从面向对象视角认识基于模型的系统工程

陈红涛

中国航天系统科学与工程研究院, 北京 100048

摘要 软件工程经历了从面向过程到面向对象的转变,实践证明了这种转变的成功。相对软件工程更为一般化的系统工程,也遇到了系统日益复杂的问题。基于模型的系统工程(MBSE)是对建模(活动)的形式化应用,以便支持系统要求、设计、分析、验证和确认等活动,这些活动从概念设计阶段开始,持续贯穿到设计开发及其后来的所有寿命周期阶段。MBSE采用从统一建模语言(UML)发展而来的系统建模语言(SysML)构建系统模型,其本质是面向对象的系统工程(OOSE),也将获得类似面向对象的软件工程(OOSWE)的优势,而且MBSE的工作流程,应以面向对象为指导原则来探索和实践。

关键词 基于模型的系统工程;系统建模语言;面向对象的系统工程

软件工程中存在面向过程(process-oriented)和面向对象(object-oriented)两种思路。我们又知道,软件工程是系统工程的特例,SysML(systems modeling language)又源自于UML(unified modeling language),系统工程是否也存在面向过程和面向对象之分?从传统的系统工程(traditional systems engineering,TSE)发展到基于SysML的MBSE(model-based systems engineering),是否也是系统工程思路的重大转变——从面向过程到面向对象?MBSE又能否借鉴面向对象带来的优势,去处理日益复杂的工程系统?

1 软件工程从面向过程发展到面向对象的启示

软件工程已经从面向过程发展,转变到了面向对象,那么,面向过程和面向对象各自的本质是什么?这种转变又有什么意义?

1.1 面向过程的特点

传统的软件工程是面向过程的,就是把准备求解的问题按照算法进行分解[1],从程序分解到子程序、函数、子函数,直至计算机自带指令集中的最基本指令。例如求解sin5,就要展开为泰勒级数,再分解为加减乘除的计算,再把这些分解后的运算步骤映射为计算机的指令,如取数、加和、存储等,然后由软件(此时即一系列指令组合)指挥着计算机硬件完成相应的各种操作。这与计算机诞生之初主要用于科学计算有关。

1.1.1 面向过程意味着状态序列分解

过程的本质是一组连续的状态序列,状态是对象在某个时间点上的特征取值的组合,例如物体的运动过程,是其质点坐标、姿态等特征取值的变化。状态序列类似拍电影时的1 s有24个画面,1个画面对应于拍摄对象的1个状态。对过程的分解、不断细分,即状态序列分解,也即状态之间的时间间隔不断缩小。

在计算机软件的解题过程(如上述计算sin5的值)

收稿日期:2018-08-31;修回日期:2019-03-07

作者简介:陈红涛,高级工程师,研究方向为系统工程、基于模型的系统工程、SysML,电子信箱: cht7855@sina.com

引用格式:陈红涛. 从面向对象视角认识基于模型的系统工程[J]. 科技导报, 2019, 37(7): 36-43; doi: 10.3981/j.issn.1000-7857.2019.07.005

中,计算机硬件的初始状态与输入值对应,目标状态与计算结果对应,计算机的整个状态变化过程,对应着一步步的计算过程。也就是说,计算机的解题过程,就是计算机硬件的一系列连续状态变化过程。计算机软件就是根据人对解题过程的分析、设计,对计算机硬件的状态变化过程、状态序列进行描述,在指令层次,就是计算机的控制器发出对键盘、存储器、运算器等硬件的一系列指令,而一条指令,又对应着计算机硬件状态的一次状态切换。面向过程的软件开发思路,是把安装了软件的计算机当作一个系统,把它从数值输入到结果输出的整个计算和处理过程(也是整个状态变化过程)进行分解,得到更小的状态变化过程,直至这些小的状态变化过程和计算机所能够完成的状态变化过程相对应,即执行一条指令,计算机的状态发生一次切换。因此,面向过程的软件开发也就是软件开发中的算法分解。

1.1.2　数据和操作相分离

面向过程的思路把程序分为数据和操纵处理数据的过程(函数)两部分,软件体现为算法+数据结构两大部分,并且以算法为中心(往往以算法是否高效来衡量程序员水平的高低)。这种思路和做法,把数据和处理数据的过程(函数)分开,相当于把对象的属性(状态)与状态变化(行为)强行地分开,这显然有悖常理。因为有什么样的属性,才会有什么样的状态变化,状态变化是存在路径依赖的,是在前一时刻状态的基础上发生的。任何一个对象、事物都是其静态属性和动态属性的统一,是结构决定功能,是慢变过程决定功能这一快变过程。把数据和对数据的过程(函数)在逻辑上分开与割裂,就容易造成整个设计思维的割裂。而在面向对象的软件开发中,不存在数据处理的概念,没有函数的概念,只有对象的调用及对象的状态变化。

1.1.3　问题描述和解决方案描述语言之间的巨大差别

计算机软件所要解决的现实世界中的问题、难题,例如大型银行的信息系统,包括了各种终端、数据库、服务器、单机等,每天处理天文数量级的业务,并与各种外部公司的信息系统交互数据,如证券公司、保险公司、业务伙伴、第三方支付软件等,信息系统的复杂度非常高。而且,问题往往用自然语言提出、思考并记录,并不是用计算机的编程语言。自然语言是主语、谓语、宾语都全的语言,正如姚振武在《人类语言的起源与古代汉语的语言学意义》中所说:最初的思维形式、语言形式、逻辑形式是三位一体的:其底层就是“本体—属性”的概念,其语言表达就是“指称—陈述”的分化(有时按一般的习惯,粗略地称为“名词—动词”)[2]。而编程语言是无主语,只有谓语和宾语的语言[3]。上文所说的软件开发的过程分解、算法,是计算机动作的分解,也就是默认计算机是这些动作、功能、指令的主语,过程分解就是对动词进行分解。而当用计算机来模拟现实世界时,现实世界有很多的“主语”,它们有自己的状态变化规律和行为规律,例如银行信息系统外部的各种企业用户、个人用户等。在开发软件时,就不得不在计算机语言和描述现实世界的自然语言之间进行繁琐的转换。随着软件系统的日益复杂,这种转换是软件开发难度激增的重要原因。

1.2　面向过程方法面临的挑战

1.2.1　系统越来越复杂

现实世界中对象的行为丰富多彩,而计算机本身的行为、动作只有数百种,即计算机自身的指令集(如精简指令集计算机,reduced instruction set computer,RISC),因此,必须借助复杂的算法,把现实世界中对象的行为(例如银行系统中各类主体的行为)变换到计算机指令的组合,这个难度可想而知。

1.2.2　难以应对用户的需求蠕变

软件用户的需求即对应着计算机系统状态变化过程中一头一尾2个状态(例如A状态变化到B状态),当需求变化时(可能是A状态变化到C状态),即状态序列的两端发生了变化,那么功能分析、状态序列分解的结果也会变化,相应的软件功能模块也会发生较大的变化。面向过程的软件开发中,系统有明确的边界,因为只有先确定系统的范围和边界,才能明确并不断分解系统的状态序列,使计算机的基本指令靠拢。因此,面向过程思路受初始状态、最终状态,也即用户需求(对应着从初始状态变化到最终状态)的影响很大,这种思路开发出的软件,不易扩充和修改,难以应对用户需求蠕变。

1.2.3　人们对复杂事物的认识必然要经历一个不断深入的过程

面向过程的软件代码是对数据结构中各个数据进行加工处理的,要首先确定数据结构,而且数据结构的不同维度之间还存在一定的耦合关系。而一旦数据结构改变,有关的操作乃至整个程序都需要重新组织和设计,代码就要重新编写。数据结构的实质是事物的

特征组合(例如学生信息系统中,关注学生的年龄、学号、成绩等特征),也就是说,数据结构的变化意味着数据结构所描述的对象不是原来那类对象(例如学生信息系统中,所关注的特征变化后,很可能不再是在校学生而是往届校友),那么对象的变化方式(代码、函数)自然也不同。

同时,事物可能很庞大、很复杂,其特征值的维度也可能很多,面向过程的设计思想,试图一下子就完全把握对象的所有方面的特征,这似乎不太现实。因为人们对这些事物的认识是循序渐进、逐渐深入的,今天发现了事物的一个特征,明天又要新增某个需要关注的特征,这也是需求、问题的认识过程。

面向对象的思路反其道而行之,就是承认我们对事物、对象的"无知",慢慢从"无知"走向"有知",就是逐步地明确对象的特征及对象的操作、变化方式。并且随着我们对事物认识的深入,而不断地给事物增加特征、增加特征的变化方式,这个过程就是认识不断深入的过程,也是逐步特化(specialization)的过程。

1.3 面向对象和面向过程的对比

面向对象的软件工程是先从编程语言开始的,如C++、Java,直至UML。面向对象的软件开发,采取数据结构+算法的方式,从以前的算法为中心,转变为数据结构为中心。其主要特点是:

1) 对象=数据结构+算法;

2) 程序=(对象+对象+…)+消息;

3) 消息的作用就是对对象的控制。

orient的含义是"把兴趣对着……、集中在……",因此,"object-oriented"的意思是"以对象导向,把兴趣集中在对象上"。以对象为主要的兴趣点,并不是说不再关注过程,而是以对象为关注点、为分析思路,然后用对象的动作、操作,来组成系统的变化过程,让系统发生各种各样的状态变化,使系统能够满足用户的需要。正如著名的UML专家Grady Booch在《面向对象分析与设计》一书中所说:"如果过程和函数是动词,数据是名词,那么面向过程语言的程序就是围绕动词组织的,面向对象的程序就是围绕名词组织的"[4]。

在软件开发时,可以把软件所要描述、计算的现实世界中的事物、实体(如学生信息系统中的每一个学生)看成是对象,并与计算机中的实体(内存、硬盘上的一段代码,也即描述对象的一段代码)进行对应,现实世界中的对象叫做问题空间的对象,计算机中的实体叫做解空间的对象。

1.3.1 弥补了问题空间和解空间之间的鸿沟

软件开发中描述问题的自然语言和解决问题的编程语言之间,存在着巨大的语言鸿沟,这个鸿沟,表面上看是符号的区别(一个是自然语言,一个是程序语言),实质上是概念上的鸿沟:前者是现实世界中某一个问题领域的概念,后者是解空间即计算机中的概念,如计算机的指令、内存等。

人们不断发明新的编程语言缩小人机沟通语言之间的鸿沟,先是二进制的机器语言(要计算机服从指挥、帮助人们做事,就必须用计算机能够"听懂"的语言),在此之上又发明了汇编语言、高级语言,以此来填补自然语言和机器语言之间的"语言鸿沟"[1]。面向对象的软件工程的语言变成Java、C++或者UML,使得人机沟通语言之间的鸿沟日趋缩小,但核心是思路也即头脑中组织概念的方式更加一致:都是一系列的对象及对象之间通过消息实现控制。

1.3.2 弥补了需求分析和设计的鸿沟

软件不能真正满足用户需要,主要源于两个方面:一是不能彻底理解用户需求,二是用户需求在不断的变化,即需求蠕变。用户为什么要改需求?因为用户对自己的需求也有一个认识的过程,同时用户本身所处的环境(如交易对手、外部接口环境等)也在不断地变化,因此,软件开发过程中,需求蠕变是不可避免的。

面向对象的软件工程中,需求方和研制方使用同一种语言——UML描述,可以有效弥补需求和设计的鸿沟。采用业务建模、用例等方法,进而进行系统用例的建模,这是用户和设计者双方都能够理解的语言和建模方式,便于双方进行沟通。

面向对象的软件工程,相对于面向过程的软件工程有诸多优势,尤其是在复杂软件开发中。

2 面向对象方法学及其对工程系统设计思路的启示

2.1 从面向对象的软件工程到更为抽象的面向对象方法学

面向对象的软件工程与面向过程的软件工程,在编程语言、建模语言上的区别是表面上的,核心是思维方式的转变,也即头脑中组织问题域和解域的相关概念的方式,是以实体为主线还是以过程为主线。

2.1.1　面向对象的软件工程的启示:对象分解而不是过程分解

客观世界是由许多各种各样的对象所组成的,每种对象都有各自的内部状态和运动规律,不同对象间的相互作用和联系就构成了各种不同的系统,构成了人类所面对的客观世界。把客观世界中的一个实体抽象为问题域中的对象(object)。对象具有状态,一般用数据值描述对象的状态。从动态观点看,对对象施加的操作就是该对象的行为。通常,客观世界中的实体既具有静态的属性又具有动态的行为[5]。

从动态角度或对象的实现机制来看,对象是一台自动机,具有内部状态S,操作$f_i(i=1,2,\cdots,n)$,且与操作f_i对应的状态转换函数为$g_i(i=1,2,\cdots,n)$。转换是不及物动词,操作是及物动词,对输入的X进行加工,输出是$f_i(X,S)$,如图1所示。此时的输入相当于是对对象的操作f_i进行了调用,同时,对象的状态由S转换成S'。例如,碎纸机处于"待机状态"S,输入是一张A4纸,同时调用了碎纸机的操作"粉碎纸张",输出是碎纸屑。

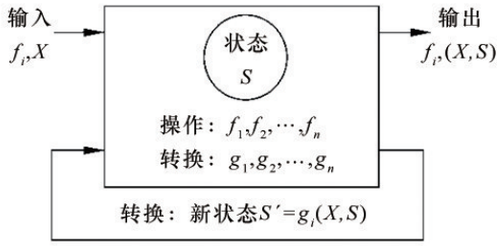

图1　用自动机模拟对象

2.1.2　面向对象方法学的提出

对象是一种普遍适用的基本逻辑结构,是一个以有组织的形式含有信息的实体。它既可以表示一个抽象的概念,也可以表示一个具体的模块;既可以表示软件,也可以表示硬件。因此,面向对象的方法学既提供了一个分析、设计和实现系统的统一方法,又提供了描述、设计和实现硬件和软件系统的统一框架。

面向对象方法学尽可能地模拟人的思维方式,并且和方法学相适应的面向对象技术也提供了这样的可能性:即可以随着对某个系统的需求逐步具体的过程而逐步地设计和实现这个系统。

面向对象方法学在概念和表示方法上的一致性,保证了在各开发活动之间直接地平滑过渡。面向对象的软件工程中,分析、设计、编码等开发活动之间并不存在明显的边界。分析与设计之间要多次迭代,这就意味着两个团队的人员要不断地交流。交流就要用某种语言,语言就要包含相应的概念,语言就是对概念的组织,那么按照对象分解的思路进行概念组织,还是按照功能分解的思路进行分解和组织,会极大地影响交流的效果:按照对象分解思路,使得各方更容易沟通。

2.1.3　从认识论层次看面向对象方法学

恩格斯在《路德维希·费尔巴哈和德国古典哲学的终结》中指出:"必须先研究事物,而后才能研究过程。必须先知道一个事物是什么,而后才能觉察这个事物中所发生的变化"。因此,在认识和研究客观世界时应从"对象"入手,然后再转向"过程"。经验也告诉我们,在客观世界以及作为它的映射的软件系统中,"过程"和"操作"是不稳定的、多变的,而"对象"和"数据结构"则相对稳定得多。因此,如以"过程"入手(或以"过程"为中心)设计软件,思维成果的可重用性必然较差;而以"对象"或"数据结构"入手(或以"对象"或"数据结构"为中心),则软件的主体结构比较稳定,其所得的思维成果的可重用性就有可能较好。但又必须考虑到对进一步转向研究"过程"的可能和方便[6]。

《大英百科全书》描述了分类学理论中有关人类认识现实世界所普遍采用的3个构造法则:区分对象及其属性;区分整体对象及其组成部分;形成并区分不同对象的类。面向对象思想正是根据以上3个常用的构造法而建立起来的。在实际应用中,它采用对象及其属性,整体和部分,类、成员和它们之间的区别等3个法则来对系统进行分析和设计,遵循了分类学理论的基本原理[7]。

2.2　面向对象方法学对工程系统设计思路的启示

从软件(计算机系统)开发这个特殊,走到一般化的工程系统研制,看软件开发中的面向对象能不能为工程系统开发所用?

在面向对象的软件工程中,人们采用面向对象的思路对问题空间中的客体进行抽象,软件、计算机只负责记录、处理这些对象的属性信息,并不实际改变它们的属性。而工程系统包括处理物质能量的部分和处理信息数据的部分,后者的代表是计算机系统。工程系统的设计,最重要的是搞清楚系统的零部件之间是如何相互作用的,如何通过零部件的相互作用实现系统的功能,所以就产生一个思路:能不能把系统的零部件也当作对象,在计算机建模环境中模拟出零部件的所

有动作(即属性变化),并且让这些动作的组合能够实现系统功能。最后,再把这些已经知道了其属性变化的零部件实现出来(车间加工、市场上直接采购等)。

2.2.1 问题空间和解决方案空间的同构

工程师要想研制出工程系统去要解决用户所面临的问题,就需要从问题领域跨越到解决方案领域。2个领域有不同的知识(概念及其关系),2个领域的人相互理解比较困难。

问题空间中存在的是对象。问题空间的本质是对象的状态不符合人的需求,人们对这个状态不满意,因此解决问题的实质就是使对象的状态从不满意变化到满意。无论是工程系统开发,还是软件开发(计算机系统开发),都是要解决人们在现实世界中的问题,也就是帮助人们从目前的状态(不满意状态)走向目标状态(满意状态)。人们要解决现实世界中的问题,就必须先对这个问题进行认识和定义。通常,人们对现实世界的认识方式都是面向对象的,也就是先看到物,再看到物的变化所形成的过程、物的行为,也就是人们认识世界的方式是面向对象的。对计算机计算sin5的值来说,是从输入值对应的状态(按下键盘的"5"这个键),变化到输出结果对应的状态(屏幕上显示出结果),这2个状态在内存中有分别的对应。对碎纸机来说,是从整张纸这种不满意的状态,变化到"粉碎的"这种满意的状态。因此,问题空间必然对应着一组对象,每个对象当然可以再分解为小的对象。

解决方案空间中存在的当然也是对象,如零部件、各种商用现货等。解决方案就是通过对解空间中对象的操作、让解空间中的对象之间相互作用来解决问题。

既然问题空间和解空间中存在的都是对象,而面向过程的开发方法(如上文所说的面向过程的软件开发),却采用面向过程的思路进行从问题空间到解空间的映射,例如进行功能分解得到功能架构、再由功能架构转换到物理架构等,这是一种思路上的扭曲,造成了问题空间和解空间无法同构(相同的分解结构),无法按照面向对象思路进行层层分解。只有采用面向对象的思路,才能实现从问题空间向解空间的映射同构。因此,在分析问题、分析需求时,设计解决方案时,应坚持以对象为中心的思路,即面向对象的思路,以便实现问题空间和解空间的同构。

2.2.2 各方人员之间的相互理解沟通变得容易

无论是开发软件还是开发导弹,从用户需求到系统架构,到设计方案到实现过程和维护过程,实际是人的思维成果的转化,是站在不同视角的人,对系统进行认识而得到的不同视图,是不同视图在不同人之间进行传递,而思维成果必然以某种语言表示,而且要具体落实到某种建模载体(纸张、实物模型或计算机上的图示、图形等)上,以便整个团队进行交流、沟通。如果从问题空间到解决方案空间,用户、系统架构师、设计师都采用面向对象思路工作(当然要借助类似UML、SysML这样的语言及相应的软件),沟通会非常顺畅。

3 MBSE是面向对象的,而TSE是面向过程的

在对比了软件工程中的面向过程和面向对象,看更一般的系统工程中的面向过程和面向对象。

3.1 TSE是面向过程的

目前把型号研制中正在使用的系统工程称作TSE,它是面向过程的。就是把工程系统在运行中从初始状态到最终状态的变化过程,进行层层的状态序列细分,直至简单的状态变化,并且可以根据该状态变化找到合适的硬件来实现(例如分解到控制电路闭合的电源开关)。这个分解过程,与软件工程中把计算机作为一个整体进行状态序列分解,是同一个思路。

从历史来看,系统工程从诞生之日起,就一直在采用面向过程、功能导向的设计方法,和软件工程的结构化编程及功能分解的方法是一样的[8]。例如,1969年的Mil-std 499及后续的Mil-std 499A、Mil-std 499B均提出进行功能分析,其中499B明确提出了系统工程过程(其中的第2步是功能分解与分配)。直至1998年发布的EIA 632,才提出在进行"逻辑解决方案表示"时,可以用功能分析方法,也可以用面向对象的分析方法。此时,软件工程界的面向对象已经得到广泛应用(UML在1997年11月发布了1.1版)。

TSE把目的系统当作一个大的过程,对应着一头一尾2个状态,那么中间的状态都是什么,就需要面向过程的分解把这些状态找出来。此时把系统当作一个有输入和输出的过程,每一对输入输出,对应一头一尾2个系统状态,这2个状态又对应输入物的状态,如ATM机的输入物是银行卡、密码信息,输出物是卡片、钞票、打印凭条、提示信息等。

功能分解类似写报告的分级标题,层层分解。比

如第1层分成1.0、2.0、3.0、4.0,每一个相当于一个大的子过程、一个大的步骤,例如ATM机的2.0步骤,就是“接受并发送操作申请”。但是这一步骤仍然在物理上无法直接实现,那就对2.0继续分解,分解到2.1、2.2、2.3、2.4、2.5这5个步骤(图2),然后看这几个步骤能不能直接找到硬件完成,如果不能,则继续分解,直到最底层,比如分解到了2.1.1.1、2.1.1.2这一层次(可形象地称为是4级标题),这些动作就比较简单,可以交给简单的零部件完成。例如最简单的情形,2.1.1.1就是一个开关动作,对应着开和关2种状态(从开到关,也构成了一个状态序列),那么,就可以交给一个现成的零部件——开关(switch)实现;再如,向服务器发送账户信息,则可以交给天线完成。

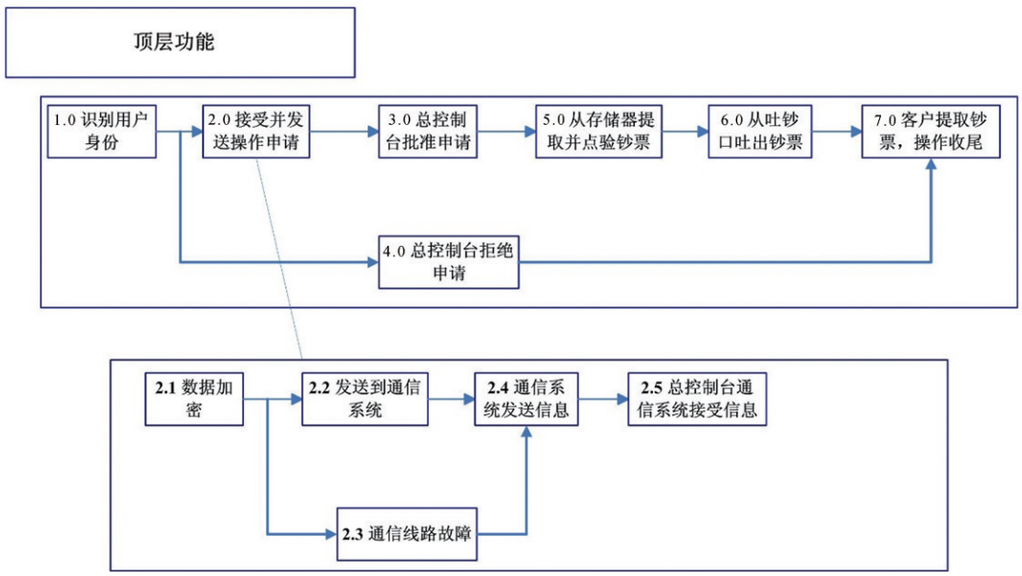

图2　自动取款机的功能分解示意图(功能流方框图FFBD)

在分解的过程中,1.0、1.1、1.1.1等这些层次,实质上是没有内容的,只是不同层次的一个概括,最终内容都在最底层的动作。比如1.1.1,分解成1.1.1.1~1.1.1.7这7个动作,这7个动作完成了则1.1.1的动作也就完成了。然后再向上逐级地汇总,1.1.1~1.1.4的动作完成了,则1.1的动作也完成了。如此汇总,直至整个系统的动作的完成。

当系统开始运行时,首先执行1.1.1.1这个动作,其它的动作不执行,也就是对应1.1.1.1这个动作的零部件的状态发生变化,其它的零部件的状态不变化,那么从宏观上看整个系统,它的状态虽然只是在这个局部发生了变化,但是系统的状态仍然是发生了变化,在系统层面,也是2个状态,对外体现了系统的一个动作(完整功能的一小部分)。那么,当后续的1.1.1.2等一系列的动作执行时,系统的状态就一直在发生变化,这样,等系统把所有“4级标题”的动作执行完(例如最后一个是4.4.5.8),则系统就运行了一遍,完成了系统的功能。

这种面向过程的分解的弊端,类似前文所述面向过程的软件工程的弊端,不再赘述。

3.2　系统工程应该也可以从面向过程发展到面向对象

目前工程系统越来越复杂,主要是因为其外部用户众多、使用方式多变、软件密集等。计算机控制从本质上增加了系统的复杂性,这也是系统工程的重要关注点。在软件密集型、赛博物理系统中,软件所发挥的功能由原来的7%增长到70%[9];型号研制涉及的专业知识门类多,需要考虑的专业工程多;与新老系统的互操作性的要求和系统的可扩展性的要求,这些都是系统和项目日益复杂的原因。

以赛博物理系统的开发为例,越来越像是复杂软件的开发,可以借鉴复杂软件的思路开发复杂工程系统。此时,把复杂工程系统中除计算机之外的硬件,如发动机、伺服机构等当作型号计算机的外部设备。同

时,已经有了系统建模语言这种从面向对象的软件工程建模语言(UML)发展而来、针对系统(不再区分软硬件)的通用性语言,因此,面向对象的系统工程是可以实现的。

3.3 MBSE的实质是OOSE

系统工程作为"创造系统的方法和技术",作为复杂工程系统研制组织管理的技术,当然包括系统建模技术和系统建模的组织管理技术,因为管理的基础是沟通,复杂工程中沟通的基础是系统模型,系统模型的基础是系统建模技术,系统建模工作中包含着系统建模技术,即建模语言、建模思路和建模工具[10]。

MBSE是面向对象的系统工程(object oriented systems engineering,OOSE),而TSE是面向过程的系统工程(POSE),两者的区别,和OOSWE与POSWE的区别是相同的。TSE转向MBSE,关键是设计思路的转变,建模语言和工具从属于设计思路的转变。设计本身是一个思维过程、建模过程,而且是很多人(包括用户、分析者、设计者、实现者、试验者、维护者等)共同参与的团队思维过程。而设计思维又和设计语言密切相关,SysML则是这种面向对象的设计语言、建模语言,同时也有一系列支持SysML的建模环境(如NoMagic公司的Cameo System Modeler等)。

3.3.1 面向对象的系统工程的工作流程

面向对象的系统工程的思路,就是在工程系统的不同层次(例如,地空导弹属于包含了雷达、指挥控制系统、导弹在内的作战大系统的一部分),顺序地使用系统建模语言的8种图形,即需求图—用例图—参数图—块定义图—活动图或顺序图—内部块图—状态机图(图3)。

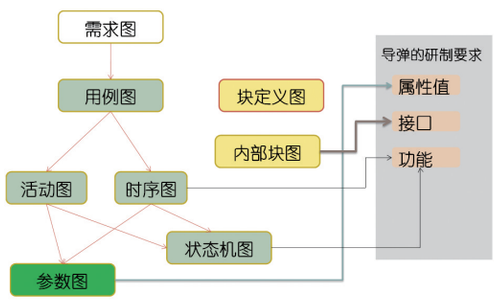

图3 以对象为中心的设计过程与SysML图形

第1步:明确用户对目标系统的要求,创建需求图。第2步:从用户对目标系统的要求中提取出用例,创建用例图。第3步:用参数图表示具体的要求的指标。需求图中定量指标,交给参数图。第4步:确定目标系统包含哪些分系统,用块定义图表示。第5步:用活动图或顺序图描述参与者和目标系统的一个完整的交互过程,把第二步所画的用例图具体化,变成用例场景。第6步:用内部块图把各个部件之间的连接关系以及各种流明确下来。第7步:对那些有状态机行为的部件再进一步说明。经过这7个步骤,完成了对要研制的对象(如地空导弹本身)的定义:包括属性和操作两大方面;属性就是对象的各方面的特征,当然也是建模者、软件开发者感兴趣的、与开发活动有关的特征。操作用于修改类的属性或执行某些动作,通常也称为功能。然后对该对象进行进一步的分解,继续使用上述步骤,直至得到货架产品或预研成熟或预计能够成熟的部件。

3.3.2 SysML的9种图形在描述对象时的关系

系统建模语言的9种图形,在面向对象的开发思路中发挥各自的作用,围绕着块定义图不断完善信息。块定义图主要显示工程系统的层次关系,显示对象本身的静态属性、动态属性;内部块图主要表示工程系统各个元素如何连接在一起,以及各个元素之间的各种物质流、能量流和信息流,各个元素的外部接口等;状态机图展示对象在各种刺激下的动作与状态变化,用来检查对象的操作是否完备;时序图则用生命线代表相应的对象,表示对象之间的消息传递;参数图以约束块来连接有关对象的值属性,并把对象所服从的相关物理规律、数值关系表示出来,如牛顿第二定律的相关数值关系。

3.4 从面向对象视角认识MBSE相对于TSE的优势

MBSE的优势实质是OOSE的优势,或者说MBSE的优势表面看是因为采用了系统建模语言,更深层次是面向对象相对于面向过程的优势。

3.4.1 便于复杂工程系统开发时的原始性创新

工程开发,例如开发飞机,必须从需求出发。目前,中国飞机工业设计水平低,从第一步的需求分析时就水平低,难以准确全面理解飞机用户的各种需求(如飞行员、空乘、各类乘客、飞机维护人员、后勤人员、机场空管、适航认证部门等),并且把这些需求传递到设计环节。采用MBSE方法,可以把上述各种需求主体(例如飞行员、空乘等)以及和飞机有连接和交互关系的元素都看成是对象,分析这些对象之间的各种信息和物质能量交互,以此来分析各类利益相关者对飞机的需求。

3.4.2　适宜处理软件密集型系统,实现软硬件的和谐开发

目前的型号研制,是先把系统的初步设计完成之后,才形成型号软件的需求说明书,一旦软件的开发中遇到某些不易克服的问题,则会对系统的总体方案形成巨大的冲击。MBSE在对系统进行逐级地分块时,并不区分软件和硬件,直至最底层才把逻辑部件分配给相应的硬件部件、软件部件和操作程序等,这样有助于系统功能在软件硬件之间的合理和优化分配,能够获得更加优化、平衡的系统模型。可以像使用UML对复杂软件建模、编程那样,使用SysML对整个系统(含软件)进行"编程"、调试、优化,直至非常完善之后再发图进行硬件生产,定下软件部分的规格进行软件编码实现。

3.4.3　需求分析和设计过程的可视化

和UML用于软件开发类似(程序员的思路可以通过他所建立的UML模型理解,而不再是一行一行的代码),用SysML对工程系统进行设计,这种设计思路同样是可视化、易理解的,并且便于沟通和知识复用。

其他方面的优势还包括:全系统仿真、需求蠕变的应对、便于研制单位的知识管理和知识复用,具体请参见《基于模型的系统工程的基本原理》一文[11]。

4　结论

软件工程的发展历程表明,OOSE是系统工程的发展趋势。系统工程从面向过程到面向对象,对于当前的复杂工程系统研制具有重大意义,这是战略层次的转变。相应在战术层次,MBSE的具体工作流程可以多种多样,但能够充分发挥面向对象方法学优势的,才是好的方法。

从面向对象角度看,MBSE的重大意义在于:把人类对现实世界的认知思路和人在进行工程系统研制开发时的建模思路、建模技术(语言、工具、方法)一致起来,从而解决人在认知、建模、分析、设计中的瓶颈,整体提升人类对现实世界的认知水平、认知速度,并进一步提升工程系统设计和研制水平。在中国研究借鉴MBSE、SysML这些方法论、语言及相关工具时,有必要从设计思路、设计语言等层次加深对MBSE的理解,这些相对抽象、枯燥的分析,有助于对MBSE原理、作用的认识,有助于更好地应用MBSE。

参考文献(References)

[1] 邵维忠, 杨芙清. 面向对象的系统分析[M]. 北京: 清华大学出版社, 2007.

[2] 姚振武. 人类语言的起源与古代汉语的语言学意义[J]. 语文研究, 2010(1): 6-20.

[3] 张孝详. Java就业培训[M]. 北京: 清华大学出版社, 2003.

[4] Grady B. 面向对象分析与设计[M]. 北京: 人民邮电出版社, 2009.

[5] 王崑声, 袁建华, 陈红涛, 等. 国外基于模型的系统工程方法研究与实践[J]. 中国航天, 2012(11): 52-60.

[6] 汪成为. 面向对象分析、设计与应用[M]. 北京: 国防工业出版社, 1992.

[7] 黄志坚. 工程系统概论: 系统论在工程技术中的应用[M]. 北京: 北京大学出版社, 2010:122.

[8] 陈红涛, 袁建华, 赵滟. 系统工程的昨天、今天和明天[C]//中国系统工程学会第18届学术年会, 合肥, 2014.

[9] Alexander Kossiakoff. Systems engineering principles and practice[M]. New Jersey: John Wiley & Sons, 2012.

[10] 陈红涛, 侯俊杰, 赵滟, 等. 工程系统工程的基本原理和未来发展[C/OL]//中国系统工程学会第19届学术年会, 北京, 2016. http://www.wanfangdata.com.cn/details/detail.do?_type=conference&id=8952468.

[11] 陈红涛, 邓昱晨, 袁建华, 等. 基于模型的系统工程的基本原理[J]. 中国航天, 2016, 32(3): 18-23.

Understanding model-based systems engineering from an object-oriented point of view

CHEN Hongtao

China Aerospace Academy of Systems Science and Engineering, Beijing 100048, China

Abstract　Software engineering has transformed from process-oriented thinking to object-oriented thinking, and this transformation has been proven successful by practice. Being a more general subject than software engineering, systems engineering is dealing with systems that are more and more complex. Since model-based systems engineering (MBSE) adopts systems modeling language, which has evolved from unified modeling language, MBSE is object-oriented systems engineering in nature, and will share the same advantages of object-oriented software Engineering. To develop and apply MBSE methodology, we should comply with the principle of object-oriented thingking.

Keywords　model-based systems engineering; systems modeling language; object-oriented systems engineering

(编辑　徐丽娇)