪、和估计。

OOA和其它方法的区别对照如下：

方法	原则					
	抽象		信息隐藏	分类及继承性	组织的方法	行为分类
	过程	数据				
功能分解	*					
数据流	*					*
信息模型（仅数据）				*	*	
面向对象	*	*	*	*	*	*

四、后记

OOA方法是一个相对年轻的方法，在实践中会继续进一步地发展。下面是一些进一步发展的可能方向：

（1）用OOA来描述OOA自已。

（2）允许属性也成为对象本身。

（3）为OOD（面向对象的开发）加上风险估价、分析和管理的特定指导原则。

（4）将OOA和OOD作为一个完整方法的两个方面。

参考文献

[Cox, 1986]Cox, Brad, Object-Oriented Programming. Addison-Wesely, 1986.

[Meyer, 1988]Meyer, Bertrand, Object-Oriented Software Construction. Prentice Hall, 1988.

[Smalltalk, 1986]Smalltalk/v Tutorial and Handbook. Digitalk, Inc., 1986.