# 基于Ada的软件开发新模式

陈火旺　　齐治昌　　张孝秋

（国防科技大学计算机系）

**摘 要**

本文分析了Ada程序设计支持环境APSE在大型嵌入式软件开发过程中的成就与不足，论述了基于Ada的自动化软件开发新模式，讨论了实现这一模式的途径和面临的困难。

针对大型嵌入式应用软件的开发，美国国防部用了五年时间，耗资上亿美元，于1981年推出了Ada语言[1]；并于1980年公布了与Ada语言配套的石人计划[9]，它指出了对Ada程序设计支持环境APSE的要求；随后制定了从1983年开始，为期十年的STARS计划[2]。建立一个可移植的、用于分布式环境、基于A代码的APSE是STARS计划最后五年的重要任务。人们预料，STARS计划的执行可以将美国军用软件生产率提高四倍以上。

APSE由数据库、软件工具集、用户与APSE的软件接口[3]以及各种软件之间的接口组成。与以往的高级语言相比，Ada语言有一个与它配套的、集成化的支持环境APSE[4]，这无疑是个很大的进步。在APSE上进行Ada软件开发的典型模式如图1所示。

使用这个开发模式至少有以下几个优点：

（1）Ada软件开发与维护在一个集成化的环境[8]中进行，这个环境有标准的接口，完整的工具和良好的人机界面。

（2）有标准的中间代码，如DIANA码[12]，支持Ada编译器、语法制导编辑器、美化打印程序，为这些软件的移植创造了条件。

（3）数据库支持Ada软件开发的始终，支持Ada应用软件的可重用性。

（4）在DIANA码上建立起来的语法制导编辑器、调试程序和美化打印程序支持生成正确的Ada源程序和相应的DIANA码，提高了编程效率。

（5）Ada语言的低级特征和时钟。通过APSE映照为目标机物理控制信息和物理时钟，支持Ada语言控制功能和实时性。

（6）APSE和目标机操作系统支持Ada语言的任务机制和并发性。

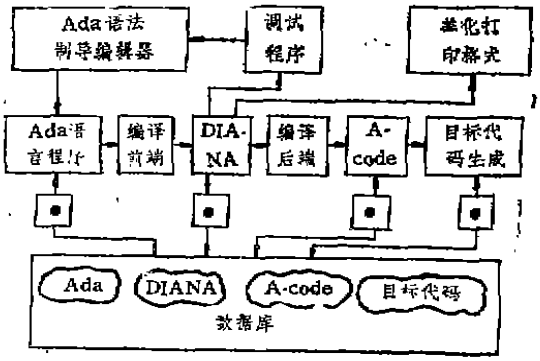


图1　Ada软件开发的典型模式

这些优点表明，APSE从方法学和工具两方面有力地支持Ada软件开发。

然而，大型软件开发的实践告诉我们，提高软件生产率、增强软件产品的可靠性、改善软件产品的质量、解决当前面临的软件危机，单靠目前的程序设计支持环境是不够的。因为这个环境：

（1）不支持用户参予系统设计；

（2）不支持系统的可行性研究；

（3）系统维护停留在程序代码上，而代码和规范的不一致很难判断；

（4）在系统开发出来之前，没有给用户提供相应的原型供用户参考。

针对APSE存在的这些弱点，我们试图在APSE基础上，建立软件开发新模式[5]；该模式由需求分析工具、支持原型技术的工具、形式化规范及相应的变换工具、软件环境信息库组成；能够自动或半自动地支持开发大型嵌入式应用软件。这种开发模式如图2所示。

这种开发模式的优点是：

（1）由于采用原型技术[6][7]，用户能够参予系统设计。

（2）需求或规范阶段的错误可以通过原型及早发现，回避了因系统功能或概念上的失误给整个软件开发带来的损失。

（3）采用形式化规范，规范到程序代码的转换自动进行，保持需求、规范、程序代码的一致性。为系统管理提供有利条件。

（4）系统无需测试，维护可以在需求级或形式规范级进行。

（5）提高系统的可靠性。

（6）自动地开发文档。

（7）系统维护采用重播方式。

（8）采用这种模式，大量与用户和程序员见面的是需求说明语言，这表明程序设计语言又提高到了一个新阶段。

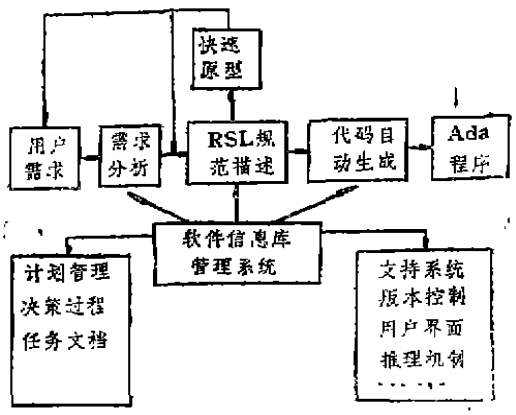


图2 软件开发新模式

下面从需求分析、形式规范、快速原型及软件信息库等方面，进一步阐述图2所示的开发模式。

## 一、需求分析与需求说明语言

需求分析的任务[11]是，从用户要求生成需求说明。其分析流程如图3所示。

**1.需求** 需求包括：用户期望系统完成的功能，系统设计约束。需求分为功能需求和非功能需求。功能需求描述软件的输入输出行为；非功能需求描述系统的简明性、可扩充性、可维护性、可靠性、用户接口等等。

**2.需求说明语言** 需求的描述一般是非形式的。需求说明可以用多种形式的语言描述。显然，人们最容易接受的是自然语言。自然语言不仅十分复杂，而且还存在二义性。于是，人们往往使用下面几种需求语言。

（1）受限自然语言 如，结构英语。

（2）形式语言 用某些词、短语及某种文法定义的关系语言。如，Teichoew和Hershey的PSL。

（3）基于表格的专用语言 使用这类语言比形式语言容易，但应用范围很窄。

（4）图形语言 如美国和以色列联合开发的ST-ATEMATE，它不仅能描述用户需求，而且能自动生成Ada目标系统。

无论需求说明语言采用什么形式，它必须满足以下几个条件：

（1）可理解性 可读性好，无二义性，适用于不同的读者。

（2）可分析性 能够预测设计决策的影响，模拟系统行为，检查各种描述的等价性、一致性、有效性。

（3）可维护性 程序及需求说明必须支持系统整个开发过程，具有良好的可修改性。

（4）可验证性 需求说明语言应便于验证需求说明是否与用户的需求一致；需求说明语言应采用适当的软件结构模型，以简化由非形式需求建立形式规范的过程。

**3.需求说明语言的模型选择** 每种需求说明语言都选择一种模型反映程

序的行为。通常，这些语言多采用控制流模型或数据流模型。选择模型的原则是：根据应用领域的特点，易于将这些需求语言描述

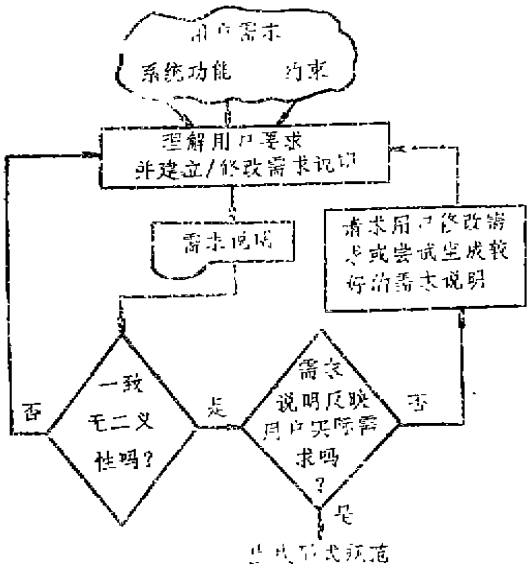


图3 由用户要求产生需求说明的过程

的抽象概念形式化。

我们认为，采用面向对象的需求说明语言模型，符合人们分析问题的习惯，描述软件需求比较自然、合适。

**4.需求分析** 我们试图利用面向对象的图形需求说明语言GRSL，描述大型嵌入式软件系统的需求。需求分析的任务是：

(1)系统分析员理解，并用GRSL语言描述用户需求。

(2)对需求说明进行验证，检查由GRSL描述的需求说明是否反映用户的要求，是否满足完全性、一致性等等。

(3)自动生成文档。

(4)能够指明资源上的约束。

支持上述需求分析的软件工具主要有：

(1)需求说明编辑器(RSLE) 系统分析员利用它把用户需求转变为需求说明，并将有效信息装入软件信息库。

(2)需求分析报告器(RSARG) 对由GRSL描述的需求说明进行整理和分析。检查系统的不一致性和不完全性，输出描述性报告、列表报告、分析报告、图形报告，便于生成形式规范。

## 二、 形式规范语言RSL

**1.形式规范与规范语言的选择** 采用自动编程技术是提高软件生产率的有效途径。而形式规范是确认系统、实现自动编程、对系统进行维护的基础。

系统设计员借助形式规范语言的结构编辑器等支持工具，将需求分析的最终结果，如需求说明的各种报告、表格、图形等等，转换为形式规范。

支持大型软件开发的形式规范语言应具有下面几个特性：

(1)有严密的数学定义。

(2)有强的表达能力。

(3)支持原型的开发。

(4)与需求分析语言的设计风格相协调，便于从非形式的需求说明转换为形式化的规范说明。

目前流行的基于代数的规范语言如Clear[15]，CIP-L，Larch，ASL虽然有严密的数学定义，但太抽象，用户和系统设计员都难以接受，自动生成目标代码困难很大[16]。基于模型的规范语言如Meta-IV[17]，表达能力较弱；对并发、模块化不提供支持；开发大型软件有困难；缺乏严格的数学语义和抽象能力。面向进程的规范尚不成熟。

我们认为，比较适合大型嵌入式软件开发的规范语言可能是欧洲共同体ESPRIT计划中的RAISE[18][21]（Rigorous Approach to Industrial Software Engineering）课题所采用的规范语言RSL（RAISE Specification Language）。RSL有严格的数学定义，表达能力强；面向实现；支持大型软件的开发。另外，RSL还支持基于可重用软件的原型开发，能与面向对象需求说明风格相协调。RSL的缺点是语言比较庞大，实现自动生成工作量大。

**2.规范语言RSL** RSL是面向模型的规范语言，它以VDM的元语言为基础，借鉴了Clear，OBJ，ML，CSP和Occam而形成。该语言提供显式或隐式规范，并可用作用方

式和施用方式表示这些规范，其主要成分有结构、类型、值、变量、操作和进程。

（1）结构（Structures） 结构是RSL中的构造块和抽象单位。在结构中，可以定义类型、值、变量、操作和进程这些实体。从语义上讲一个结构由一个标志(Signature)和一个模型类构成，其中标志给结构中的实体关联某些类型信息，而每一模型给结构中的实体关联某些数学对象。一个模型构成了一个环境，可以在环境下解释结构中的表达式和语句的含义。可以将模型看成一个实现的抽象。由于规范的不确定性（Underspecification），导致了结构具有模型类，而不是一个模型。

在RSL中可以用多种方法定义结构：

· 平坦结构(flat) 它仅封装(encapsulation)一些实体(如类型、值、变量、操作和进程)。

· 分层结构(layered) 它利用已定义的结构来定义新结构。分层结构可看作参数化结构，支持大型系统的描述。

· 结构的替换(substitution) 它是通过替代某分层结构中所使用的另一结构产生的。

· 组装结构(fitted) 它是对某结构中的实体换名、隐藏、拷贝而产生的。

（2）类型（Types） 从语义上讲，类型可看成对非空值集合的说明。RSL中提供了两类类型定义：

· 抽象类型定义 类型被命名，但不显式地给出该类型的说明。该类型的说明用对该类型实施操作的实体(如常数、函数、操作和进程)描述。

· 显式类型定义 类型通过等价的类型表达式定义。类型等价通常是按结构等价，带标记的类型定义按名等价。VDM中递归的类型定义在RSL中也是允许的。类型表达式可以是预定义类型名或抽象类型(如实型、布尔型、字符型)以及子类型按某些规则(如笛卡尔乘积规则)构造而成。

（3）值（Values） 值定义用于定义值以及相应的类型。值可以通过表达式的语义来显式定义，也可以通过公理给出值的隐式描述。

RSL中值表达式十分丰富。这些表达式对应于原子值、构造类型的合成和分解值、操作和函数应用所得到的值以及条件值。

（4）状态和操作（States and operations） 状态是值与变量的一种关联。结构中通过定义变量引出了状态，通过拷贝一个结构可产生该结构的不同状态事例。对外部结构来说，其内部结构的状态是不可见的。通过调用定义在结构中的操作，可改变和控制该结构的状态。

RSL用赋值、合成、调用、循环等语句描述操作，也可通过前、后置条件给出操作的描述。操作的返回值用值表达式描述。

（5）进程（Processes） RSL中，并发活动可通过进程（基于CSP)概念来描述。一个进程可看成一个实体，它能够(1)沿(单方向）通道和其它进程进行通讯；(2）访问变量。进程由进程语句和非进程语句描述。进程语句包括原子进程、进程间通讯、进程选择、进程的并发合成、进程交叉、通讯隐藏、参数化进程的引用、通道换名以及字母扩展（alphabet extensions）等等。

## 三、 RSL形式规范到Ada目标代码的开发[14]

**1.目前采用的技术** 从形式规范到目标代码的自动生成是软件自动化的核心和难点。程序自动开发的研究与应用目前尚不十分成熟。目前采用的方法有下面几种：

（1）逐步开发法，对现有的规范$S_1$构造下一规范$S_2$，构造的每一步是数据求精或者是操作求精。为了验证每一步构造的合理性，必须明确定义"$S_2$正确地实现了$S_1$"的准确含义。目前常见的方法是引入所谓的"抽象/恢复函数"。

（2）变换规则法 系统提供正确的、足够的、充分的变换规则集，对规则的使用条件加以验证；当用户根据需要选择一种变换规则后，系统就可完成变换。

（3）增长开发法 它便于系统扩充，但

要求语言能处理部分规范。

(4) 层次开发法　该方法操作性强。

我们认为：在尽量采用较成熟的变换技术的同时，用构造加验证的方法去补充其不足；以及为不同类型数据或操作求精提供相应的证明规则可能是一种行之有效的方法。

**2.RSL与Ada**　从RSL形式规范自动生成Ada目标程序，有两个有利因素：

(1) RSL面向实现。RSL借鉴了许多程序设计语言的特性，这些特性本质上是可执行的，它们某些部分可解释执行。易于自动生成Ada目标系统。

(2) RSL形式方法的构造理论和Ada程序设计语言相协调，它与Ada语言有一致的表达能力和通用性。RSL语言中所提供的参数化结构与Ada中的类属概念相似；RSL中的结构可用Ada中的程序包来实现；RSL的并发机制与Ada中的任务有类似之处。

**3.支持环境**　从形式规范到目标系统的生成需要支持环境，支持环境分为两类。

(1) 流行的环境　主要包括：

- 数据库及数据库管理系统
- 定理证明器　验证变换的正确性、规范的有效性以及检查语义条件都必须使用定理证明器。

(2) 人工智能环境　主要包括：

- 知识库及专家系统
- 借助于启发式推理的定理证明器。

## 四、速成原型

**1.原型**　原型是可执行模型或者是目标系统的初始版本。产生原型的快速构造活动称之为速成原型[19]。速成原型是确认需求，展示系统重要特性的必要工具。通常，原型展示的系统特性有：人机界面；基本框架；模块界面；基本功能；I/O格式；时间约束等。通过展示这些内容支持用户和系统设计员共同参与系统设计。

**2.方法**　目前开发原型的方法[20]有如下几种：

(1) 可重用方法　利用成熟的软件进行开发，这样可降低软件开发费用，提高开发过程的速度，减少一些必要的测试。

(2) 可执行规范　通过模拟或直接执行规范的方法展现目标系统的功能[21]。

(3) 程序变换方法　将形式化的规范变换成可执行的代码。

(4) 其它，如脚本式设计、增量式设计、演化式设计、以及逐步求精设计方法。

**3.环境**　基于可重用软件的自动原型开发有着广阔的前景。我们企图构造一个集成化的软件工程环境，以RSL规范为基础支持大型嵌入式系统的原型开发，它包括下面三个方面：

(1) 原型语言PSDL (Prototype System Description Language)　该语言有较高的抽象级别，用于描述嵌入式实时系统的计算模型。

(2) 原型构造的系统设计方法　基于模块化技术和可重用软部件。

(3) 原型支持工具　包括四部分：

- 软件库　检索和选择可重用的软部件。
- 执行支持系统　示范和估量原型的特性
- 设计者界面　如语法制导编辑器。
- 设计数据库　存放设计与版本信息等。

值得指出的是：RSL规范语言所提供的抽象描述（如标志(Signature)）与软件可重用部件的抽象描述形式可以统一。有利于软部件的检索和使用。

**4.问题**　需要进一步研究的问题有：

(1) 如何根据目标系统多方面的特性做出比较简洁的原型。

(2) 能否支持需求级、规范级等多级别上的原型。

(3) 采用什么样的方针指导原型的评估过程。

(4) 如何在原型开发中利用人工智能的研究成果。

## 五、软件信息库

**1.内容**　软件信息库应支持系统开发的

全过程，需要存放和管理的信息有：

（1）支持整个软件生命周期的基本信息；

（2）需求说明---RSL形式规范---Ada程序的变换信息；

（3）Ada目标产品历史信息，用以开发和维护产品，产生必要的文档；

（4）软件产品评估信息，用于软件产品的功能分析、管理分析和实用分析。

**2.要求** 对软件信息库的要求是：

（1）存放和管理上述的信息；

（2）提供版本控制，并支持配置管理；

（3）管理软件开发中的过程信息。

**3.信息模型** 软件信息库的核心是建立信息（数据）模型，用于描述各类信息。我们先定义一些信息构造单位（如：Type，Class，Entity，Relatioship和Event），然后用它们建立更高级的软件信息概念。

软件信息库管理系统至少包括三方面的内容：

（1）软件信息库（SIB） 其主要框架如图4所示。

（2）基于图形的用户界面 包括人机工程和多窗口技术。

（3）环境扩张能力 可以对软件信息库支持环境进行扩充，为SIB增加新的对象类型。

软件信息库的研究是一个较新的课题，应充分利用现有的环境和工具进行开发。

当前以软件生产自动化为基础的软件开发是一个引人瞩目，难度较大的课题。虽然人们还面临着软件工程方法学和形式数学推理在软件开发中的许多困难，需要深入地进行大量的研究工作。但是，仍有许多学者正致力于收集总结这个领域里的理论、方法和技术，并用于工业界。欧洲共同体以RSL规范语言为基础的RAISE课题就是其中的一个例子。我们相信，借助于计算机辅助软件工程(CASE)[23]及人工智能的研究成果，研究现有软件开发方法和技术，必将会促进和丰富我们对Ada软件自动化的开发[24]，完善基于Ada的软件开发新模式。

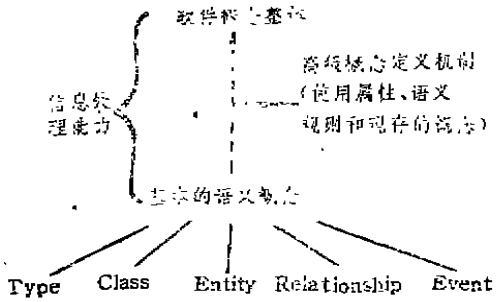


图4 软件信息库（SIB）框架

## 主要参考文献


[1] Reference Manual For the Ada Programming Lauguage, U S D o D Ada Joint Program Office, 1982.7.

[2] Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy", Mar.1983.

[3] G.Goos, J.Hartmanis "Ada Software Tools Interfaces" Lecture Notes in Computer Scince 180, 1984.

[4] "Configration Management and the Ada Programming Support Environment".

[5] R.Baizer, T.E.cheatham "Software Technlogy in the 1990's: Using a New Paradigm" IEEE.Computer, 1983.11.

[6] LUQI"Research Aspects of Rapid Prototyping" 1987.3.

[7] T.W.G.Docker, "A Flexible Software Analysis Tool" Information and Software Technology".Vol.29,No.1,1987.

[8] Mark Dowson. "Integrated Project Support With Istar", IEEE Software November 1987.

[9] Requirements for Ada Programming Support Environments, STONEMAN, DOD 1980.2.

[10] G.Booch "Software Engineering in Ada", Seoend edition, 1986.

[11] R.J.Abbott, D.K.Moorhead "Software Requirements and Specifications : A Survey of Needs and Languages" The Journal of Systems and Software 2.1981.

[12] G.Goos, W.A.Wulf "DIANA : An Interrnediate Language for Ada" Lectures Notes in Computer Science, Vol. 161, 1983.

[13] M.Rueher, "From Specification to Design : An Approach Based on Rapid Phototyping", Fourth International Workshop on Software Specification and Design, 1987.

[14] H.K.berk "Formal Methods of Program Verification and specification" 1982.

[15] Burstull, R.M., Goguen, J.A. "The Semantics of CLEAR, A Specification Language", Lecture Notes in Computer Science, Vol.86, 1980.

[16] Guttag, I.V. "Notes on Type Abstraction", Proc. Specification of Reliable Software Conf., 1979.

[17] Bjorner, D. and Jones, C.B., "The Vienna Development Method: the Meta-Language", Springer-Verlag 1978.

[18] J.Jprgenson, S.V.Plam "Preliminary Definition of the RALSE Specification Langusge", Esprint 315 1988.

[19] Welch, T."Rapid Prototyping System", 1986.

[20] D.C.INCE, S.HEKMATPOUR, "Software Prototyping Progress and Prospects", Informations and Software Technology.Vol 29, No.1, 1987.

[21] G.MICHAELSON, "Interpreter Prototypes from Language Definition Style Specication", Information and Software Technology. Vol.30, No.1, 1988.

[22] Erin meiling, Chris W.George, "The RAISE Language, Method and Tools", 1987.

[23] Wayne K.Frey "Computer Aided Software Engineering (CASE) Environment ISSUES" 1987.6.

[24] 齐治昌"软件开发新模式对APSE的影响" 计算机工程与科学 1988.4.

（上接68页）

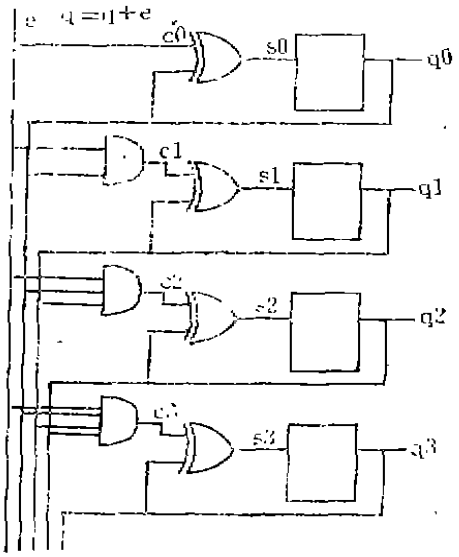


图5 计数器

用图4替代图2中的加法器就得到了图 5 所示的完整而详细的电路，后者就是众所周知的同步二进制计数器。我们特别注意到在组合电路中的等式s=q+e与顺序电路中的赋值q:=q+e之间微妙而根本的区别。

上述内容的寓意是，电路应该用正文形式定义，而实际电子电路可以据此按照一组固定的规则得到。对物理现象的模拟仍然需要，但可以把它简化为单个时钟周期期间的信号传播，可以把它简化为有限的几种情况。正文形式必须用硬件描述语言给出。最重要的是，这种语言必须经过严格定义，可以使用数学变元。其用途不是实现模拟，而是进行证明。

情况似乎是这样：不仅可以用适当的体系结构支撑程序设计语言与操作系统，而且反过来也可以用程序设计语言支撑体系结构的设计。将来大概不只是支撑设计，而且有可能用程序设计语言设计体系结构。

参考文献（略）

〔徐宝文 译自《ACM SIGPLAN Notices》1987, Vol.22, No.10, 示 瓦校〕