事件空间与软件设计

Enent Space and Software Design

王蔚韬[1] 张建高[2]

（重庆建筑大学计算机科学系[1] 管理学院[2] 重庆400045）

Abstract In this paper, the dynamic technique of software designing and special interface designing is discussed. In order to design and build a dynamic interface which is opened completely and can deal with user's unknown demands, we put forward the event space theory of software designing and the method of analyzing events, where some essential basic concepts such as event base, event base library and so on are involved. Then the method is suggested how the event base library is build and based on which how an opening dynamic interface is designed.

Keywords Event space, Event base, Event base library, Isomorphism

1. 软件设计中的动态技术

提到软件设计中的动态技术，也许每一个软件设计人员都知道一些，也有不少的软件声称其界面是全动态、全开放的，但是，就我们所知，目前还没有一个软件在其所涉及的领域内在界面和其响应的事件上做到完全的开放性和动态性。当前，人们普遍认为的界面的开放性，从严格的意义上讲，只是界面花样的开放性和其响应的事件上的一种预定义动态性。我们可以称这种预定义动态设计为准动态技术，以区别于完全开放的动态技术。准动态技术的一种典型设计方式是下面的动态菜单和动态按钮的设计方式。

首先，界面上必须有一个菜单项或按钮，其功能是创建一个新的菜单项或新的按钮，而新的菜单项或新的按钮所能执行的操作或能响应的事件都是软件设计人员事先预定义好的，如一个动态菜单可以是下面的方式。

软件设计人预定义好若干的事件处理过程如下：

```
procedure NewCommand1Click(Send:TObject);
procedure NewCommand2Click(Send:TObject);
procedure NewCommand3Click(Send:TObject);
……
```

当用户在界面上定义一个新的功能"NewCommandk"时，软件设计人员所编的程序代码便执行下面的程序，创建菜单项事件触发器，赋给 TMenuItem 变量如 MI：

```
MI:=TMenuItem.Create(FileMenu);
MI.Caption:='NewCommandk(或用户定义的菜单名字)';
MI.OnClick:=NewCommandkClick;
```

这样一来，用户似乎就看到了自己定义的菜单项，并能执行软件中为用户保留的扩展功能。

对于不同的环境和不同的操作显示不同的界面，一般来说也是在软件中预定义好的，只是改变控制界面上各部件的可视性，诸如此类而已。

但是，对于任何一个应用领域而言，软件设计人员不可能预定义用户所希望的全部事件，也不可能知道用户所希望的全部操作，而且这种预定义的方式，也会使软件显得过于庞大，还会使一部分用户觉得软件太复杂。

为了解决上述问题，在本文中，我们将阐述事件空间理论的基本思想，提出事件基(Event Base)的概念，讨论事件分析方法和怎样建立软件的界面事件响应的事件基库。在我们阐述的事件空间理论的指导下，软件设计人员只要找出了软件所应用的领域的事件空间中的事件基，并预定义好基中的事件的处理过程或处理函数，同时建立少量一般事件表达方式的通用抽象处理过程或函数，便可以使应用软件几乎能解决应用领域中的所有问题，而且用户也可以定义他们所需要的菜单和按钮，这些菜单和按钮所处理的事情和所具有的功能是软件设计人员想不到的。

2. 事件空间理论

我们考虑任一应用领域，对于该领域中的一个动作或一种操作，称为一个事件，该领域中的所有事件之集合，称为事件空间。对于给定的一个应用范围而言，

事件空间中的每一个事件，一般说来都可以分解成其更基本的事件。换句话说，事件空间中的每一个事件都是由更基本的事件组成。例如，设置某段文字的颜色，可以分解成如下的事件：指定文本中的第 k 个字符，选择当前字符后的 r 个字符，选择一种颜色，将某种颜色指定给选择的字符。因此，如果我们设"设置某段文字的颜色"为事件 A，"指定文本中的第 k 个字符"为事件 B(k)，"选择当前字符后的 r 个字符"为事件 C(r)，"选择一种颜色"为事件 D，"将某种颜色指定给选择的字符"为事件 E，则 A 可以表示成：

$$A=B(k)+C(r)+D+E$$

其中的加法一般是不能交换次序的，我们称这种加法为有序加法或非交换加法。所以，事件 A 可以表示成事件 B(k)、C(r)、D、E 的有序和。

我们还可以定义事件的倍数或数乘，一个数乘以一个事件，其含义是：如果事件 A 可以表示成事件 B_1、B_2、B_3 的和，即：

$A=B_1+B_2+B_3$，其中 $B_1=B_2=B_3=B$

则说事件 A 是事件 B 的3倍，记为 A＝3B。我们称事件 A 是数3乘以事件 B 的结果。同理，可以定义数 k 乘以一个事件 B，称为事件 B 的 k 倍，记为 kB。类似地，我们也可以定义事件的其它运算。

我们用 Ω 来表示定义了运算的事件空间。

事件基 设 S 是事件空间 Ω 中的一个事件集合，如果 Ω 中的任何事件都可以用 S 中的事件来表示，并且 S 中的任何事件 A，除了可以被它自己表示自己以外，不能被 S 中的其它事件所表示，则称 S 为事件空间 Ω 的一个事件基。在这里我们所说的"表示"不一定是线性的，可以是非线性的，也可以是一般的函数关系表示。

给定一个事件基，我们称事件基中的事件为基本事件；不在事件基中的事件称之为非基本事件。显然，非基本事件都可以被基本事件所表示。

为了进一步研究事件空间，我们先介绍下面的同构性原理。

同构性原理 同构性和同型性，是指各种不同系统在结构上的一致性或相似性，这就可以用同一模式、原则、规律来描述完全不同的系统。同构性原理为把一个领域发现的规律合理地正确地过渡到另一些领域提供了依据。

按照同构性原理，事件空间一般应该与某一个抽象空间同构。如果对一个具体的事件空间，能够找出其同构的抽象空间和同构映射，那么就可以利用现有的数学理论建立事件基，并对事件基中的每一个基本事件，在软件中预定义界面响应事件(过程或函数)。我们称所有这些对应于事件基中的基本事件的软件界面响应事件(过程或函数)为软件的事件基库。

同时根据抽象空间中一般对象由基中对象所表示的表达方式，预定义抽象表达式的处理事件。这样一来，一个全动态全开放界面的基础核心框架便建立起来了。剩下的软件工作是多数程序设计人员都熟的了。

如果事件空间不能与一个抽象空间同构，而可以同构于抽象空间中的一个子集，也可以用抽象空间理论来建立事件基库和抽象表达式的处理事件，从而建立起软件的基础核心框架。

然而，对于许多应用领域而言，事件空间可能是很复杂的，一些事件可能还是模糊的，因此在某些情形下难以找到同构映射，这时，也可以直接分析事件空间，试着找出其事件基以及用基本事件来表示任一事件的表达方式。

对于一个给定的具体应用领域，建立其事件空间和事件空间到抽象空间的同构映射并不是一件简单的事情。对于同构的抽象空间，找出它的基和任何一个对象关于基的表达方式也有一定的技术难度，而这正是软件设计者和专业人员相互协作需要探讨、研究和努力去完成的任务。

按照事件空间的理论，软件设计可以看成一个多学科专业人员合作的具有反馈信息的系统。图1是对我们的设计思想的一个简单解释。

3. 更好的可靠性与可扩充性

用同构映射到抽象空间的方法建立的软件事件基库和软件的基础核心框架，有着比传统设计方式更好的可靠性和可扩展性，因为抽象空间中的基一般来说是完备的，抽象空间中的任一对象用基中的对象来表示的表达式也是严格的和精确的，有数学上的严密逻辑推理和证明作保证。因此，在这个基础上建立起来的事件基库有最少的必需的基本事件，它减少了软件设计中的代码编写量，使代码的重用性达到了最高的效率。同时，软件的基础核心部分也变得简单明了。如果该应用领域有了实质性的进展和发展，事件空间扩大，原有的事件基中需要添加新的基本事件才能成为新的事件空间中的事件基，那么扩充应用软件的事件基库以适应应用领域的新发展也很简单，只要对添加的新的基本事件设计软件中相应的响应事件(过程或函数)即可。在必要的时候，也可能需要修改处理事件表达方式的抽象处理过程或函数，或者增加一些对象的处理方法。

在传统设计方式中，由于设计人员不知道哪些是必需的基本事件，因此常常对许多看似不相似的事件定义不同的代码，而对看起来复杂的事件定义很复杂的代码。这一方面大大降低了代码的重用效率，另一方

面也降低了软件的运行效率和可靠性。同时，使应用软件有由大杂会堆积起来的感觉，缺乏逻辑性和层次性，既不利于调试和维护、也不利于扩充和扩展。当然，也不可能做到在事件应用上和界面上的全面开放性，因为总有设计人员想不到的，并且也不知道具体事件的表达方式，而这些事件可能是某些用户经常性使用的。

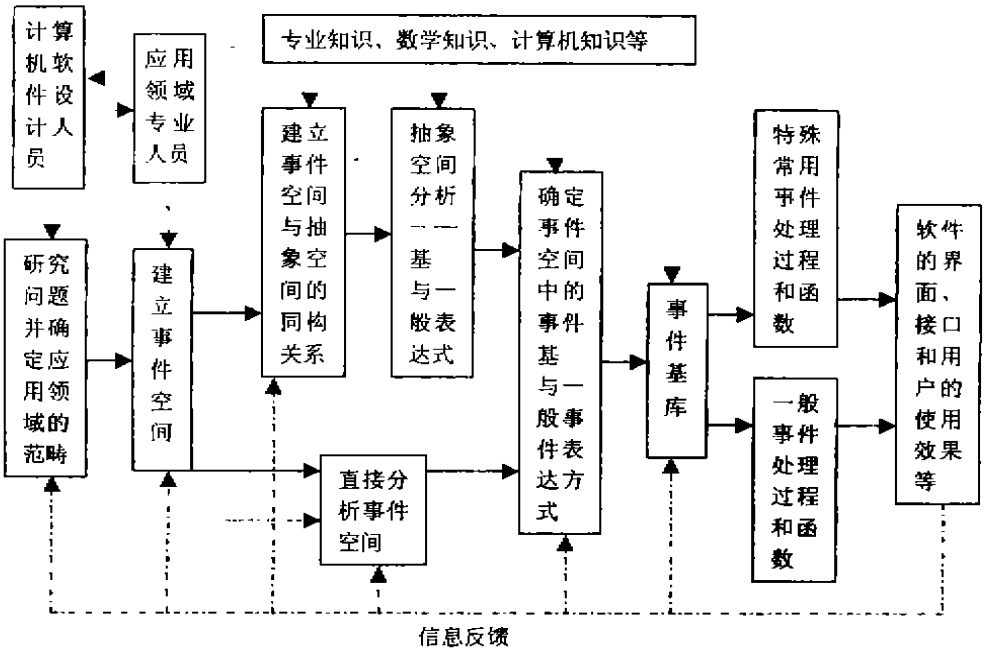

图1

4. 应用于经济评价软件中的模式

我们在国家"八五"攻关项目"污水处理 CAD"中设计软件的经济评价部分时，采用了这种设计方法。我们按照国家计划委员会和建设部发布的"建设项目经济评价方法与参数"对经济评价问题作了全面而细致的分析，直接定义了若干基本事件处理过程和函数，而每一个过程和函数在实质上都是不同的，由于时间比较仓促，当时没有定义有关"不确定性分析"中的图形显示部分内容的相关事件，因此不能将分析结果和数据转化为图形，所以我们定义的事件处理过程和函数构成了一个非完备的事件库。但是，我们定义了一个抽象事件表达式处理核心基类，其中的一般表达式处理函数可以处理经济评价中的所有定量评价内容和评价方法。此外我们还定义了一些常用的特殊函数，如"内部收益率"函数 IRR、"净现值"函数 NPV，等等。同时创造了一个"建设项目经济评价方法"的描述环境和模式，用户可以根据我们给出的基本事件和描述模式(与专业人员的自然语言相似)来描述自己的评价内容、方法和常用的事件，并可以给它们取名字以生成菜单项或按钮。以后，当用户点击这些菜单项或按钮时，就出现含有用户自定义的评价内容的基本数据输入框，用户输入基本数据后，选择运行菜单项或按钮，便得到用户要做的经济项目和特定评价内容的经济评价结果。

对于较熟悉计算机应用的管理专业人员而言，可以说我们设计的软件在一定的时期内完全解决了建设项目经济评价问题。

结束语 采用事件空间理论进行软件设计，使软件的内部逻辑结构变得更加清晰，层次结构更加分明；使代码的重用达到了更高的效率，提高了软件的可靠性和可扩展性；更容易做到界面的开放性和对用户的开放性。但是，这要求更多的更深刻的专业知识和数学知识。因此，我们希望软件设计者有更广的知识面和更深厚的基础理论功底。同时提倡多学科合作、协同作战以解决大型实际应用问题的精神和思想方法。

(下转第39页)

何利用需求分析成果来直接生成系统的体系结构说明。

◇**加强对进化的研究** 目前的 ADL 有些可以实现组件与连接子的进化，但这样的进化能力是有限的。其一，是因为这样的进化大多是通过子类型来实现的。其二，系统级的进化能力才是最终目的。系统级的进化必然涉及到系统族问题，对这方面的研究也将促进领域工程的发展。

◇**支持工具** 尽管现有的 ADL 都提供了支持工具集，但将这些 ADL 与工具应用于实际系统开发中的成功范例还很有限。支持工具的可用性与有效性较差，严重地阻碍了这些 ADL 的广泛应用。支持工具须在多视图、细化与回溯、跨体系结构层一致性检查等方面进一步加强。

参 考 文 献

1 Perry D E, Wolf A L. Foundations for the Study of Software Architectures ACM SIGSOFT Software Engineering Notes, 1992(Oct.): 40～52

2 Garlan D, Shaw M. An Introduction to Software Architecture: [CMU/SEI-93-TR-33]. Pittsburggh, Pa, SEI, CMU, Dec1993

3 Shaw M, Garlan D. Software Architecture-Perspectives on An Emerging Discipline. Prentice Hall, 1997

4 Garlan D, Shaw M. An Introduction to Software Architecture: Advances in Software Engineering and Knowledge Engineering, volume I. World Scientific Publishing, 1993

5 Luckham D C, Vera J An Event-Based Architecture Definition Language. IEEE Trans. on SE. 1995(Sep.): 717～734

6 Garlan D, et al Summary of the Dagstuhl Workshop on Software Architecture, February 1995. Reprinted in ACM Software

7 Garlan D. In: Proc. of the First Intl. Workshop on Architectures for Software Systems, Seattle, WA, April 1995

8 Wolf A L. In: Proc. of the Second Intl. Software Architecture Workshop (ISAW-2), San Francisco, CA, October 1996

9 Clements P C. A Survey of Architecture Description Languages. In: Proc. of the Eighth Intl. Workshop on Software Specification and Design, Paderborn, Germany, March 1996

10 Medvidovic N. A Classification and Comparison Framework for Software Architecture Description Languages, February 1996

11 Garlan D, et al. ACME: An Architecture Interchange Language Submitted for publication, 1996

12 Allen R. HLA: A Standards Effort as Architectural Style. In: Same to [8]

13 Luckham D C, et al. Specification and Analysis of System Architecture Using Rapide. IEEE Trans. on Software Engineering, 1995(April) 336～355

14 *Wolf A L. Succeedings of the Second International Software Architecture Workshop (ISAW-2).* ACM SIGSOFT Software Engineering Notes, 1997(Jan): 42～56

15 Shaw M, et al. Abstractions for Software Architecture and Tools to Support Them. Same to [13]

16 Garlan D, et al. ACME: An Architectural Interconnection Language; [Technical Report, *CMU-CS-95-219*]. Carnegie Mellon University, November 1995

17 王昕，金淳兆. 软件体系结构的研究与发展现状 计算机科学，1998，25(3)

(上接第53页)

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 汪应洛，主编. 系统工程理论、方法与应用. 高等教育出版社，1998

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

9 国家计划委员会、建设部发布. 建设项目经济评价方法与参数. 中国计划出版社，1995

10 吉田耕作. 泛函分析. 人民教育出版社，1981. 7

11 肯尼思 法内科尔. 分形几何——数学基础及其应用. 东北大学出版社，1993. 5

12 Balakrishnan. P V Hall N. G A Maximun Procedure for the Optimal Insertion Timing of Executions. European Jouranl of Operational Research, 1995, 85: 368～382