事件驱动编程和面向对象技术的软件开发范式

盛传捷 （上海交通大学国家模具 CAD 中心 200030）

引言

当前，我国正逐步形成市场经济体制，人才的流动相对以前频繁，因此一个软件的开发过程中，开发人员更换的情况并不罕见。为了保证开发工作的连续性，避免换一批人就不得不重头开始，严格的需求规格说明书.数据流图.数据字典.阶段性的计划和报告书等文档，就显得十分重要。软件投入使用后，维护阶段将一直延续至软件废弃不用为止，其间除了不断纠正程序的逻辑错误外，还可能会应用户的要求增加新的功能，这些工作更可能由非原来的软件开发人员承担，所以符合软件工程定义的文档资料是决定软件使用寿命的重要因素。因此，在软件开发过程中应用生存周期方法比较适合当前的现实情况。

我们在开发一些课题项目的软件时，研究生和本科学生参予较多。曾经发生过一批学生离校后，换一批新的学生接替开发工作，由于文档资料不全，无法顺利交接，只能从头开始。严重的时候一个规模不大的软件要从头开始三、四次。而在开发“863”高技术计划中的项目时，我们坚持应用软件生存周期方法，在开发人员交替时，由于有严格规范的文档，交接的过程比较平稳，没有发生不得不从头开始的情况。

因此，笔者认为，在新的软件技术环境下，以生存周期方法作为软件工程的主框架，对生存周期的一些阶段的方法和技术作一些必要的改进，是行之有效的正确思路。

1.需求分析阶段的原型方法

根据生存周期方法的阶段划分，需求分析阶段的目的是得出目标系统“必须做什么”的结论。常用的分析工具主要有层次方框图，Warnier 图和 IPO 图等。这些方法都是建立在认定用户对要开发的软件系统有完整、精确的需求的基础上，而系统分析员的工作是在于如何沟通与用户的通信交流。长期的实践提示，这个对用户的假定是有偏差的。首先，所谓用户是由行政领导、部门主管、具体操作人员等不同层次的人员构成的，这些人员对目标系统的认识往往只局限于自己的工作范围，对总体的要求并不清楚；其次，这些用户如果并没有在一个已有的计算机系统中工作和实践过，由于人类认识能力的限制，不可能预先准确无误地说出全部需求。而一些纵向课题，是依据上级部门的指令建立本单位本部门的计算机系统，用户一般地对目标系统只有模糊笼统的认识；第三，一个单位建立计算机系统后，工作结构甚至部门的划分都会有所变动，因此用户实际使用目标系统后，也常常会改变原来的某些想法，提出一些新的需求，这样的过程甚至会反复多次。

我们在“863”计划上海二纺机点的软件开发过程中，由总师组和厂方领导编制的需求规格说明书拿到该厂 CIMS 研究所讨论，发现与该所技术人员的意见大相径庭。

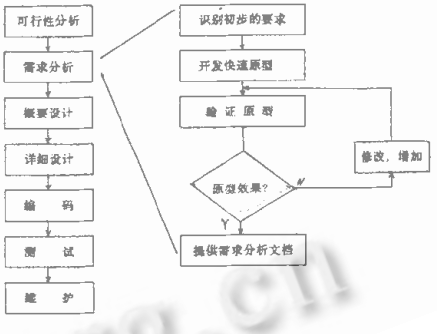

图 1

另一方面，自从六十年代后期提出“软件工程”概念以来，计算机硬件的性能经历了几“代”的增长：CPU 速度提高了 70 倍，内存容量提高了 200 倍；而事件驱动的程序设计主要是围绕输入来编写程序的，在设计的早期即能较清晰地看出输入与输出的关系。这些因素都提示了在需求分析阶段可以引入“原型化”方法，以一个快速构造出来的软件“原型”把实际的软件功能展示在用户面前，用户使用了原型系统以后，能够以相当高的准确程度指出软件系统的哪些行为特性是符合要求的，哪些是他们感到无用的，以及还需要哪些新的功能。人类的天性是对物理模型的理解，要比对静态描述和图表的理解来得容易和正确，感性认识先于理性认识，因此在需求分析阶段用原型法代替预先定义的方法，是符合唯物主义认识论的。图 1 表示了原型法在软件生存周期

方法中的位置和各个步骤，以下加以详细说明。

(1)识别初步的需求。开发人员利用可行性分析阶段获得的信息，同时通过对已有软件系统的调查和与用户的交流，迅速发现一组基本的需求。它不必是完整和精确的，但是一般应该比较最终得出的需求分析，有50%以上的准确性，以免后阶段有过量的迭代过程。

(2)开发快速原型。在识别出基本的需求以后，开发人员利用集成开发环境型的语言工具，如 Microsoft Visual C++，Boland C++，Microsoft Foxpro 等，快速地构成一个软件的原型。集成开发环境型的语言工具提供了半自动的编程功能，开发人员利用各种向导(Wizard)和类库，可以自动完成软件界面和控件的程序生成。笔者在某个软件项目的开发前期阶段，用 C 语言开发了一个交互屏幕型的原型，包括开始图标.功能菜单.各种对话框、状态条和打印机输出，用了近二个月时间，而用 Visual C++ 4.0v 开发同样的原型，只用了不到二个小时。当然前者连图形文件在内，运行文件只有300KB 的长度，而后者长达 2MB。正如前文提到过的，由于计算机硬件技术的发展，软件人员不必再为节省一点存储器容量和提高一点软件运行速度而费尽心机；由于计算机软件技术的发展，他们也不必化费大量的时间和精力去编写一些重复的，基本相同的程序功能段。正是软、硬件技术的进展，使原型技术的应用得到了长足的发展。

这个软件的原型应该包含目标软件系统的核心部分，相对目标系统有一定的深度和广度，以便开始与用户进行有意义的讨论，并从它开始迭代。用于识别需求分析的原型一般有两类，第一类称作水平原型，包括目标系统的大部分功能，只是对这些功能都进行了简化。水平原型是需求分析阶段的主导原型，它一开始就把目标系统具有的功能菜单.控件特性和输出方式都展示在用户面前，应该是原型法需求分析的良好开端。第二类称作垂直原型，作为水平原型的一个分支，只包含目标系统的一个(或几个)子功能。这是因为当目标系统的规模很大时，可以将要探讨的问题分解为多个原型。如果只有几个特殊的子系统带有不确定因素，就可以各用一个原型专门讨论这个不确定的子系统，垂直原型是水平原型的补充方法。

(3)验证原型。通过让用户实际操作原型系统，熟悉目标系统，进而评估基本需求。应该在用户和开发人员的交互中找出隐含的错误，发现不正确的行为特性，补充缺少的功能，并且改善软件系统的用户界面。特别需要指出的是，根据我们以往的经验，应该在验证原型时，提出软件测试的思路和建议。这一点将在下文中详述。

(4)原型效果。判定的结果可能有不同的转向，或者是修正原型，增强原型的功能，进行下轮迭代；或者是整理原型，提供需求分析阶段的文档。

当然发现严重的理解错误产生废品的可能也是存在的，此时应该废弃原型，重新识别一组基本的需求，构筑新的原型。

(5)修改、增加。用户和开发人员对目标系统的理解加深，对原型的功能和行为特性进行修改和扩充，以正常的迭代直到用户感到满意为止。

(6)提供需求分析文档。原型方法作为需求分析阶段的一种方法，像其他软件生存周期方法的阶段一样，必须提供严格、正确、详尽的文档，为下阶段的开发服务。原型方法通过动态的演示，以用户为中心获得和识别需求，原型即是一个自动的文档。定义系统成份和它们之间所有的关系，即形成了数据字典；分析数据的结构和变换即形成了数据流图；评估和认可原型的功能和行为特性，即形成了需求规格说明书。提供了这些文档，我们认为运用原型技术的需求分析阶段已经圆满结束。固然一个“好”的原型可以继续演进为目标软件系统，但是九十年代对于“好”的软件的主要着眼点是可读性和可维护性，信息隐蔽和局部化成为软件开发的重要概念，因此，本文建议废弃用于需求分析的原型，转而根据生存周期方法的阶段划分，进入概要设计阶段。

2.概要设计阶段的改进

概要设计阶段的任务是设计软件的结构，也就是确定软件可以划分成哪些程序模块，以及这些模块之间的相互关系。

事件驱动的程序设计使得开发人员围绕输入和消息编程，面向对象技术在数据和处理操作之间建立联系，两者都使信息和处理模块化。在概要设计阶段，根据数据字典定义数据结构就产生了数据抽象，即对象的属性；根据数据流图定义对数据的操作，就描述了对象，软件的结构也就建立起来了。借助使用消息机制，就得到了接口的定义和描述。所以面向对象的概要设计最好是从中间开始设计：选择一些明显应该是对象的成份，设计好之后，另外一些对象便显现出来，逐步地就分析出一个对象集合，这个对象集合体现了软件功能的大部分要求。这时，再重新比较已有的对象和数据流图，找出“漏洞”，添加新的对象，直至完全体现目标软件系统的要求功能。如图 2 所示。

类和对象内包含的信息(数据和过程)对于不需要

这些信息的其他程序段来说，是不能访问的，因此极好地体现了信息隐蔽的概念。而局部化的概念是与信息隐蔽概念密切相关的。所谓局部化是指把一些相关的成份物理地放得彼此靠近，主要也是为了有助于实现信息隐蔽。我们通过上述的方法得到的类的集合，天然具有公共环境耦合，概要设计人员应该尽可能地使存在耦合的两个对象，一个只往公共环境送数据，另一个只从公共环境取数据，形成比较松散的公共环境耦合。

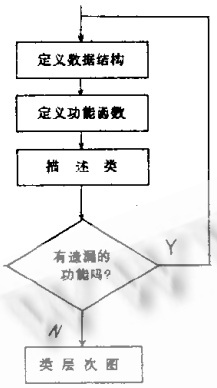

图 2

对象中的成员函数由于是根据数据流图来识别和划分进类的，因此通常是通信内聚和顺序内聚的，本来就是独立性较好的内聚程度。当然可以通过修改设计力争做到功能性内聚，但是会引起接口复杂。因此，设计类的内聚程度，可以参考软件工程的启发式规则由开发人员折衷权衡。

综上所述，面向对象技术使目标软件自然具有信息隐蔽和局部化性质，因此一些传统的概要设计工具如 HIPO 图等，不再成为必要的设计手段，本文提出的简化方法加上开发人员的手工修正，修改后的概要设计一般都能获得较高的模块独立性。

3. 测试阶段的变化

原型技术应用于软件生存周期的需求分析阶段后，用户参加软件系统开发的主动程度有了很大的提高。实际上，在原型演示和与用户交互的时候，用户和开发人员必然会把软件测试提到议程上来。因此特别是单元测试计划很可能提前到需求分析阶段，在迭代原型时就已经开始了。经验表明，这一阶段提出的大部分是黑盒测试，充分利用用户的技术背景知识，划分出测试的等价类和边界值，进而制定单元测试和总体测试的黑盒部份的计划，是高效率的表现。而白盒测试计划，则可能要等到测试阶段正式开始时才能制定。测试计划可能会变成在两个不同的生存周期阶段制定和进行。

重要的变化是用户加入测试工作的时间提前了，加入的程度加深了。因此对非软件专业的用户必须作心理上的准备：即，需求分析阶段用户的加入是“建设性”的，是力图勾勒出一个目标软件的轮廓；而测试阶段用户的加入是“破坏性”的，是竭力证明目标软件不能按照需求正确地工作。

同时还要向用户证明，穷尽测试黑盒法是不可能的。我们往往用这样一个例子来说明：假设驱动某个对象的“消息”是一组 5 个整数，计算机字长 32 位，5 个整数的各种可能值的组合共有 $2^{32}\times 2^{32}\times 2^{32}\times 2^{32}\times 2^{32}=2^{160}$ 种，假如每执行一次程序需要 1 毫秒，执行这么多次大约需要一百万年！

一个具有充分技术背景知识又充分理解测试条件受限的用户，在制定测试计划时帮助划分等价类和确定边界条件是十分有用的。

当然也要证明白盒法的穷尽测试同样是不可能的，有这样一个例子：一段很小的程序，对嵌套有二分支 IF 语句的循环执行 10 次，则共有 $2^{10}=1024$ 条路径。请见图 3。可以推知对目标软件的每条路径都执行一次也是不可能的。

图 3

由于在需求分析阶段就鼓励用户积极加入制定测试计划，让用户理解软件工程的重要断言：“测试只能查找出程序中的错误，不能证明程序中没有错误”，是用户加入“好”的重要基础。

（来稿时间：1998 年 11 月）