计算机科学2001Vol.28№.6

基于UML和SDL的实时通信软件建模方法

UML and SDL Oriented Realtime Communication Software Modeling Method

龚如宾　徐盛林　潘金贵

（南京大学计算机软件新技术国家重点实验室　南京210093）

Abstract　This paper introduces a realtime communication software modeling method called RCSM (short for Realtime Communication Software Modeling). RCSM method combines the UML OO analysis technique together with the SDL design technique to support the applications in the area of realtime communication systems. It is based mainly on UML and SDL notations. It can satisfy the complete requirement from requirement analysis modeling to automatic code generating and test case generating in the realtime communication software development domain. RCSM can satisfy the rapid development of various communication services and make the higher quality software and shorter development period possible.

Keywords　UML, SDL, Realtime communication software, Modeling method

1　引言

早期的实时通信软件使用结构化的方法来进行分析设计。由于采用结构化方法分析设计的软件系统在可重用性、可修改性等方面的局限，研究人员逐渐开始尝试使用面向对象的方法来分析和设计实时通信软件。面向对象的分析方法使用直观的面向对象的概念（如类、继承、聚合、相联等）来描述现实世界中的对象，其优势是可以从总体上理解和把握问题，不用事先给出关于目标软件系统的完整、一致和无歧义的描述，目标软件系统的细节问题留到设计和实现阶段去解决。近年来出现的UML（Unified Modeling Language）[1]语言以面向对象技术为基础，在通用软件建模领域中可以应用于从软件需求分析到设计编码的所有阶段。但在实时通信软件建模领域中，需要考虑时限性、可预测性和并发处理等实时系统独有的问题，因此需要对目标软件系统的各方面，包括体系结构、接口和行为等给出完整的、形式化的描述，而UML语言却是半形式化的语言，不能完成实时通信软件设计阶段形式化建模的重任。在通信领域广泛使用的SDL（Specification and Description Language）语言是一个基于扩展有限状态机的完全形式化的语言。它可以产生清晰的模型，产生完备的可执行的代码，并能进行模拟执行和自动产生测试用例。另一方面，SDL又是一个高层语言，这使得设计者可以集中精力于应用问题领域的设计，而不需要考虑底层的编程。

本文提出了面向对象的实时通信软件开发方法RCSM，它集中了UML在面向对象分析和设计与SDL在实时通信领域形式化描述方面的优点。RCSM方法在系统开发的需求分析和系统分析阶段使用UML来建模，而在系统设计和对象设计阶段使用SDL来建模。RCSM的目标是以面向对象的方式完成实时系统开发中从需求分析到设计实现所有阶段的模型创建和转换工作。

2　RCSM方法的基本概念

RCSM方法将软件开发的主要阶段很好地衔接起来，从需求分析阶段到目标软件的测试执行阶段，模型之间过渡紧密自然。RCSM方法又是一个开放的方法，各个阶段可以融入最新的技术和方法，同时能使用通用的语言简洁地表达静态或动态模型。下面介绍RCSM方法中涉及的主要概念。

2.1　关于实时系统

面向对象技术在实时通信领域的运用仍然有限，因为实时系统有很多独有的特征，比如时限性、可预测性和并发调度等，其中解决好实时通信系统的并发调度问题尤其重要。传统的调度算法主要有RM（Rate Monotonic）调度理论和最早时限优先算法（Earliest Deadline First）等，使用面向对象的方法来进行通用实时系统应用软件的开发，最困难的问题是把对象的概念和并发的概念结合起来。传统的面向对象的并发模型分为两种：显式并发模型和隐式并发模型。显式并发模型完全从并发的角度出发考虑问题，而不是从应用的角度出发，在一定程度上破坏了面向对象的概念。隐

式并发模型在开发早期，只考虑理想化的并发对象模型，而在设计和实现阶段考虑对象和进程的结合。RCSM 方法中将采用隐式并发模型技术，以更好地支持面向对象的分析和设计技术。RCSM 方法在系统设计和对象设计阶段使用基于 SDL 的建模方法，可以充分利用成熟的技术来支持将 RCSM 中的对象模型映射到通用实时操作系统的进程或线程中去，同时决定了对象间的同步或异步机制在系统中的具体实现问题，从而解决了面向对象实时系统开发中对象与进程的结合这一难题。

2.2 关于 UML 语言

UML 是一个通用的建模语言，可以适用于广泛的软件开发领域，甚至很多 UML 表示法可以适用于实时系统软件建模。但是实时系统有很多独有的特征，比如时限的严格性、可靠性和可预测性等，在这些涉及具体实现的方面，由于 UML 的半形式化特性，完全使用 UML 来为实时通信软件建模有很多不能充分表达建模需要的地方。OMG 已经认识到 UML 在实时领域的缺陷，创立了 Real-Time Analysis and Design working group(RTAD)[2]，目的就是扩展现有的 UML 以支持实时软件开发。但是一个没有良好定义的形式化语义的语言将无法支持高级的特性，比如模拟执行、代码自动生成和自动测试生成等。

2.3 关于 SDL 语言

SDL 语言完全形式化的特性[3]使得它可以在 UML 不适合的实时通信软件的设计与实现领域充分发挥作用。SDL 语言可以形式化地描述包括体系结构、接口和行为在内的实时系统软件的所有行为。由于历史的原因，SDL 长期用于传统的结构化实时系统设计，SDL-92标准开始尝试引入面向对象的开发方法，已经取得了一定的成果。但是完全使用基于 SDL 的面向对象开发也有其不完善的方面，比如：①由于引入的类型(相当于面向对象中的类)的种类多达7种，彼此重点不同，缺乏一个一致的类型概念，因此无法使用通用的面向对象的方法来划分类型中的属性/行为的界限；②与系统通信相关的信道和信号路由在其面向对象的概念中没有定义；③类型的专门化从结构、行为和数据三个方面来考虑，不同于通用的面向对象语言的操作(方法)重载机制，对封装和信息隐藏缺乏考虑。因此，完全使用 SDL 来进行面向对象的软件开发不能充分发挥面向对象方法的优势，而适合在软件开发的设计实现阶段进行形式化的描述。

3 RCSM 方法的开发过程

RCSM 方法可分为五个阶段，包括需求分析、系统分析、系统设计、对象设计和实现阶段。其中每个阶段都接受来自前阶段的模型，通过变换和创建工作，产生本阶段的模型，并作为后阶段的输入。各种模型之间的转换操作可以由一系列的工具来支持完成。图1表示了RCSM方法中的各个阶段和相关的主要模型。为了

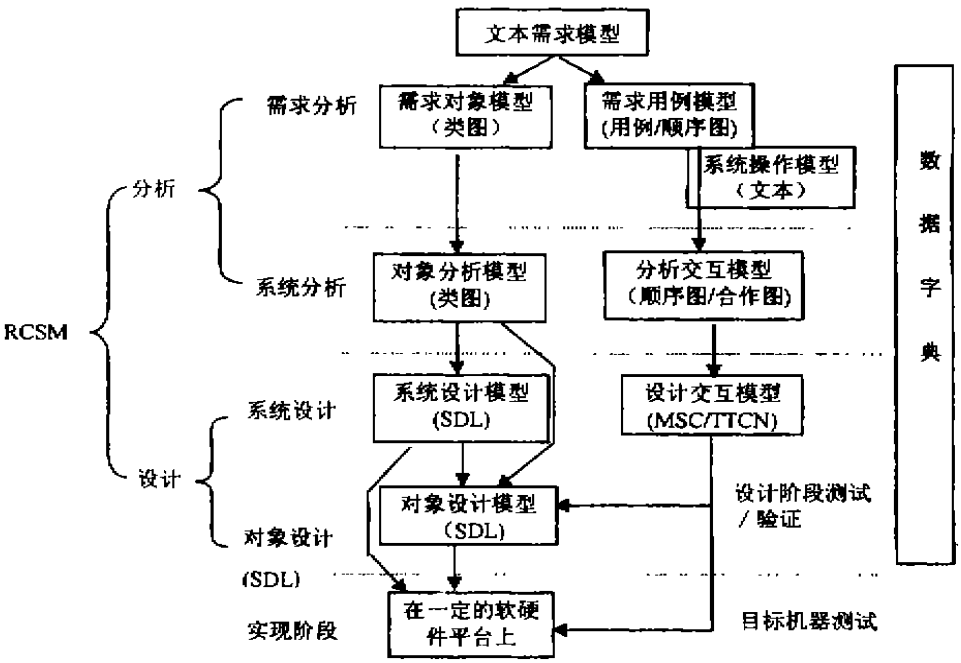

图1 RCSM 的主要模型和步骤

进一步说明 RCSM 方法，我们给出一个局内呼叫(Intra-office call)的例子，软件需求的简要描述如下："用户 A 提起话机、听到拨号音，用户 A 拨打完用户 B 的号码、用户 B 的话机震铃、用户 A 听到震铃音。用户 B 举起电话，用户 A 和用户 B 正常通话。要求设计相关的软件以支持上面的局内呼叫过程。"由于篇幅限制，只能给出主要的模型和说明。

3.1 需求分析阶段

需求分析是本方法的第一步、需求分析的目标是面向问题领域的分析和面向待设计系统的需求分析。本阶段的要点是把需要分析的系统看成是与相应的目标环境交互的黑盒子，需求分析的主要工作是去获取和分析相关的应用领域和相关的用户需求，这个阶段将主要产生5个模型：①文本需求模型；②需求用例模型；③需求对象模型；④系统操作模型；⑤数据字典。

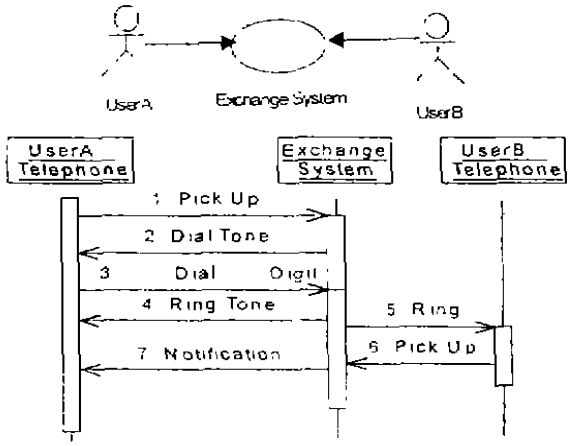

图2 需求用例模型中的用例图(上)和顺序图(下)

文本需求模型是通常的文本需求说明，它可以是从客户取得的用来进行系统开发的说明书。需求用例模型包括一系列基于 UML 表示的使用实例(use case)图和顺序图(sequence diagram)。本模型的目的是从用户的观点出发，获取和验证需求，从而确保系统可以正确地解决问题。我们根据需求描述建立起需求用例模型，如图2所示。需求对象模型包括一系列基于 UML 表示的类图。类图包含了多个对象和它们之间的关系(包括继承和聚合关系)。需求对象模型中的对象包括在系统边界的和系统外部看到的对象。比如说，系统的用户和系统与外部交互的接口对象。本模型具有下述作用：

(1)使用合成类图来描述系统，使用角色(Actor)来与系统交互

(2)记录了需求分析中发现的所有概念和它们之间的关系，以保证开发者和客户对于问题领域有一个统一的认识。根据需求描述建立的对应需求对象模型如图3所示。

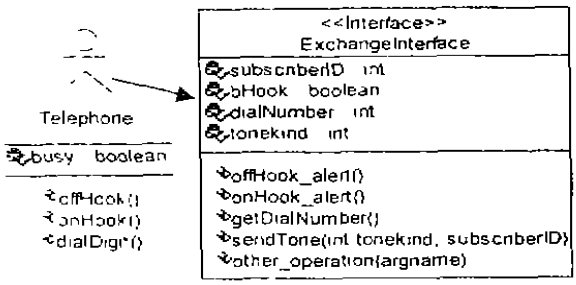

图3 需求对象模型

系统操作模型描述了系统需要完成的原子事务。系统操作模型可以被看成部分更细致化的用例模型，可以使用文本等来表示。需求分析期间，还需要创建一个数据字典。数据字典包括需求分析期间识别的一些概念列表和这些概念的简单解释。它在整个开发过程中被逐步使用和精化，比如包括需要运行的操作系统特性或者各个消息的交互时限等下能方便地使用 UML 语言表示的概念。

3.2 系统分析阶段

系统分析阶段的主要目标是分析确定系统的体系结构和标识系统中完成主要功能的最重要的对象。在系统分析阶段，使用面向对象的方法分析需要建造的系统。这个阶段使用的模型主要包括：

(1)对象分析模型，使用基于 UML 的类图描述系统的体系结构、表示系统中的对象。

(2)对象交互模型，使用基于 UML 的顺序图或合作图描述对象之间的交互。

而下适合使用上述模型来表示的决策分析和体系结构分析将记录在数据字典中。分析人员应当决定系统的初步体系结构和仔细考虑系统中必须标识的重要对象，这些对象可以来自需求分析阶段产生的需求对象模型，或者来自本阶段加入的对象。此时可以引入有关实时系统体系结构的设计模式(Design Pattern)技术，作出一些决定，比如决定采用何种调度算法，通信方法和安全性可靠性算法等。根据需求分析阶段的输入得到的系统对象分析模型如图4所示，其中包含了实时调度算法和内存分配算法，表明运行的操作系统调度特性和资源的管理特性，可以选择使用合适的设计模式来表示，其中不适合使用 UML 表示的部分将在数据字典中记录，以在设计阶段使用。对象交互模型表示了系统对象的动态交互特性，对应例子的对象交互模型如图5所示。

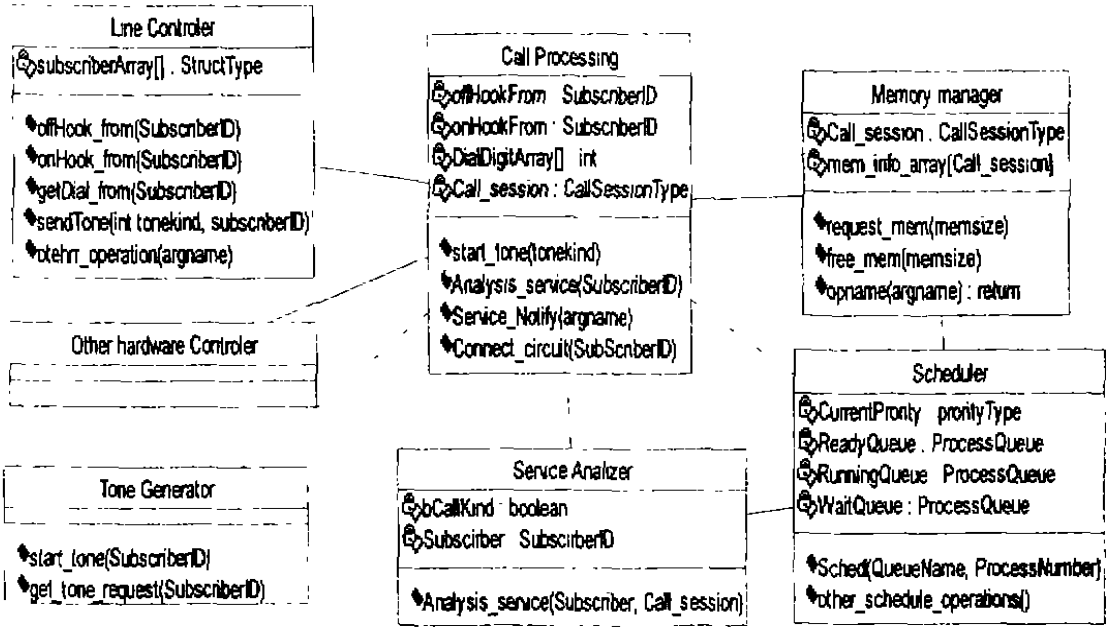

图4 对象分析模型

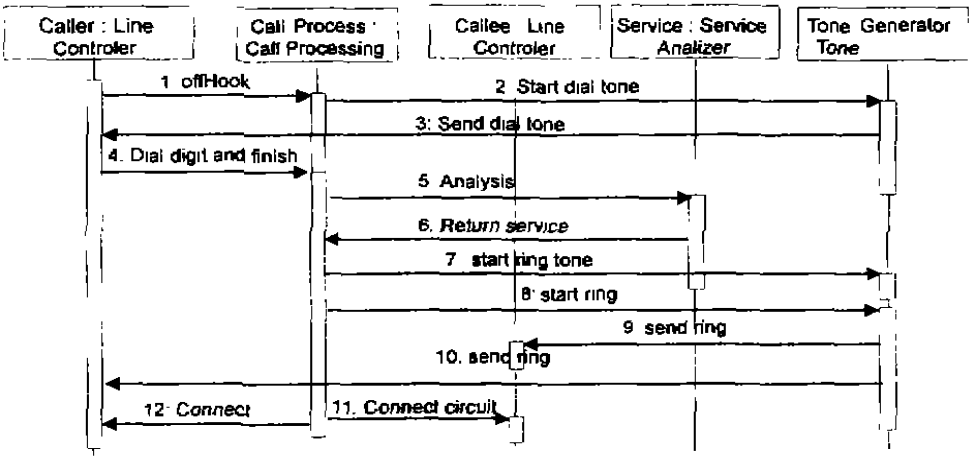

图5 对象交互模型

3.3 系统设计阶段

系统设计阶段的目标是定义系统的实现方案，同时制定系统的总体设计策略。这个阶段将把整个系统分解成若干部分，由不同的小组去设计实现。本阶段的要点是考虑设计的可重用性和未来的进化性。这个阶段使用的主要模型包括：

(1)体系结构模型的定义，包含：a)SDL 系统/块图，定义最终目标应用的体系结构；b)SDL 接口定义，定义系统部件之间的信号和远程过程调用接口。

(2)设计交互模型，使用 MSC[5]图定义系统中不同部件之间的动态交互过程。

对于不适合形式化描述的部件，比如用户界面定义和其它的非功能需求，将把它们记录在数据字典中。由于系统分析阶段得到的系统需求对象模型和需求交互模型使用 UML 来表示，而对应的系统体系结构模型和对应的设计交互模型使用 SDL 来表示，因此需要一系列的工具来支持从 UML 模型到 SDL 模型的转换，这是本方法中十分关键的一步。目前我们只能使用有限的工具来支持这两类模型的转换，很多工作需要开发者来决策完成，进一步的辅助工具正在开发之中。对应例子的系统体系结构模型如图6所示，包括了 Exchange 系统的系统/模块结构图。由于本例中使用

MSC 绘制的系统交互图和系统分析阶段的顺序图内容大致相同，在此不再给出。

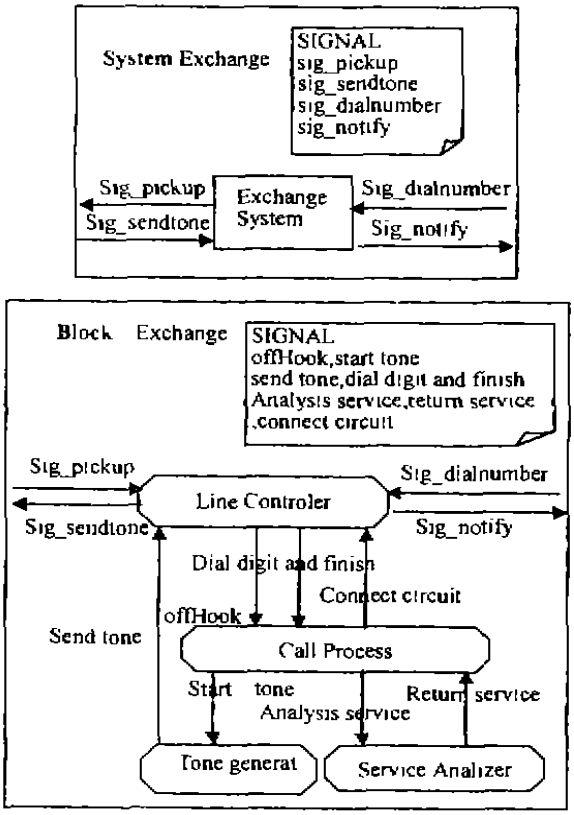

图6 系统(上)/块图(下)结构

3.4 对象设计阶段

对象设计阶段的目标是产生一个完备的、可以形式化验证的对系统行为的描述。对象设计阶段详细地定义了所有对象的行为和功能。这个阶段使用的模型全部使用 SDL 来描述，并使用 SDL 中的进程(Process)来描述系统设计阶段的主动对象的行为。系统的设计者在创建对象模型的时候特别需要考虑两个方面：(1)如何合适地在本阶段表示分析对象模型，比如需要考虑某个对象是主动对象或者是被动对象；(2)设计对象的动态行为，满足对象间的并发要求。

对象设计阶段晚期还包括测试/验证工作，其目的是验证系统功能的需求，比如说来自系统设计阶段的交互模型，可以在采用一定的 SDL 工具在本阶段进行验证。图6的块图中的 Call Process 主动对象使用图7来形式描述。(限于篇幅的关系，只能给出 Process 的大致流程)。在对象设计阶段可以充分使用 SDL 提供的数据类型、进程、过程和类似的高级语言的机制来详细完整地描述一个系统。其中的 SubscribeID 和 State-Type 可以使用 ASN 1[4]的形式来描述。比如 SubscribeID 可以描述成：

```
SubscribeID ::=SEQUENCE{
        SubscribeKind SubscribeKindType DEFAULT
        PB;
        MainAddress MainAddress Type;
        ……
        };
```

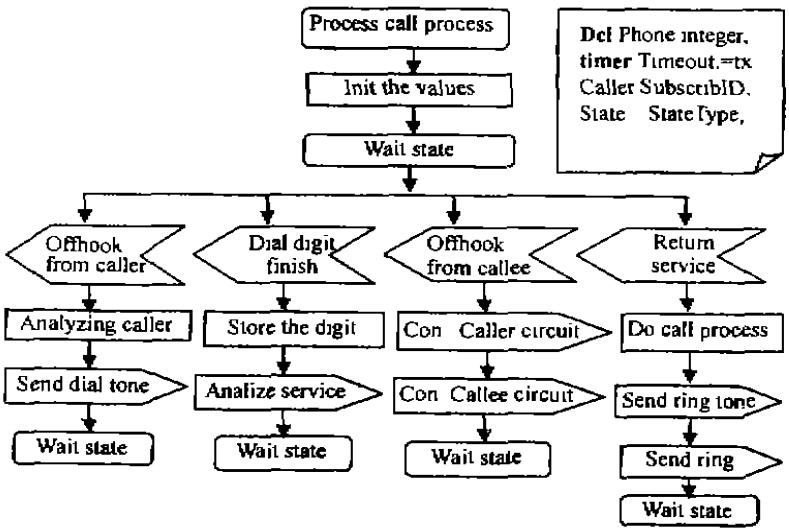

图7 Call Process 的进程图描述

3.5 实现

实现和测试的目标是产生最后的应用程序，比如，可在一定硬件上执行的软件。这个阶段的活动依赖于

(下转第117页)

为：

```
type signed8 is new Interfaces.C.signed char;
type signed16 is new Interfaces.C.signed short;
type unsigned32 is new Interfaces.C unsigned long;
subtype CRTOS.signed8 is signed8;
subtype CRTOS.signed16 is signed16;
subtype CRTOS.unsigned32 is unsigned32;
```

在文件 CRTOS-task.ads 中，程序包 CRTOS.task 的 tcreate 的说明信息为：

```
function tcreate (tid: CRTOS.signed16, depth: CRTOS signed16, priority: CRTOS.signed16, stack
    start: CRTOS.unsigned32, address: CRTOS unsigned32);
return CRTOS.signed8;
pragma Import(Stdcall, tcreate, "tcreate");
```

在文件 CRTOS-task.adb 中，程序包 CRTOS.task 的 tcreate 的函数体为：

```
function tcreate (tid: CRTOS.signed16, depth: CRTOS.signed16, priority: CRTOS.signed16, stack_
    start: CRTOS.unsigned32, address: CRTOS.unsigned32);
return CRTOS.signed8;
begin
   return tcreate (tid, depth, priorty, stack start, address);
end tcreate;
```

3.3 Ada 任务的创建过程

在 Ada 应用程序中，使用 task type 和 task body 语句定义一个任务，如在程序包说明文件(*.ads)中声明任务：

```
task type<task_identifier>[is task_definition];
   entry(task_entrypoint);
end[task_identifier];
```

在程序包函数体文件(*.adb)中描述任务执行的代码：

```
task body(task_identifier) is
    (declarative_part);
begin
    (handled_sequence_of_statements);
end[task_identifier];
```

当应用程序运行创建该任务时，首先调用 Initialize ATCB 函数，设置 Ada 任务控制块(Ada Task Control Block)等参数，接着调用 create_task 函数，将 Ada 任务的参数转换为 CRTOS 任务的相应参数，最后调用 CRTOS API tcreate 函数，创建一个 CRTOS 任务，由 CRTOS 内核对该任务进行调度管理。

结论 Ada 语言用于嵌入式系统的开发，需要 RTOS 提供实时运行支持。我们在 CRTOS 实时执行体上设计和实现的 Ada 语言绑定库，满足了整个系统的实时性和嵌入性等要求，其体系结构、设计特点和实现技术具有一定代表性和先进性，易于扩充，易于移植到其它 RTOS 上，对于提高国内军用及民用高端嵌入式应用的水平具有积极的推动作用。

参考文献

1 徐仑峰，熊光泽，刘锦德. 超微内核嵌入式实时操作系统的设计. 计算机科学，1998，25(3)
2 蔡建平，等. 实时嵌入式系统与 Ada 支持. 见：第八届抗恶劣环境计算机学术年会论文集
3 魏杰. Ada 语言在大型软件工程中的应用 见：第八届抗恶劣环境计算机学术年会论文集
4 IEEE Computer Society. IEEE 1003 5: Ada binding to POSIX. 1, July 1998
5 Baker T P, Giering E W III. The Gnu Ada Runtime Library (GNARL): Design and Implementation. Available at http://www.cs.fsu.edu
6 Giering E W, Mueller F, Baker T P. Features of the Gnu Ada Runtime Library. http://www.cs.fsu.edu
7 Teleogic Tau SCADE Available at http://www.csverilog com

(上接第122页)

应用程序的执行环境，比如需要考虑具体的对象概念如何映射为操作系统中的进程或线程等概念，对象间的并发机制在实际系统中的实际实现等。大体上说可按照以下步骤完成：

(1)使用自动代码产生工具来从 SDL 设计中产生代码；

(2)把产生的代码用于它的模拟执行环境，比如考虑外部的信号处理等；

(3)通过使用实时操作系统和交叉编译技术产生在相应硬件上可执行的代码；

(4)在目标环境里实现和执行测试用例(来自系统设计阶段的设计交互模型)。

我们将在前阶段产生的 SDL 模型上使用 SDL 支持工具，产生完整的目标代码，同时支持模拟执行、验证和测试，限于篇幅关系，目标代码和测试代码暂不给出。

总结 RCSM 方法主要面向实时通信软件系统的开发。目前 UML 模型和 SDL 模型之间的转换还需要人工的帮助，我们将逐步改进以支持从 UML 到 SDL 模型以及各个层次模型之间的智能转换工具。RCSM 的最终目标是以面向对象的方式完成实时系统开发中的从需求分析到实现和测试阶段所有模型的创建和转换工作。

参考文献

1 Rational Software Corporation. UMLV1.3 Specification, 1999
2 Lyons A UML for Real-Time Overview. Available at: http://www.Objec Time.com
3 Ellsberger I, Hogrefe D, Sarma A. Chapter 5. SDL formal object-ortented language for communicating systems. Prentice Hall, 1997
4 ITU Recommendation Z.105, SDL combined with ASN.1 (SDL/ASN.1), 1995
5 ITU Recommendation Z.120, Message Sequence Charts (MSC)