结构化分析与面向对象分析

D. de Champeaux 等

1990年10月21—25日在加拿大渥太华市召开了OOPSLA（面向对象的程序设计：系统语言和应用大会）与ECOOP（欧洲面向对象程序设计大会）的联合会议。会上有一些著名的计算机软件科学家，如E. Yourdon、L. Constantine等在一个关于结构分析和面向对象分析的小组讨论中，发表了一些观点，值得注意。

——译校者

Dennis de Champeaux

面向对象的风范仍面临公开的挑战：常规且费用低廉地实现庞大的软件系统。引用Ed Yourdon的话——"现在，我们对一个用C++语言编写的100，000行代码的系统不可掉以轻心，但若用COBOL写一个100，000行的系统我们都不会有什么麻烦。只有在一个具有一百万～一千万行代码的系统需要开发时，面向对象的程序设计才会得到真正的检验。"尽管面向对象研究界通过继承性开拓软件重用，对探索性程序设计和快速原型建造做了一些工作，但是我认为如果我们对目标系统将要完成什么工作不予以充分考虑，就无法真正地开发一个庞大系统。应当考虑的是产生一个（电子的）（图式的）（伪形式化的）文档，即需求，让顾客签字认可。我们同样认为：对于一个庞大系统，需要一种独立于设计活动的一个编程语言，它能在需求分析和实际的编程工作之间架起桥梁。显然我们并不认为这些活动应组织成一个瀑布模型。

所以面向对象研究界需要讲讲这样一个问题：那些早已确立的分析技术，象SA，JSD等，能否被重用到OO系统的开发过程中，或者是不是需要一种专门的OO分析（及设计）方法。更确切地讲，如下所述：

- SA能否有效地用来产生一个系统的需求分析，而这些需求分析将被我们以OO的方式进行设计和实现？
- 若不行，是否有可能调整SA，那么又需要添加什么内容？若SA根本不能运用，主要障碍是什么？
- 假若SA和OOA有不同的应用范围，我们如何从正、反两方面对这些范围进行划分？是否有重叠？

我们很高兴看到大会组织委员会选取了这个重要的题目。

Larry Constantine

对于传统的结构化方法，问题或应用模型和求解该问题的软件模型之间存在很大差别，并且各自以完全不同的记法来表达。采用OO组织方式，就有可能使设计的软件模型与应用的结构模型更加贴近。至少从原则上讲，用OOA生成的模型与用OOD生成的模型都能用基于共同原理的相同或等价的记法来表示。

传统的结构化分析能够且已被成功地用在OOD和程序设计的"前端"，但所得到的模型对进一步开发面向对象的设计模型所起的作用是有限的。经验告诉我们：为达到上述目的，基于实体-联系模型及其拓广的一些方法要比数据流模型好一些。随着大的应用

项目的开发，就更加迫切地需要一些专用的面向对象分析模型及方法。

迭代的，探索性的，或原型化的方法进一步使OOA和OOD的界线变得模糊。然而比SA和SD耦合得更紧的OOA和OOD并不是等同的两个活动。从管理的观点来看，甚至希望人为地拉大它们的差别，以便增加OO软件开发过程的可控制性。

Ivao Jacobson

软件开发过程中最常用到的两种风范是：函数数据方法和OO方法。

函数数据方法用两个概念来为系统建立模型。说得粗略点，函数是主动的，并具有行为，而数据是被动的信息接纳者并被函数所操纵。函数数据方法通过程序和数据为传统的计算机系统的行为抽象地建模。许多传统的软件工程方法，如SA，JSD，SREM(RD-D)，SADT都是函数数据方法。

面向对象方法(也包括基于对象的技术)仅以对象为基础为一个系统建模。这些对象向周围世界提供服务(行为)，同时自身还包含信息。现实世界中的实体被直接映射到模型世界的对象上去。这和函数数据方法不同，因为后者是把现实世界中的实体映射到函数、数据这两种结构上。

两种方法都在实际中运用了许多年，就算25年吧，尤其在数据处理系统中，有更多人采用函数数据方法。OO方法最初是用于象电讯系统这种庞大而复杂的应用中，但现在看来，五年或十年之内，OO方法将会在技术应用和信息系统中占主导地位。

我的观点简单概括如下：

1)应尽可能早一点把OO方法引入到系统的生命周期里。所以，应该把OO技术用于复杂系统的建模，系统分析，系统设计及程序设计中。从一个阶段所用风范到另一阶段所用风范的过渡非常复杂，这需要对这两个不同的风范进行培训。开发者必须把一组模型概念手工地翻译到到另一组。

2)假定人们有了一个基于函数数据方法的分析模型，有一种方法把这一模型手工地翻译成OO模型。这要比从最初的需求规格说明开始着手来得快。我们建议这种翻译应由哪些分析早期模型的人员来完成。

3)用于SA的一些图表技术对OOA亦适用。状态转换图有助于一些对象类型的建模。数据流图在较低层次上可用于复杂对象的建模。

Stephen Mellor

SA和OOA的关系主要体现在二者的定义上。从分析方法和所得到的文档的框架结构出发，SA里"结构化"可定义为"有组织的"和"系统化的"。功能分解(传统上被认为就是SA)，事件-响应方法(McMenamin and Palmer, Word/Mellor)和OOA，它们或多或少都被看成是结构化分析这一家族中的成员。

功能分解和事件-响应均不能运用OO方式有效地构造一个系统，因为前者的分析是通过对处理过程的检验，而后者的分析是通过对事件的考查。这两种方法中的数据均是为了支持数据处理而组织的。另一方面，抽象数据类型是OOA的基础：首先分析员找到概念性的实体，然后描述它们的行为，最后才推导出处理过程。

然而，另一些分析方法均不如OOA有效。功能分解根本无法涉及某个问题的语义。事件-响应方法是为了解决语义问题而确切定义的一种方法，但它通过系统的外观视图来解决这个问题，这更适合于规格说明(其它用户所看到的)，而不太适合分析(世界实际上是怎样的)。因此，我们要问的应该是：将OOA用于传统的系统设计是否会碰到障碍，而不是有何优势。

一种分析技术的应用范围可以按应用的技术划分成两类：问题类型及设计类型。所有的系统均有语义——即使是功能化程度极高的系统，所以OOA这种采用语义驱动的方法是适用的。

设计就是对一个系统中的数据组织、控

制、算法所遵循的规则进行选择，它是独立于系统分析和程序设计语言的。没有一种分析方法需要与预先选定的设计方法的一些概念进行对应，面向对象的设计也不例外。这一点特别适用于继承性，继承性只在现实世界的一定限度或范围内存在。OOA完全满足下述要求：

——用概念化的实体精确地定义问题的语义；

——基于对现实世界的抽象；

——允许设计者选用最合适的继承结构。

Paul Ward

在系统开发过程中，SA是一种折衷方法。它已经逐渐发展到能和不同的记法及不同的模型建造的启发方法相适应。在附加上一些OO建模原则后，借助对象的类和其继承性质，SA能有效地用于表示规格说明。这种规格说明能直接转换到OO设计。

SA的建模记法，最初主要限于数据流图，现已发展到包括实体-联系图及拓广方法，也包括图式的和表格式的状态机表示。这一组记法能够表现OO规格说明模型中的众多视图。一个两级的数据流模型能够体现相互作用的对象，每个对象上的操作和被封装的数据(示例变量)。实体-联系模型能够体现问题领域中的对象类及其相互关系。状态图或表可以描述一个对象的生命周期。

SA的模型划分原则，起初仅限于(对"逻辑"模型来讲)功能处理的划分和即席数据的划分，现已进化到能够概括各种可能的划分，如事件-响应过程的划分和基于对象的数据的划分。如果响应能分解成使用单个数据对象的部件，则事件-响应模型就能划分成相互作用的对象。如果实体-联系模型标识出父类对象，那么在数据流模型中，可以把相关的父类处理(操作)嵌入到子类中。

采用SA的数据流模型这种基本的自顶向下的结构特征最初是和开发模型过程中的功能分解的策略紧密联系在一起的。近来出现这样一种方法，它最后得到的模型是自顶向下组织的，而在此之前是采用一种粒度更适中"(moddle-out)"的模型构造策略，这种方法的初步策略是对象的识别，然后在组合对象的基础上进行自顶向下的包装。

当前许多CASE产品实现了各种各样的SA记法，如果加上刚才所提及的指导原则，那么这些产品就可以用来产生面向对象的规格说明模型。

Edward Yourdon

我们被要求考虑一些有关SA和OOA之关系的问题。我对这些问题的看法是：

1.能否有效地运用SA来为系统产生需求分析，且这个需求分析将以一种OO方式来设计和实现。我认为问题的关键是"有效"二字。从这一角度看，答案是否定的。本来从结构化分析转换到结构化设计已经够困难了，从SA到OOD/OOP的转换更是难上加难。原因是两种模型的记法非常不同，再加之SA中所强调的(功能)和OOD/OOP所强调的(对象)又大不一样。虽然从SA到OOD的转换可以用Brute-force方法，但这不是一种天衣无缝的结合。

2.如果不行，能否调整SA，那么又需添加什么内容？如果SA根本就不能用，主要障碍又是什么？一些人声称已经能把SA和OOA结合起来，他们运用了McMenamin和Palwer在Essential System Analysis中所描述的"事件-划分"方法。这种方法是将最初通过数据流图得到的需求模型"对象化"，它使得从SA到OOA的转换成为可能。然而我认为"经典的"结构化方法存在三个问题，使之不适于用OO方式来实现系统：(1)过程、数据和时间依赖性这三者的记法分别是DFD(数据流图)，ERD(实体-关系图)和STD(同步时间图)，是彼此相互独立的，在现实工程世界中，记法中的某一种(常为DFD)通常占主要地位，并排除别的记法。(2)因而微妙地影响到软件工程师把每个系统开发项目看成是一种"原则优先设计"(design from

信念修改的理论与方法

黄智生 （荷兰阿姆斯特丹大学计算机科学系留学生）

摘 要

目前非单调推理、知识与信念逻辑(亦称关于知识的推理)和信念修改理论已成为人工智能理论研究中的三大新的热门课题。近半年来，笔者参加了国际上先后召开的四次重要的人工智能理论学术会议：JELIA'90(人工智能逻辑，荷兰阿姆斯特丹，1990年9月)、CSL'90(德国海德堡，1990年10月)、MEDLAR'91(英国伦敦，1991年3月)和KR'91(美国波士顿，1991年4月)。本文将简要介绍有关信念修改的基本概念、理论和方法，也介绍了这一理论研究的进一步发展梗概。

在人工智能理论研究中，目前非单调推理，知识与信念逻辑(又称关于知识的推理)和信念修改理论已成为三大新的热门课题，而且各有自己对应的专门化国际会议以促进其研究。对于非单调推理，有每两年一度的非单调推理国际会议(Workshop on Nonmonotonic Reasoning)。对于知识和信念逻辑，有关于知识的推理理论问题国际会议(Theoretical Aspects of Reasoning about Knowledge)。对于信念修改理论，有理论变化的逻辑国际会议(Workshop on the Logic of Theory Change)。当然，上述三个领域并非是各自独立的，它们之间具有很强的联系，但各自又有不同的侧重面。本文将简要地介绍有关信念修改的基本概念，理论与方法。

一、知识与信念

在目前的人工智能理论研究中，“知识”和“信念”是极为相近又有一定区别的两个重要概念。对于存贮在数据库和知识库中的信息，有时我们称它们为知识，有时称它们为信念，这取决于不同的情况。一般地说，知识是指那些为多数人所接受，具有较高可信度的又可用于推理的信息式数据；而信念是指那些仅为个人所相信的，为个人推理所用的，相对不稳定的信息式数据。在知识库管理中，当外来的新信息与已有存贮在知识库中的信息相矛盾的时候，我们就需要对原有的知识库进行修改和更新，而对于研究如何进行这种修改和管理的理论，我们现在习惯地称之为信念修改理论(Theory of Belief

---

first principles”，而不看成是异常情况设计”(design by exception)的OO风范。(3)SA记法不利于人机接口模型的构造，而这一模型在GUI(图形用户界面)环境中正变得越来越重要。

?.假若SA和OOA的范围不同，我们如何从正反两方面来划分这些范围，这种划分是否重叠？我认为这是一个文化问题而不是一个技术问题。即使从技术上看OOA比SA应用更广，许多数据处理组织对于这种改变仍有太多的惰性。只有当旧的风范不能解决该组织的问题时，新的风范才会被普遍接受。因此我感到只有在SA不能胜任的环境中，OOA才得到充分的承认，这些环境可能是一些庞大的，可视的工程，也可以是那些从一开始就把着眼点放在重用及图形用户接口的工程项目。

〔刘晓丹译 自《SIGPLAN Notices》25(1990).№.10 王振宇校〕