计算机科学2001Vol.28№2.10

基于UML的需求分析模型度量

The Metrics of the Requirement Analysis Model Based on UML

贾 亮 潘秋菱 刘宗田

(合肥工业大学微机所 合肥230009) (上海大学计算机学院 上海200072)

Abstract For a long time, the requirement model is difficult to measure. This paper introduces the developing status of requirement analysis modeling and metrics, brings forward the point of turning the objective of metric to UML, after presenting the requirement modeling methods of using UML, puts forward the metrics of ours.

Keywords UML, Requirement analysis model, Metrics, Scale, Complexity, Depending degree, Depended degree

一、引言

从最初软件度量概念的提出到现在,软件度量学已经历了四十多年的发展,它作为软件工程的一个研究方向,其根本目的就是要用软件度量学的方法来科学地评价软件质量,更有力地对软件开发过程进行控制和管理,合理地组织和分配资源,制定切实可行的软件开发计划,以低成本获得高质量软件[1]。然而事实上软件度量学并没有充分发挥其应有的作用,目前的各种软件度量方案大多还具有事后性,不能在软件生命周期的早期进行有效的度量,从而在早期不能对项目进行有效的评价和估计。因此,针对需求阶段的模型进行度量一直是软件度量学关注的一个焦点。

在目前,软件界有两种占主导地位的需求分析建模方案:结构化建模和面向对象建模。针对二者,人们提出了一些较好的度量方案。

1.1 结构化建模的度量方案

结构化建模方案使用的是实体-关系图、数据流图和控制流图,针对它,比较具有代表性的度量方案是功能点度量[3]和Tom Demarco的bang[2]度量。

基于功能点的度量是针对数据流图提出的一种度量方案,它通过系统中的用户的输入输出个数、用户对系统的询问个数、所用文件个数以及外部接口个数来估计未来系统的大小。这种度量方案强调对数据的度量,而没有考虑系统的行为和功能,于是后来人们对其进行了扩展,其中比较典型的是特性点[4]和三维功能点[5]度量。特性点除了考虑一般功能点的度量外,还考虑有关算法复杂度的度量,因此它比较适合于实时系统、控制处理系统以及嵌入式软件系统这些具有高算法复杂度的软件。三维功能点主要从三个方面对需求分析模型进行度量:数据维、功能维、控制维。数据维的度量和一般的功能点度量内容很相似。功能维的度量主要是考虑数据从输入到输出的转换所需操作的复杂性,控制维则主要是考虑状态转变的个数等信息。

Tom Demsrco的bang度量方案以上述三种建模图为度量对象,首先抽取需求分析模型中所有不能再细化的元,然后对它们进行度量,再将这些元加权进行累加。此外它还对计算方法进行了分类,根据是功能型系统还是数据型系统,而分别采用不同的度量算法。

1.2 面向对象建模的度量方案

面向对象建模是一个兴起不久的建模方案,对于它,人们更多的是提出一些对设计阶段模型的度量,而对于需求分析模型,还没有系统的度量方案。

1.3 基于UML软件度量

针对面向对象的需求分析模型没有提出较系统的度量方案,因为以前面向对象建模语言和方法较为混乱。但目前各种建模语言和建模方法已逐渐统一,形成了统一建模语言UML,这种语言集众家所长,占有了绝大部分面向对象建模市场,而且正在占领结构化建模的领域。在这种情况下,针对UML开展度量研究的必要性也就日益突出,而且由于UML本身在表示内容上的丰富性、规范性等优点,使得针对UML表示的模型进行度量可以提取较多、较完善的有用信息。

UML是一种使用图形对系统进行描述的语言,

贾 亮 硕士生,研究方向:软件工程;潘秋菱 博士生,研究方向:软件工程,Petri Net;刘宗田 博士生导师,研究员,研究方向:软件工程与人工智能。

它的九种图可以表示从需求分析到方案设计的绝大部分中间产品，在它的基础上可以开展需求度量、设计度量等度量研究工作。我们实验室长期致力于软件度量的研究，开发的C++源码度量工具运行良好[6]，目前我们又将度量研究的焦点转向UML，并开发出了一个基于UML的度量工具原型。下面将在介绍UML需求建模基本方法的基础上，阐述我们的度量方案。

二、基于UML的需求建模方法

UML中需求分析模型一般可以用用例图、活动图、类（概念类）图、对象（实际对象）图、状态图及交互图（包括合作图和顺序图）等图来表示。其中用例图包括执行者、用例两个要素，执行者是与系统进行交互的人或外界系统，用例指的是用户与系统的一次典型交互，根据系统的不同，以及分析人员的不同可以得到不同的用例划分。对于每个用例又可以使用活动图来进行描述，活动图包括：活动结点、状态结点、转移、事件、同步结构、判断结点等要素，其中活动结点和状态结点可以包含子结点或者用子活动图来表示。从活动图中可以分析抽取实际对象之间的关系、概念类之间的关系从而形成类图和对象图。对于每个概念类或实际对象，如果其状态变化较复杂，还可以画出其状态图。另外，根据现实世界中对象（类）之间的消息发送响应等，可以画出实际对象之间的交互图。这些图并不是一定都要一一画出，系统分析员可以根据自己的需要有选择地绘制。

以一个销售系统为例，其需求分析及结果如下：首先进行用例的获取，得到用例图（如图1所示）；其中每个用例可以用活动图表示，以输入新定单为例，得到其活动图表示（如图2所示）；对于现实世界中所存在的对象（类）及其关系又可以得到对象（类）图（如图3所示）。

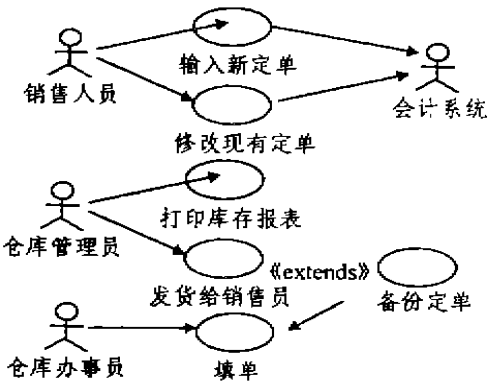

图1　用例图

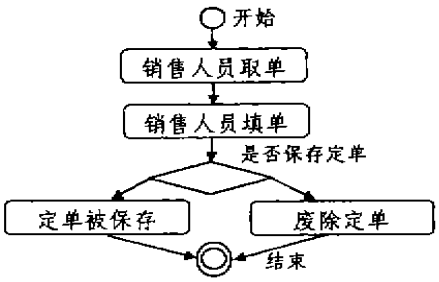

图2　活动图

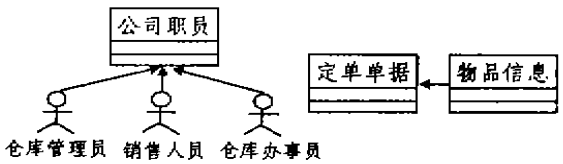

图3　类图

三、UML需求分析模型度量

3.1　用例图度量

用例图是需求分析的一个主要表示图，它给出了一个系统功能需求的总体概览，通过用例图我们可以度量以下指标：

·用例总数和虚拟用例总数：对于用例，不同的人会有不同的划分方式，如何划分用例一直是一个难点，通过用例的总数及虚拟用例总数的计算，可以了解用例划分情况，用例划分得过多或过少都不合理，对于不同大小的项目用例划分情况也不一样。一般来讲系统的大小和用例的个数呈正线性关系。另外，虚拟用例的个数也是一个有价值的数据，这个数据过大，会使系统的关系复杂，而太小又不利于刻画系统中的功能结构。

·执行者总数　执行者是与系统交互的外界系统或人，它也可以从侧面反映系统的规模。一般说来执行者数目较多的系统其规模也较大。

·与某个执行者相关的用例个数，与某个用例相关的执行者个数　与某个执行者相关的用例个数反映的是某个执行者的重要程度，如果这个数字很大，就要考虑执行者的获取是否正确，是否有必要将该执行者再划分为几个执行者。而与某个用例相关的执行者个数，则从侧面反映了该用例的重要程度，如果这个数字很大则该用例一般较复杂，有必要重新评估用例的划分是否正确。

以上信息都是系统需求分析模型的一些宏观信息，通过这些信息，用户可以对整个系统需求有一个大概的了解。

3.2　活动图度量

活动图中包含有不同种类的元素，而状态结点和活动结点这样的元素还可以出现嵌套结构，这样就可以使用针对一般拓扑图的度量方法来评估活动图。于是提出如下活动图度量指标：

·规模度　活动图中包含有活动结点等不同的图元，每种图元对整个活动图的规模度的权值应该是不同的，而且即使是同一种图元，由于其内部操作的烦琐程度不同，对活动图的权值贡献一般也不一样，除了考虑权值外，还有其他因素必须考虑，象判断结点的分叉数、循环的个数。于是得到活动图规模度的计算方法如下：

令活动图的规模度为 Scale，循环个数为 k，第 i 个循环结构的权值为 C_i；第一层普通结点（活动结点或状态结点）的个数为 n，其中第 i 个普通结点的权值为 W_i；第一层判断结点的个数为 m，其中第 i 个判断结点的分叉数为 T_i，权值为 D_i；第一层同步结构的个数为 L，第 i 个同步结构的子结点数为 Z_i，其中第 j 个子结点的权值为 S_i 则有：

1. 规模度 $=\sum_{i=1}^{k} C_i+\sum_{i=1}^{n} W_i+\sum_{i=1}^{m} T_i\times D_i+\sum_{i=1}^{L}\sum_{j=1}^{Z_i}(1+S_j)$

2. 对于第一层中一般结点以及同步结构中结点的权值计算则有两种情况：

2.1　如果存在嵌套子图，则以这个子图为度量对象，转1；

2.2　如果不存在嵌套子图，则其内部 Action 可分为四种，on entry，on exit，do，on event，前三种 Action 个数总共为 r，On event 的事件发生概率为 P_i，其 Action 个数为 s 则有：

$$W_i(\text{或 } S_j)=r\times W_i+\sum_{i=1}^{s} P_i\times s\times W_e$$

由此可以得到其规模树型分布图（如图4）。规模度考虑的是系统可能具有的工作量，所有类型结点的存在都会对系统的工作量有影响，这个度量指标可以使系统分析员对将来开发的系统的大小进行估计。

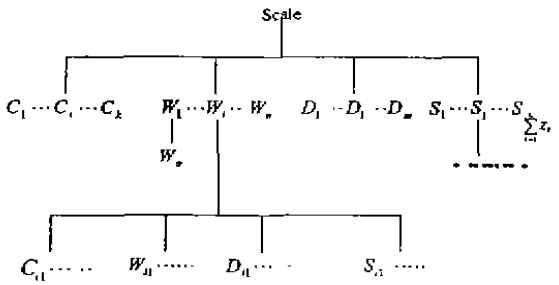

图4　规模度树型图

·复杂度　复杂度是由活动图中所使用的判断结点，同步结构、循环结构等特殊结构影响的。其计算方法如下：

令第一层结点中，循环结构的个数为 k，第 i 个循环结构的权值为 C_i；普通结点的个数为 n，第 i 个普通结点权值为 W_i；判断结点的个数为 m，第 i 个判断结点的分叉数为 T_i，权值为 D_i；同步结构的个数为 L，第 i 同步结构中子结点的个数为 Z_i；其中第 j 个子结点的权值为 S_j。则有：

1. 复杂度 $=\sum_{i=1}^{k} C_i+\sum_{i=1}^{n} W_i+\sum_{i=1}^{m} D_i\times T_i+\sum_{i=1}^{L}\sum_{j=1}^{Z_i}(1+S_j)$

2. 对于第一层中一般结点以及同步结构中结点的权值计算则有两种情况：

2.1　如果存在嵌套子图，则以这个子图为度量对象，转1；

2.2　如果不存在嵌套子图，则 $W_i=0$ 或 $S_j=0$。

这样就可以得到复杂度树型图，它和规模度树型图很相似，只是不考虑一般结点的贡献，而且对各种结点使用的权值大小也不一样。

复杂度反映的是活动图中各种特殊结构对复杂度的影响。普通结点对整个活动图复杂度来讲，只是增加了活动图的规模，因此只是增加系统开发的强度，对开发的难度并不会有很大影响。而特殊结点的存在不仅会增加活动图的规模，而且会增加活动图逻辑上的复杂性，这势必是影响开发难度的一个重要因素，

规模度和复杂度，前者强调整体的规模大小，反映的是系统可能的工作量，后者强调结构的复杂性，度量时只考虑特殊结构的存在，反映的是系统开发的可能工作难度。二者从不同方面对系统进行了描述。二者都使用了树型结构，这种结构可以对系统的工作强度和难度的分布有一个较全面的了解，系统分析员可以根据这些分布图了解哪个结点的工作量较大，哪个结点的工作难度较大，而这个结点的工作量较大或工作难度较大又主要是由它的哪些子结点引起的，这就为以后的设计提供了参考，设计人员就可以对以后系统设计的大小和难度分布事先有一个较好的了解。

3.3　对象（类）图及交互图度量

在需求分析阶段的类还是概念类，并没有映射为设计中的类，但它们的继承关系以及其他消息发送等关系仍可用简单的对象（类）图和交互图来表示。这时 c&k 的度量体系[7]还不能完全映射到需求分析的度量，此时仅可以较有效地度量以下指标：

概念类的个数　在这个阶段类的个数可以反映现实系统从概念类意义上的大小，一般来讲，概念类越多的系统越复杂。

概念类的依赖度关系　分为两个方面，一个是依赖度，另一个是被依赖度。其中概念类之间的依赖度指的是某个类所依赖的类的数目与类的总数的比值，其计算公式如下，令类的总数为 C，类 i 所依赖类的数目为 D，依赖度为：

依赖度＝D/C

而被依赖度指的是依赖于某个类的类的个数与类的总数的比值。令类的总数为 C，依赖于类 i 的类的个数为 D'，被依赖度为：

被依赖度＝D'/C

依赖度反映的是该类的稳定性，一个类对其它的类的依赖程度越大，这个类越不稳定，这个类就需要更认真的规划设计；而被依赖度反映的是一个类的重要性，一个类越是被依赖的程度越深，则这个类设计的优劣对系统的影响越大。依赖度和被依赖度可以为设计阶段类的设计提供参考依据。

3.4　状态图度量

状态图是对状态变化较复杂的类或对象进行补充说明的一种图，对于它主要是度量出该类的复杂性，以使开发人员对这个类事先有充分的估计。状态图中主要是状态和转移两种结点，故对于状态图的度量，只需简单地计算状态的个数。

结束语　一个好的度量方案不仅要能度量出一些关键的指标，而且要尽可能再现一个系统所有可能有用的特征，这好比我们认识事物，只有把事物的尽可能多的特性了解清楚，才能更真实地把握事物的运动规律，才能以不变应万变。针对需求分析模型的度量，是软件度量的一个重要部分，它可以使得在项目还没有设计之前对项目情况有一个较好的了解。由于目前 UML 在市场上的主导地位以及其本身信息的完善性、规范性等优点，使得基于 UML 的软件度量研究成为当务之急，本文正是从这点出发，提出了自己的度量方案。

参 考 文 献

1　潘飚. C++语言面向对象的软件度量学研究.[硕士学位论文] 合肥工业大学，1999.5

2　Albrecht A J. Measuring Application Development Productivity In. Proc. IBM Application Development Symposium，Monterey，CA，Oct 1979. 83～92

3　DeMarco T. Controlling Software Projects Yourdon Press，1982

4　Jones C Applied Software Measurement. McGraw-Hill，1986

5　Whitmre S A. An Introduction to 3D Function Points Software Development，April 1995. 43～53

6　李心科，刘宗田，等. 一个面向对象软件度量工具的实现和度量实验研究. 计算机学报，2000(11)

7　Chidamber R，Kemerer F. A Metrics Suite for Object-Oriented Design. IEEE Trans，Softw.，Eng ，1994，20(6)

(上接第127页)

```
else
 for (i=1;i<=n;i++)
   {s=a[i]-(xorall() ^ a[i]);
    if (s>0)
      {printf("computer take %u from No. %d \n",s,i);
      a[i] ^ =xorall();
      break;
      }
 }
  if(sum()==0)
     {printf("computer win!");break;}
  status();
  while(1)
     {printf("Input your selection\n(exp 1 2 means take 2
        from No. 1);\n");
      scanf("%d %u",&e,&t);
      if ((e>=1)&&(e<=n)&&(a[e]>=t))
        a[e]-=t;break;}
      else
        printf("data error! re-input…\n");
      }
if(sum()==0)
{printf("you win!");break;}
}
}
```

2.2　取石子游戏2

任给 N 堆石子，堆数 N 及每堆石子数由游戏者给出，两人(游戏者与计算机)轮流从任一堆中任取(每次只能取自一堆)，规定每方每次最多取 K 颗，计算机先取，取最后一颗石子的一方获胜。试设计一种方法，使计算机有较多的获胜的机会。

分析：记第 i 堆石子数为 x_i，令 $y_i=x_i \bmod (K+1)$，$i=1,\cdots,n$，定义 $T=y_1 \text{ xor } y_2 \cdots \text{xor } y_i \text{ xor } \cdots \text{ xor } y_n$。如果初始状态时 T＝0，则为平衡态，先走的计算机一方没有获胜策略，如果初始状态时 T≠0，由定理1，必存在某个 $y_i>T_i$，可在第 i 堆中取 y_i-T_i 颗，使对方处于 T＝0的平衡状态，在以后的各轮中，对方在第 i 堆中取 r 颗，若 $x_i>k$，计算机就在同一堆中取 k＋1－r 颗，这时 y_i 不变。若 $x_i \leqslant k$，则重新计算 y_i 及 T，按问题2.1的方法选取，这时，计算机有必胜策略。具体实现细节不再赘述。

对上述算法略做修改，便可解决问题2.1、2.2的对偶问题：取最后一颗石子的一方为失败。

参 考 文 献

1　Kernighan B W，Ritchie D M. The C Programming Language(2nd). Prentice Hall，1988 48，207