计算机科学2000Vol.27№.3

动态开放设计

Dynamic Open Design

张建高[1] 王蔚韬[2] 马矿生[2] 朱晓红[2]

(重庆建筑大学管理学院[1] 计算机科学系[2] 重庆400045)

Abstract *In this paper, we discuss the present software design ideal and method from the angle of general system theory and system engineering, and expound their advantages and limitations simply. Then we introduce the concept of entropy into the system of computer science, suggesting dynamic open design ideal. And we clarify the principle of dynamic open design and the technology of open property design. Finally, we brief our works in this respect.*

Keywrods Entropy, Open properties, Information exchange, Software's system, Dynamic open design

1 对现有设计思想的思考

结构化设计和封装与组合等设计思想以及面向对象技术的出现,可以说是计算机科学,特别是软件科学的一次革命和里程碑,使得许多大型软件工程得以完成。现在的软件设计可以说和十九世纪的自然科学十分相似,即将知识和物质不断细分,然后再将其重组,以便获得整体的认识。目前的软件设计也正是这样,把一个大型软件工程划分为若干个子部分,再把这些子部分进一步细分,直到归结为具有某项或某些独立功能的模块或函数,然后再将它们封装或组装起来。这种设计方式可以动员成千上万的程序设计员来共同完成一个大型软件,而不要求程序设计员对整个软件的设计思想和运作方式等有深刻的理解和认识。

当前,大多数的计算机软件设计都是沿用八十年代或者更早的设计思想,即上述结构化设计思想和模块封装与组合的设计思想以及面向对象的技术等等。在此思想的指导下,必然导致计算机的各种操作系统和各类应用软件越来越庞大,内部的有机联系越来越薄弱,总体协调性和可靠性越来越差,许多子部分、模块和整个系统缺乏非动态开放性。同时,对计算机的处理器速度要求越来越高,存储空间要求越来越大,而当今计算机科学的发展已明显地显示出硬件的更新换代已经不能满足软件的发展需求。如果我们把整个计算机科学看成一个有机的系统,这个系统的熵似乎一直在增加,而整个系统却不知道从哪里去汲取负熵,因此,也许有一天整个系统会由于熵太大而达到崩溃的混沌状态。

在这里,我们在对现有设计思想的思考中提到了熵和系统的动态开放性,因此,有必要对熵和一般系统论中的某些概念作一些简单的介绍。

2 熵的概念和一般系统论简介

熵是热力学中的概念,由德国物理学家克劳修斯(Clausius,1822—1888)首先提出。熵是描述热力学系统状态的一个物理量,是个态函数,以符号 s 表示,一个系统熵的变化 ds 定义为该系统吸收的热量 ∂Q 与绝对温度 T 的商,即 $ds \geqslant \partial Q/T$,式中等号对应于可逆过程,不等号对应于不可逆过程,任何自发过程都由非平衡态趋向平衡态,而熵也在不断增加,到了平衡态就不再变化,所以系统的熵在平衡态时达到最大值。因此,熵的最大值又可以作为自发不可逆过程限度的准则。随着近代科学的发展,特别是交叉科学的发展,熵的概念由物理学走进了各个学科,已被推广应用到各个领域中。熵作为一个科学概念,用来表征事物(系统)不确定、无规则、无组织、混乱或无序的性质;作为一个量,用来度量事物(系统)不确定性的大小、无规则(无组织、混乱或无序)的程度。美国学者杰里米·里夫、特德·霍华德在他们的著作[7]中宣称统治人类各方面的将是熵的定律,即热力学第二定律,并赋予熵这个物理学的概念以哲学的意义,广泛应用于哲学、心理学、经济学、政治学、社会学等各个学科,认为人们只有以熵定律作为新的世界观,来考察社会生活的各个方面,才能减缓熵的增长和延缓走向"热寂"的速度。

一般系统论由美籍奥地利理论生物学家 L. V. 贝塔朗菲创造。一般系统论指出不论系统的种类、组成部

分的性质和它们之间的关系如何，存在着适用于综合系统或子系统的一般模式、原则和规律。贝塔朗菲创立的一般系统论不仅提出了一些有价值的思想和原理，而且已被广泛应用于许多领域，从物理学、化学、生物学、心理学到社会经济学等各方面，都取得了显著效果。一些科学家在各自的领域中沿着贝塔朗菲开创的开放系统理论深入研究一般系统论，并获得了关于系统的一系列重要规律。其中最著名的有I. 普利戈金的耗散结构理论，M. 艾根的超循环理论和H. 哈肯的协同学。一般系统论原理指出，开放系统的运动在一定条件下可以是一个减熵的过程，即系统不断与环境进行物质、能量和信息交换以获得负熵，从而延缓系统走向消亡的过程。在本文中，我们将把一般系统论原理和开放系统理论引入计算机软件设计中，提出动态开放设计思想和原理。

3 计算机软件设计中的熵

在计算机软件设计中，面向对象技术的提出，可以说在某方面为计算机科学的发展提供了一个汲取负熵的方法。把实际问题中要处理的对象抽象出来，使之成为具有一定普遍意义的抽象对象，从而将研究混沌无序实际问题转化为研究较为有序的抽象问题，是一个使计算机软件设计从无序走向有序的减熵过程。软件设计所针对的是处理抽象了的对象，使得所设计软件不但可以处理实际问题中所提出的具体对象，也可以处理与该实际对象具有相同或相似属性和品质的其它对象，或者说可以处理某些较一般的对象，也使得计算机科学系统中的熵在减少。因此，面向对象的技术得以发展和普及并受到计算机科学界的广泛重视，正是其具有使计算机科学这个有机系统获得负熵，使整个系统得以减熵的性质。

但是，却没有多少人看到面向对象技术同时也是一个增熵过程，并且随着计算机科学的不断发展，其增熵的速度已越来越大于减熵的速度。从系统科学的角度看，动态开放性是系统的生命力所在，而面向对象技术从其本质上来说是一种非动态开放性技术，独立的封装和机械式的组装，使得整个系统的各层次之间和各子系统之间的有机联系变得越来越薄弱。当今的许多软件就象工业革命初期的机器一样显得十分笨重和庞大，同时效率低下。一个软件就象一大堆积木，软件的总体设计师是搭积木的人，一般的程序设计员是做积木块的人。但是，当积木搭得足够大时，就会缺乏稳定性。因此，现在的大型操作系统和大型应用软件常出现异常情况和死机是不可避免的。同时，随着计算机应用领域的不断扩大，被抽象的对象也越来越多，而这些抽象对象已逐渐显示出混乱或无序的状态。面向对象的技术在设计思想上是针对抽象对象的，必将随着抽象对象的增多和无序化而出现组合爆炸。因此，面向对象的技术同时还是一个快速的增熵过程。

一般系统论和耗散结构理论指出，任何生命系统都是远离平衡态的一种动态开放系统，能够汲取负熵使无序状态转变为一种在时间、空间或功能上的有序结构。为了使计算机科学作为一个有机系统能够更好地汲取负熵，避免出现组合爆炸，追求软件设计、编码等的高效率，使软件不再靠简单的堆砌组装（包括派生等）来提高性能和扩展功能，同时减少大型软件的代码容量，我们将在下面阐述软件的动态开放设计思想和原理，并介绍我们在这方面所作的初步尝试和探讨。

4 动态开放设计思想

我们提出的动态开放设计思想包括三个层次上的动态开放性。

（1）总体设计上的动态性和开放性。是指在总体设计上，各部分的有机链接以及软件的基础核心、具体对象和用户界面等方面都具有动态性和开放性。这种动态性不是狭义的软件设计中的动态连接和动态连接库，而是指设计上、构思上、整体框架上的动态性。这里所说的开放性，也不是一般软件设计人员所理解的开放性，而是指整体软件系统的开放性，软件功能范畴和应用领域的开放性。如软件基础核心对应用领域的开放性，它要求基础核心既有抽象性和一般性，又有其独有的具体性和特殊性，等等。

（2）程序设计和模块设计的动态性与开放性。程序和模块设计的动态性，并非是简单的动态链接，而是要求其内部结构具有动态性，这种动态性体现在编译、运行和性能的扩充上等等。因此，这种动态性是动态中有静态，静态中有动态，动态与静态是有机地相结合在一起的。其开放性要求程序和模块既有抽象的特征，又有具体的内容，既有一般性构架又有特殊的应用范畴，动态开放的程序和模块的功能和对象的扩充或修改，可以采用平行的嵌入或简单的替换，不象目前的面向对象技术那样，只有靠类的派生或继承来完成。因此，我们的动态性和开放性设计技术是面向对象的技术和结构化设计等的延伸和发展，而不是其否定。

（3）面向用户的动态性和开放性。这里提到的面向用户的动态性和开放性不是象现在风靡一时的Linux等那样，依靠完全开放源代码来达到对用户的开放性和动态性。因为用开放源代码的方式使用户获得所期望的某些功能和修正，要求用户必须具备相当的程序设计能力和编程技术，这对于大多数用户来说是永远不可能的，我们不可能要求每一个人都象掌握加减法那样掌握不断发展、不断进步的程序设计技术和编程

技术。因此，我们所说的开放性是指软件在形式上的开放性和内容上的开放性以及自组织、自管理模块的开放性，对于完全不懂程序设计技术和编程技术的用户而言，仍可以通过自组织、自管理模块的开放性对软件的若干形式和内容进行修改，以获得希望的效果。对于懂得一些基本的程序设计、编程技术或者只懂一些基本数据库设计的用户，则可以通过形式上的开放性和内容上的开放性对软件进行形式内容替换，将其改造成自己所想要的软件，也通过设计上和程序模块的开放性将自己设计的特定功能程序模块装载到一般性构架中以获得所要求的特殊功能。

因此，动态开放设计思想所包含三个层次上的动态开放性是密不可分，有机地结合在一起的。要达到完全的面向用户的动态开放性，必须要求总体设计上的动态开放和程序模块设计的动态开放，要在软件设计上做到这一点，就必须用开放系统的观点来看待软件，把软件看成一个具有生命力的开放的系统。因为在我们生活的世界上每一个有生命力的系统都是一个动态开放系统，所以这样的系统能够和外界进行能量及信息的交换，能够从外部世界汲取负熵，并以此维持其活力。

5 动态开放设计原理

当把软件看成一个开放的系统时，我们的设计就应该使软件具有一般开放系统的特征，并使设计的软件具有一般开放系统的若干属性，才能使软件真正具有我们提到的三个层次上的动态开放性；使软件具有尽可能少的代码和尽可能强大的功能，才能使代码重用达到更高的效率。因此，按照前面提出的动态开放设计思想和一般系统论理论，我们认为动态开放软件系统设计应该遵循如下的与软件设计特性相结合的一般系统原理。

(1)整体性原理。由各部分所构成的软件系统整体，必然有新的某些功能出现和一些旧的功能消失，整体的功能并非简单地等于各部分功能之总和。整体性原理的基本内容有：a)整体的有机性。这是指软件系统整体与其组成部分(模块)、模块与模块、系统整体与环境都是相互联系、相互作用、相互制约的，从而形成了一个统一的有机的整体，整体的性质与功能是这种相互联系、相互作用的综合结果。b)整体的质不同于组成部分的质。由于整体结构的作用，整体将会出现组成部分所不具有的新性质和新功能。c)整体中的组成部分的性质和功能不同于它们在独立时的性质和功能。d)整体与组成部分存在某些共同的性质和功能。e)整体与组成部分之间量的关系。由组成部分构成整体时，软件系统整体对某些属性和功能的数量，既可以放大，也可以缩小，或者保持不变。同时，软件系统整体代码量，也可以大于、等于或小于各组成部分独立封装后再组装起来的代码量之总和。

(2)开放性原理。从系统科学的观点看，封闭的系统由于内部的熵增，必然使系统从有序走向无序，而开放系统的运动在一定条件下可以是一个减熵的过程，能使系统从无序走向有序或从低级有序走向高级有序。因此，把软件作为一个系统时，可以靠开放设计、开放程序模块、开放人机交互，或象 Linux 那样完全开放源代码等来使系统获得生命力与活力，以达到系统与环境不断地进行物质、能量和信息的交换。开放系统的稳态是一种动态平衡，具有一定的自动调节能力。因此，软件系统的开放性要求该系统必须要有自组织和自管理的机制，而实现自组织和自管理机制的模块或部分对环境(包括用户)也必须是开放的。开放系统还应具备系统的等终极性或系统发展的多途径性，即从不同的初始条件出发和通过不同的途径可以达到相同的最终状态。

(3)等级结构原理。任何一个具有生命力的系统都是按严格的等级组织起来的，具有等级结构或等级层次。不同层次上的系统运动有其特殊性。层次之间是相互联系、相互影响的。因此，在研究和设计软件系统时，要考虑层次之间的制约关系。a)结构是指系统的各要素相对稳定的相互联系、相互作用的方式，即系统内部的组织形式、结合方式和秩序。b)系统结构的基本形式表现为：i)空间结构；ii)时间结构；iii)时空结构。对软件系统来说，空间结构是指系统的各部分(要素)按照一定的逻辑关系排列组合形成的稳定软件结构框架。空间结构标志着系统的广延性，它表现为各要素之间在功能上、性质上的相互协调与适应。软件系统中的时间结构是指它在时序上的兼容性、可扩展性和对所涉及领域的面向发展的特性。时空结构是时间结构和空间结构的统一。c)系统结构的特性包含结构的有序性，结构的整体性，结构的稳定性和结构的层次性。层次具有客观普遍性，它们的基本结合方式是分层次形式，即由要素(基本模块和程序)先组合成子系统，再组合成系统，然后再组成更高一级的系统。层次结构是一种"系统套"结构，也是系统演化的时序结构。层次质变律是指不同层次的系统，具有不同的结构和不同的功能，不同层次之间相互联系、相互影响。高层次与低层次之间的相互作用形成一种纵向联系，同一层次之间的相互作用形成复杂的横向联系。纵横交错的联系，就构成了一个多层次、多因素、多变量的网络结构系统。因此，动态开放软件系统的设计流程图必然是一个多层次的空间网络结构图。d)结构与功能是相互依存的。结构是功能的基础，功能是结构的表现。结构决定功能，功能又

具有相对的独立性。

(4)动态系统原理。任何系统都有其发生、发展和逐渐消亡的过程，表现出随时间和空间而变化的动态特性。要延长软件系统的生命，就必需使其具有足够的动态性。如总体框架与内容之间、界面与基础内核之间、软件系统与用户环境之间的动态性，自组织、自管理机制的动态性等。即整个系统具有绝对的动态性，同时又具有相对的静态或相对稳定性。

(5)目的性原理。在一般系统论中，目的性原理可描述为：一个系统似乎在"瞄准"一个将来才能达到的平衡态，或者说，这个事件可能被表示为由将来的最终状态所决定的。软件系统除了要尽可能满足当前用户的需求外，还应以追求有序稳定的可扩展结构为目标、它的重要内容是要追求未来用户的需求，追求所涉及领域的发展的需求。软件系统的这种目的性不是什么超人或天才的神秘设计，而是系统自组织、自管理的结果。

(6)同构性原理。同构性或同型性，是指各种在同一层次上的不同软件系统或模块在结构上的一致性或相似性，以及各种在不同层次上的软件系统或模块在结构上的自相似性或一致性。这就可以用同一模式、原则、规律等来描述完全不同的系统或模块。同构性原理为在一个领域中建立的软件框架合理、正确地引入到其它的领域提供了依据。同构性不是指各种系统或模块的表面类似性或外在类似性，而是指内在规律的形式一致性或相似性。正因为如此，不同系统或模块的元素、属性、功能、工作方式等之间才可以建立一一对应关系，使得我们可以把为某个领域开发的应用软件，在不修改任何源代码的情况下，通过用户层次上的简单维护而成为用户所期望的领域中的应用软件。

按照上面的动态开放设计原理，就可以设计出适应现代科学技术发展的动态开放型软件。这种软件不但能满足当前用户的基本需求或满足特定领域中的需求，而且在很大程度上还能满足未来发展的基本需求或其它领域中的需求。同时软件的总代码量也不会象现在的许多软件那样无休止地膨胀，可靠性和效率也将提高。

6 动态开放设计初步尝试

我们在软件"管理之窗"中对动态开放设计作了初步的尝试和探讨，目前，这个软件还不是全动态开放，其自组织、自管理机制也不是完美的，因为按照我们的动态开放设计思想和原理，要设计一个全动态开放的软件系统，需要进行大量的严密细致的逻辑推理、分析和严格精确的计算。虽然"管理之窗"目前是一个面向行政管理的数据库软件，但是只要稍加修改源代码和在用户层次上作一些维护修改，便可以成为另一领域中的管理软件，特别是那些与行政管理在抽象性质和抽象机制上相似的领域，甚至可以在不改动任何源代码和数据库表单及其字段的条件下，也可以成为其相应领域中的管理软件。软件"管理之窗"的每个子系统或部分在设计上都是动态的、开放的、相对独立的，如界面、数据库、索引、统计、报表打印等。同时它们又是有机地结合在一起、集成在一起的。由于在代码的编写上还没有做到全动态、全开放，所以整个软件还不是一个全动态开放软件，还不能靠其自组织形式来维持其生命力，靠自管理机制来做到面向发展、面向未来、面向其它的领域。同时也不能完全靠不懂软件设计或程序设计的用户在用户层次上的修改和维护来满足用户的某些特殊需要。

但是，从我们的实践和尝试中，我们认为按照动态开放设计思想和原理，设计并编写出全动态开放的软件系统和软件框架是完全可能的。

参考文献

1 Maruzzi S. The Microsoft Windows 95 Developer's Guide. Ziff-Davis, 1996

2 Roetzheim W. Programming Windows with Borland C++ 4.5. Ziff-Davis, 1994

3 Schildt H. C++: The Complete Reference. Second Edition. McGraw-Hill, 1995

4 Bertalanffy L V. General System Theory. George Braziller, Inc., New York, 1973

5 Haken H. Synergetic Computers and Cognition—— A Top-Down Approach to Neural Nets. Springer-Verlag Berlin Heidelberg, 1991

6 Swan T. Foundations of Delphi Development for Windows 95. IDG Books Worldwide, Inc., 1995

7 杰里米·里夫，特德·霍华德. 熵——一种新的世界观. 上海译文出版社，1987

8 G. 尼科里斯，I. 普利戈金. 探索复杂性. 罗久里，陈奎宁译. 四川教育出版社，1976

9 系统科学大辞典编辑委员会编(许国志主编). 系统科学大辞典. 云南科技，1994

10 汪应洛主编. 系统工程理论、方法与应用. 高等教育出版社，1998