第3卷第4期
1992年11月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 3, No. 4
Nov., 1992

基于程序生成的软件过程模型

周善琼　居德华　　廖素萍

（华东化工学院，上海 200237）　（中国科学院软件研究所，北京 100080）

A NEW SOFTWARE PROCESS MODEL BASE ON PROGRAM GENERATION

Zhou Shanqiong and Ju Dehua

(*East China University of Chemical Technology, Shanghai* 200237)

Liao Suping

(*Institute of Software, Academia Sinica, Beijing* 100080)

Abstract CASE environment EASYCODE is a profitable exploration of new software process development paradigms, which have being increasingly attracted professional's attention to overcome the weaknesses of the waterfall model. After analysing the traditional lifecycle model in brief, the feature of EASYCODE software process paradigm is discussed in this paper, and knowledge-based program generation supporting the new paradigm is depicted as well.

摘要 随着计算机技术的发展，瀑布式软件开发模型日显不足，探索研究新的软件过程模型势在必然. CASE 环境 EASYCODE 对此作了有益的尝试. 本文在对传统的生命期模型作了简析后，讨论 EASYCODE 软件过程模型的特点，并阐述支撑该模型的基于知识的程序生成.

§1. 引　言

为了解决软件生产率和软件质量这两个基本的软件工程问题，人们在探索软件开发过程规律方面作了许多卓有成效的工作. 其中，影响最大的当属瀑布式模型的提出和发展. 该模型总结了先期的编码—修改模型(code-and-fix model)存在的问题[1]，将整个软件开发过程划分为需求分析、设计、编码、测试、维护五个阶段，并把软件开发前期的需求分析和设计视为关键活动. 根据该模型准则，在对系统未经充分的分析与设计，编写出完整的需求和设计说明书前，

本文 1990 年 10 月 17 日收到，1990 年 12 月 20 日定稿.作者**周善琼**，华东化工学院副教授，主要研究领域为软件工程. **居德华**，华东化工学院教授，主要研究领域为软件工程. **廖素萍**，中科院软件所助研，1988 年硕士毕业于中科院软件所，主要研究领域为软件工程.

不过早地进入编程实现阶段.这对提高软件生产率和软件质量,降低软件成本产生了明显的效果.但这种基于人的文档驱动方式也正是该模型的弱点所在.

·在系统开发初期,用户往往对系统只有一个模糊的需求设想,而提不出详尽的功能和性能要求.有时,虽然要求明确,但它却又处于经常变化之中.此时,按文档驱动准则编写需求说明书,煞费苦心却仍往往导致后阶段无效的设计和编码工作.

·根据手工书写的非形式化说明书,由开发人员完成阶段间的交接、转换工作.这种基于人的接口方式难以避免阶段缝隙与人为误解.

·在开发前期产生的手工文档不可执行,维护必须在源代码级进行.

传统生命期模型的缺陷源于它的问世时期,今天和七十年代初期相比,软件开发的客观环境已发生了惊人的变化,软件系统本身的复杂性及对软件的需求量也不可同日而语.这一切均使瀑布模型的缺陷日趋显露,继续沿用将导致软件供需矛盾的扩大,所以建立和采用新的软件过程模型已势在必然.

软件系统的开发涉及分析和综合两方面的工作.限于当时的技术水平,传统的瀑布模型偏重了基于人的分析过程,使软件开发的分析和综合工作呈现了不平衡.随着计算机技术的发展,在改进软件开发过程中,人们很自然地把目标集中于提高基于计算机的综合能力,把软件开发中原先由人完成的非创造性工作交给计算机实现,建立基于自动化和形式化的软件过程新模型.Balzer 等在 1983 年已指出[2],这是九十年代软件技术的方向所在.速成原型(prototype)、可操作描述(operational specification)和变换实现(transformation implementation)等都是近年来倍受关注的软件开发新范型(paradigm)[3].CASE 环境 EASYCODE 在探索这些新范型,建立软件过程新模型,实现软件过程自动化方面作了有益的尝试.本文讨论 EASYCODE 软件过程模型的特点,并阐述支撑该模型的基于知识的程序生成.

§2. EASYCODE 的概念抽象

EASYCODE 是一个适用于特定应用领域(DP/MIS)的集成化计算机辅助软件工程(CASE)环境,支持整个应用开发过程.它基于应用的概念抽象,在一套集成化工具主持下,将应用问题的描述自动变换成符合用户要求的可执行程序.EASYCODE 的开发模型、工具集成和过程自动化基于三类概念级抽象:

·应用抽象

EASYCODE 认为,一个应用问题总是一类将输入的对象变换成输出对象的活动.当然,这类变换活动往往因其复杂程度而需分层按序进行,从而构成一个复杂的多层变换活动网.但是,不管是顶层的系统、子系统,还是底层的基本模块,在抽象级上,它们同是对象的变换活动.这一抽象不仅适用于 EASYCODE 环境下开发的应用系统,也适用于其他类型的软件系统,乃至 EASYCODE 环境本身.

·对象类抽象

各种加工对象,无论是问题描述、代码、图,还是数据、表格等,在 EASYCODE 中均统一抽象为同一类对象——数表(form).

·对象结构抽象

任一数表,不管其外形如何,在 EASYCODE 中总被视为一个结构化对象,且总可用抽象

结构和具体结构对其作出完整的描述.对象结构用以表明数表内的逻辑关系.一个数表,无论其复杂程度如何,它的逻辑结构总可用三种基本结构加以描述:组合、选择和重复.具体结构用以定义数表的物理外形,可将其视为数表的"模板"(template).它由两部分组成,固定不变的"背景"和具体值待定的"变量".

EASYCODE 为描述这类结构化对象提供了下述形式的结构定义语言 SDL:

——抽象结构

组合　　$S:\{S_1,S_2,\cdots\cdots S_n\}$

选择　　$S:S_1/S_2/\cdots\cdots S_n$

重复　　$S:*\{S_i\}$

——具体结构

·S

* * * $\langle S_1\rangle$ * * *

* * $\langle S_2\rangle$ * * * * $\langle S_3\rangle$ * * *

·

·

·

$\langle S_i\rangle$

·

·

·

* * *

·

其中 S——结构或模板名,* * *——模板"背景",$\langle S_i\rangle$——模板"变量".

§ 3. EASYCODE 的软件过程模型

EASYCODE 支持如图 1 所示的软件过程模型[4].该模型把一个应用开发过程归结为应用定义、应用生成和应用执行三个阶段.用户仅需在应用定义阶段完成对应用问题的描述,系统不仅能支持从应用描述到应用程序的自动生成,还能根据应用描述自动生成测试计划和测试用例,帮助用户确认系统,并能在执行工具的支持下运行系统.传统生命期模型中的设计—编码—测试等阶段工作和多回路的分散验证维护模型在 EASYCODE 中变为定义—生成—执行—确认—再定义的单一确认维护回路.这一变更使软件开发过程的活动产生了实质性的变化.

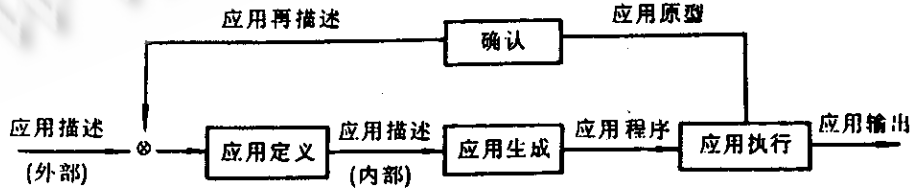

图 1　EASYCODE 软件过程模型

(1)使用应用描述定义应用问题

由应用描述来定义应用问题,使 EASYCODE 有别于瀑布模型:

·在用户概念级上进行开发活动

在EASYCODE中,软件开发成了一项非过程性的问题定义工作,而不再是过程性的代码编程,开发者的精力和时间可集中于分析应用问题、抽取反映应用实质的"事务逻辑",而不再拘泥于应付处理繁琐的开发实现细节.

·开发阶段的划分基点发生变化

在传统的生命期模型中,需求分析阶段说明系统的外部行为,设计阶段决定系统的内部结构,软件开发阶段划分的基点是"做什么"→"怎么做".而EASYCODE中,由于使用应用描述,阶段的划分基点转为"面向问题"→"面向实现".这种划分基点承认了"做什么"和"怎么做"两者的互相交融现象,较之传统模型,它更真实地反映了软件开发过程的客观规律,克服了将两者截然分离带来的许多问题[5].

·维护实施于描述级

在EASYCODE中,维护实施于描述级,而不再是源代码级,这对简化维护问题、降低维护成本,用户直接参与维护及提高软件质量等是至关重要的一步改进.

·最终用户(end—user)易参与开发活动

应用描述接近应用概念,反映用户观念,复杂程度低,最终用户稍加训练即可直接参与应用开发.

(2)自动生成可执行程序代码

EASYCODE中的应用生成是一类基于程序复用的自动变换实现,在软件知识库的支持下,应用生成器将应用描述变换成C代码程序.它使:

·手工的编码工作不复存在,大大减轻了软件开发过程的劳动强度.

·传统模型中那类基于程序的测试工作在EASYCODE中已无必要,取而代之的是对应用描述的确认,既简单又接近应用逻辑.

·系统的修改维护成为重定义——重生成过程,不存在传统模型中那种因反复修改代码而引起的系统可用性下降,甚至导致系统被废弃的危险.

(3)提供速成原型

应用描述和应用生成的结合使EASYCODE提供速成原型,缩短用户和开发者间的反馈周期成为现实.根据具体可见的系统样品,加之易为用户接受的应用描述,双方可对系统的需求进行充分甚至现场讨论,快速进行修改调整,真正实现以用户为中心的开发活动.这些均是传统模型所不及的.

(4)基于计算机辅助,实现软件过程自动化

EASYCODE提供应用定义、生成、执行、测试及管理等五大类二十多个工具,支持整个应用开发过程.各类工具的设计均基于同一概念模型,集成性好.在它们支持下,应用开发过程各阶段之间实现了基于计算机的自动无缝转接,根除了人工误转,显著地提高了软件开发过程自动化程度.

至今,EASYCODE已成功地生成了山东省肥城矿务局煤矿经营管理系统,上海市计算机应用情况统计系统及1990年第十一届亚运会成绩处理系统RIS(B)等多个实用应用系统.这些系统的开发,特别是RIS(B)的顺利生成,在提高生产率、降低成本等方面均明显地得益于EASYCODE过程模型的特点.提供速成原型,最终用户直接参加应用开发,程序知识复用和

适应频繁的需求变化等诸方面也深得用户好评，而这些正是探索软件过程新模型中引起人们共同关注的热点.

§4. 应用描述

EASYCODE 中一个完备的应用定义由三部分组成：描述系统结构的系统定义，描述数表结构的数表定义和描述基础活动输入、输出变换规则的活动定义. EASYCODE 相应地设计了三类描述语言：应用定义语言 ADL，结构定义语言 SDL 和映象定义语言 MDL. ADL 是一类基于数据流模型的图形语言，用与 DFD 图相似的数表流图 FFD(Form Flow Diagram)描述应用系统结构. 图 2 是一张 FFD 示例图，图中圆形代表活动，矩形代表数表. 若活动结点代表一个子系统，则它可进一步分解成下一层的 FFD 图，构成系统分层. 用 SDL 语言表示的数表结构定义如图 3 所示. 它描述的是一张有层次结构的预定单. MDL 是一类以输出表结构驱动的活动变换规则语言. 图 4 是一个基础活动的 MDL 变换描述示例. 它表示数表 u—i2 在该活动中既是输入表又是输出表，它的 price2 和 model2 项. 根据输入表 u—i1 的对应项 price1 和 model1，分别将值修改为 price2—price1 和 model1，其余项保持不变. u—i1 和 u—i2 二个均是层次表. 变换时，对应的操作关键字外层为 no1 和 no2，内层为 p—no1 和 p—no2. 变换后，输出表 u—i2 的记录，外、内层分别按 no2 和 p—no2 项值升序排列，对具有同一 no2 项值的记录，按 p—no2 项值分组组合(GROUPBY)，构成内层表.

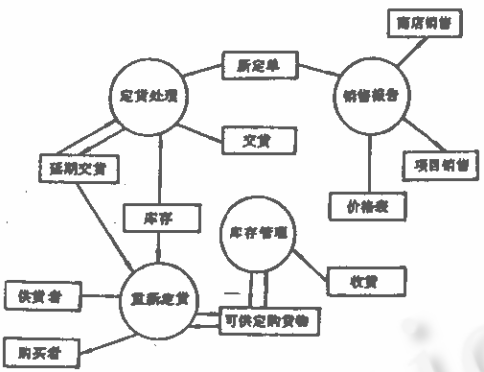

图 2　系统定义 FFD 图示例

```
u—i2 =u—i1 . no1 UPDATE u—i2. no2
    ? ASC no2 COUPLE u—i1. p—no1 u—i2. p-no2
  —no2
  —namc2
  —p ? ASC p—no2 GROUPBY p—no2
      —p—no2
      —p—item2
      —price2=u—i2. price2—u—i1. price1
  —model2=u—i1. model1
```

图 4　活动定义示例

抽象定义结构

```
order:      {order—no,date, items, customer}
order—no:TEXT(10)
date:       TEXT(8)
items:      * {catalog, qty, unit, price}
catalog:    TEXT(8)
qty:        INT(6)
unit:       TEXT(8)
price,      NUM(10. 2)
customer:   TEXT(10)
```

具体结构定义

• order　　预定单

定单号	〈　〉	日期	〈　〉
〈			〉
姓名	〈　〉	帐号	〈　〉

•

• items

类目	数量	单位	价格
〈			〉

•

• items—REC

〈　〉〈　〉〈　〉〈　〉

•

图 3　数表结构定义示例

§5. 程序知识

EASYCODE的应用生成能根据如前所述的应用描述生成达应用规模的程序系统.它是一类基于知识的变换实现,得到软件知识库的支持,库内的软件知识涉及目标程序语言知识,一般的程序知识、特定领域的概念知识和特定领域的通用算法.这些知识的获取采用程序设计示例方法(programming by example),总结特定应用领域程序设计经验,由具体到一般,从中抽取适用于特定应用领域的通用程序知识[6].和基于形式语言及定理证明的自动程序合成系统相比[7,8],这种自底向上的反向获取过程大大缩短了构造程序变换的周期,并能确保生成出满足用户要求的实际应用系统.前者虽更具数学严密性,但它目前还仅能合成实验性程序.

根据前述的概念抽象,由EASYCODE生成的应用程序也是一类结构化数表,因此,存放于软件知识库中的通用程序知识同样可用EASYCODE的结构定义语言SDL加以描述,由一组变换规则和程序模板组成,它们的描述形式如图5所示.目前,EASYCODE软件知识库中存有400余条变换规则和程序模板.

```
变换规则
        DataDec:{UserHead, FormDef}
        UserHead:/HasUserHead/Nothing
        FormDef:*{Form}
        Form:{SubColl,FormDec, I2dDec, KeyI2ds}
        SubColl:*{CollDec}
程序模板
        .DataDec
         #include "pglib.c"
         <UserHead>
         <FormDef>
        .
        .UserHead
         </>
        .
        .CollDec
         EXTERN struct <idno>el——<Name>
         {<SubEntries>
          <NextPointer>
         }*<idno>cp;
         <Pointers>
```

图5 EASYCODE程序知识描述示例

图5示例表明,程序模板由二部分组成,一是固定不变的正文代码,它是模板的"背景"(如CollDec模板中的EXTERN struct),另一个是以尖括号对"< >"括起来的待确定项(如CollDec模板中的<Subentries>),它们在变换实现过程中由应用生成器根据应用描述确定具体值.这是一种类框架(frame like)知识表示法,而变换规则则是一种类BNF(BNF like)规则表示法,它描述通用程序的逻辑框架.两种表示法的灵巧结合,使以SDL形式描述的

EASYCODE 程序知识具有良好的重用性.

§6. 应用生成器

EASYCODE 生成的 C 代码程序由四个模块组成：主控模块，描述系统的控制结构；数据说明模块，描述系统 I/O 数表的数据结构；I/O 模块，描述系统的输入和输出；变换模块，描述数表的变换规则.

与此对应，EASYCODE 的应用生成器包括四个模块生成器，它们有相同的变换机制和结构(图 6). 生成器接受二类输入信息，应用描述 SPEC 和程序框架 PF. 经语义检查器作一致性和完整性检验后的应用描述 SPEC 由 SPEC 变换器变换成 SPEC 结构树. PF 选择器根据 SPEC 信息从软件知识库 SKB 中抽取相应的变换规则和程序模板，构成适用于应用描述 SPEC 的程序框架 PF. PF 变换器将其变换成 PF 结构树. 然后，根据 SPEC 树，合成器展开或确定 PF 结构树中对应于程序模板中待填项的子树或叶结点值，建立语法实例树. 最后，根据语法树，解析器产生具体的程序正文.

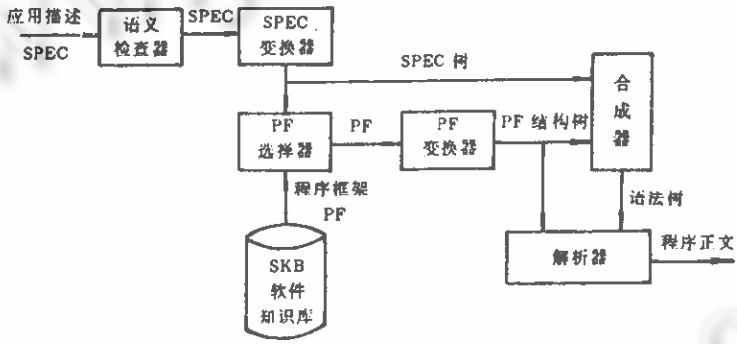

图 6　EASYCODE 应用生成器结构

EASYCODE 面向层次型对象结构，生成符合应用描述的完整的 C 程序. 考虑到应用的多变性，应用生成器按开放式结构(open architecture)设计，用户可在活动定义中插入 C 语句和调用用户自定义函数，方便地增加特定要求.

信息的外部描述虽然形式纷繁，但 EASYCODE 的信息抽象使应用生成器的处理方法得以一致，大大简化了应用生成器的设计. 此外，目前的应用生成器虽面向事务处理领域，但由于应用生成器的设计所基于的概念抽象具有通用性，在抽去面向特定应用领域的具体内容后，不难将该应用生成器框架发展成一个构造应用生成器的工具——应用生成器的生成器，拓展应用范围，实现更高层次的软件自动化.

结束语：经过几年的努力，EASYCODE 已由实验系统发展成一个实用的 CASE 环境，目前已有 SUN/UNIX 和 IBM PC/DOS 二个工作版本，并成功地生成了 1990 年第十一届亚运会成绩处理系统 RIS(B)等实际应用系统.

EASYCODE 所基于的概念抽象及对象描述方法统一了系统的设计基础，有效地促进了系统的集成，简化了系统的实现，并使系统不断方便地得以扩展和完善. 它基于程序的自动生成、应用速成原型、操作描述及变换实现等技术，探索了一类基于自动化的软件过程新模型.

EASYCODE 由华东化工学院软件工程研究室和中国科学院软件所合作研制. 多年来,得到原国家科委中国软件技术开发中心的关心与支持. 在应用 EASYCODE 生成实际应用系统的工作中,第十一届亚运会计算机工程分指挥部及山东省肥城矿务局等单位给予了大力支持与帮助. EASYCODE 项目规模颇大,先后涉及了项目开发人员三十余名,仅应用生成器部分就汇集了八名成员的工作,所以它是课题组全体成员默契配合、集体奉献所得. 作者借此向这些单位及所有成员一并致以深切的谢意.

参考文献

[1]Boehm B W, Computer, 1988; 21(5):61.

[2]Balzer R, Cheatham T E. Computer, 1983; 16(11):39.

[3]Agresti W W, New Paradigms for Software Development, IEEE Catalog No EH0245-1, 1986:1.

[4]Ju Dehua, Zhou Shanqiong, Proceedings of Software Symposium'89 in Japan. 1989:269.

[5]Swartont W, Balzer R, Communications of the ACM, 1982; 25(7):438.

[6]Hrycej T, SIGPLAN Notices, 1987;22(2):53.

[7]Goldbery A T IEEE Transactions on Software Engineering, 1986; SE-12(7):752.

[8]Partsch, Sleinbruggen, Computing Surveys, 1983; 9:199.