计算机科学2000Vol 27№ 10

面向对象的实时多任务系统设计方法*)

An Object-Oriented Software Design Method for Real-Time Multitask Systems

李 允 熊光泽 刘锦德 陈丽蓉
(电子科技大学计算机科学与工程学院 成都 610054)

Abstract In this paper, we present an object-oriented design method for real-time multitask systems using formal languages. We describe the whole development process of real-time multitask software from requirement analysis to system designing, targeting and testing. Finally, we illustrate the method by means of an example.

Keywords Real-time multitask systems, Design method, Object-orientation, UML, MSC, SDL

1 引言

随着实时多任务软件应用范围的迅速增长，实时系统本身也变得越来越复杂。同时，为提高软件质量，降低开发成本，缩短开发周期，系统在开发过程中所采用的方法就起着至关重要的作用。目前，已经提出了不少针对实时多任务系统的设计方法[1]。如RTSAD(Real-time Structured Analysis and Design, Hatley and Pribhai, 1988; Ward, 1986), DARTS(Design Approach for Real-time Systems, Gomma, 1984, 1986, 1989), JSD (Jackson System Development, Jackson, 1983), OOD (Object-oriented Design, Booch, 1986), CODARTS(Concurrent DARTS, Gomma, 1994)等。其中，RTSA 是满足实时系统的结构化分析设计方法，不适于处理任务结构方面的内容；DARTS 弥补了 RTSA 的不足，但在信息隐藏方面不如 OOD 方法；OOD 方法侧重于把系统划分为对象，实现信息隐藏，在处理任务结构方面仍显不足；CODARTS 集中了上述各种方法的长处，实现了信息隐藏技术，并适于处理任务结构。

在分析总结上述方法的基础上，本文提出了一种新的面向对象的实时系统设计方法，该方法从需求分析到详细设计等各个软件开发阶段都使用面向对象的形式化语言来实现，不但能进行信息隐藏，适合于处理多任务结构，还使实时系统的开发过程更加规范，并能借助工具软件，实现目标应用生成过程的自动化。设计过程中所使用的形式化语言包括：UML, MSC 和 SDL。UML (Unified Modeling Langnage) 是由 OMG 管理的面向对象的形式化语言规范，用来描述客观实体，以及系统与环境或是子系统之间的交互关系。MSC(Message Sequence Chart)是 ITU-TZ. 120标准，可用于描述实时系统与环境、系统各组件之间的通信行为，SDL (Specification Description Language) 是 ITU-T 定义的面向对象的形式化语言，适合于描述事件驱动的实时交互式应用。

2 实时多任务系统软件的开发过程

实时多任务系统通常包含如图1所示的软件开发行为。

需求分析用来对用户需求和与系统交互的环境进行建模，包括对象建模和动态建模两个过程，以描述系统的静态行为和动态行为。对象模型用 UML 类图描述，动态模型用 MSC 图描述。

体系结构设计的目的是定义系统的逻辑结构(软件体系结构)，用 SDL 的体系结构图和互联图来描述。

详细设计用来细化体系结构中的软件对象。需要确定两类对象：并行对象和被动对象。并行对象包含有能在系统中并行运行的控制进程，被动对象则作为并行对象所使用的资源而存在。

设计测试用例为测试工作设计出测试用例，主要用 MSC 描述。

目标化阶段用来生成最终应用，并使开发出的实时系统能够与目标环境和下层的实时操作系统协调运行。

测试用来根据测试设计所产生的测试用例，对系统进行全面测试，确保实现的系统能够满足用户需求。

实时系统可以采用图2所示的迭代开发过程。把应用划分为多个部分，对于每个部分都执行所有的开发行为。迭代开发过程使软件重用变得非常容易，后期

*)国家"九五"预研项目资助。李 允 博士生，主要研究方向：实时操作系统。熊光泽 教授，博士生导师，主要研究方向：实时计算机系统、实时软件可靠性评测。刘锦德 教授，博士生导师，主要研究方向 开放式系统、分布式实时系统。陈丽蓉 硕士，主要研究方向：实时系统设计方法。

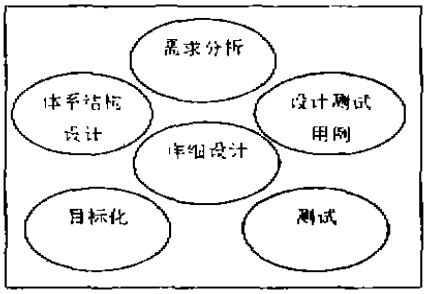


图1 实时系统软件开发行为

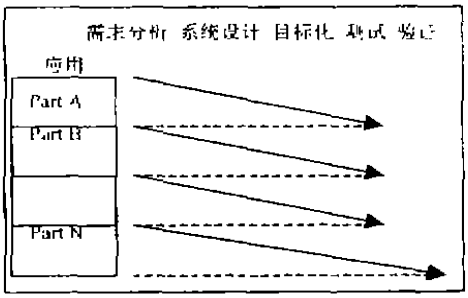


图2 实时系统软件开发过程

迭代可以重用前期迭代中的对象，并且支持快速原型。

3 需求分析

需求分析包含对象建模和动态建模两个过程。对象建模实现系统的静态模型，用来描述系统的静态关系，需要确定出与系统有关的实体，并以类图和实例图的方式表示出来。动态建模通过用户实例建模来进行，以描述系统的动态行为，需要确定出系统与环境、系统内部各组成部分之间的交互关系。

3.1 用 UML 进行对象建模

使用 UML 进行对象建模的步骤为：

(1)模块划分。把需求文档描述的系统划分为若干模块，每个模块对应系统视图的一部分，其中包含类，以及类之间的关系(如：关联、继承等)。

(2)确定类。在分析阶段，对象和类可以表示物理实体，也可以表示逻辑概念。通常，开始的时候可以通过分析需求文档中的名词来获取类。确定出的类应该反映出初始的系统结构，不考虑细节内容。

(3)确定类之间的关系。类之间的关系分为关联和聚合两种情况。关联可以表示物理设备之间的关系，或是逻辑依赖关系(如，雇佣关系等)，它表示的是两个对象之间的静态关系。关联可以通过分析需求文档中连接名词的动词来确定。对于表示拥有或是构成等关系的关联通常表示为聚合关系。

(4)确定类的属性。属性来源于需求文档中定义的数据，应该简单，不考虑设计和实现细节。对于象结构或是不定长度的数据，应该转换为类，并引入适当的关联或是聚合关系。

(5)确定操作。操作对应于期望从类中获取的服务，或是环境所需要的数据产生过程，可以通过分析需求文档中的动态视图来获取。

(6)引入继承关系，对对象模型进行细化和简化。对于具有通用结构成分的类，把通用成分抽取出来，封装成一个新的类，形成基类和派生类。

(7)创建实例图。实例图可以用来表示系统的特定内容，如：典型情况、系统启动和异常处理等，也可以用来描述相互关联的类之间的复杂关系。

对象建模是一个重复上述操作的过程，最终应该形成一个非二义性、非冗余性、容易理解、不考虑设计和实现细节，并尽可能满足用户需求的完整对象模型。

3.2 用 MSC 进行用户实例建模

用 MSC 进行用户实例建模，以捕捉系统的功能和动态需求。用户实例建模属于黑盒建模方法，系统被看作为一个整体，不考虑系统的内部结构。用户实例建模的过程为：

(1)确定系统方案体系结构。方案来源于需求文档中与系统动态行为有关的内容，把系统的动态行为划分为阶段，每个阶段对应于方案体系结构中的一个方案，为降低复杂性，可逐步细化方案，以使所有的末端方案都应比较容易通过交互关系来进行描述。

(2)用 MSC 描述方案，方案体系结构中的每一个末端方案用一个 MSC 来描述。为此，需要确定出相互通信的并行实例，对及实例间交换的消息。实例对应于对象模型中的类。消息通常对应于对象模型中类操作和操作的返回值，消息值对应于类的属性或是类操作的返回值。交互的形式如图3所示，

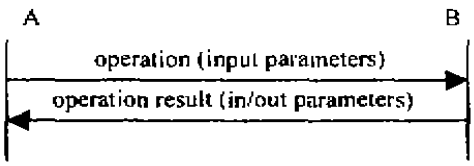


图3 系统实例之间的消息交换形式

4 系统设计

4.1 系统体系结构设计

体系结构设计通过需求分析阶段产生的对象模型和动态模型来定义应用的体系结构，可综合考虑自顶向下(细化过程)和自底向上(组合过程)的分析方法。细化用来把一个问题划分为比较容易解决的子问题，组合则是为了尽可能重用已存在的构件。该阶段的软

件元素为并行对象，还应确定出并行对象之间的通信关系。体系结构的设计过程为：

(1)描述系统体系结构。确定组成体系结构的高级对象，高级对象来源于对象模型中的并行类。在体系结构中，高级对象对应于 SDL 块，即把系统划分为若干块。划分主要是源于技术上的考虑，比如结构或是已存在构件的重用、硬件元素的封装等，不对对象模型进行分析。

(2)细化高级对象。细化用来把高级对象划分为越来越细的细节对象，可以利用对象模型中使用的关联和聚合关系来实现，如果确定出的并行对象能够很容易地转化为程序设计语言并能在真实平台上实现时，细化过程就算完成。细化过程中各对象之间的关系用 SDL 互联图描述，终结对象对应于 SDL 进程或 SDL 过程(过程由进程细化而来)。通过对高级对象的细化，把块细化为若干进程，并把进程细化为若干过程。

(3)描述对象通信关系。对象间的通信关系由 SDL 互联图来描述，其通信信道来源于并行类之间的关联关系，通信信号来源于 MSC 中的消息序列。

(4)确定被动对象。作为并行对象所使用的资源，被动对象包括：设备接口、数据管理和算法隐藏等模块。

4.2 详细设计

详细设计需要对系统行为进行全面描述，以实现期望的服务。并行对象和被动对象应该作进一步详细描述，使最终实现是可能的。详细设计由 SDL 和 UML 来描述，并行对象通过 SDL 进程图来描述、被动对象由 UML 类图来描述。

4.2.1 *并行对象的详细设计* 设计并行对象，首先要确定出进程的状态，然后描述出进程的行为。

(1)确定进程状态。确定出每个进程所有可能的状态。进程状态通过考察用户实例图来获取，其中，每个状态对应一组输入信号，进程根据历史状况对给定信号作出反应。

(2)描述进程行为。进程行为通过 SDL 提供的图形元素来描述，包括信号接收、信号输出、赋值、设置时钟、复位时钟、时钟到期、过程调用，或是进程激活、进程删除等。为描述进程行为，有可能引入新的常量、类型、变量和过程，还有可能引入新的被动对象以实现期望的服务。

4.2.2 *被动对象的详细设计* 设备接口模块用来隐藏输入输出设备的特征，可把设备需要的数据和操作 API 封装成一个设备类，实现应用与硬件环境的隔离。

数据管理模块可通过文件管理系统或是 OODBMS (Object-Oriented Data Base Management System)来实现。若采用文件来管理，需要确定存储的类、类的属性等内容。采用 OODBMS 管理对象数据则容易得多，可实现对象数据的直接操作。

算法隐藏模块用来隐藏那些可能随系统变化而发生改变的算法，提供一组操作来提供服务。

5 测试用例设计

集成测试源于对用户实例的细化，侧重于并行对象间的通信测试，单元测试则用来对被动对象和并行对象的内部结构进行测试，集成测试用例设计与体系结构的设计密切相关，体系结构设计的细化过程与集成测试的集成过程是一个互逆过程，它们可以并行操作，单元测试用例设计同详细设计密切相关，它们也可以并行操作。

测试用例主要用 MSC 描述，设计时应生成详细的 MSC 图，详细 MSC 通过细化需求分析阶段生成的 MSC 用户实例而来，细化的依据是 SDL 体系结构所描述的消息序列和对象的内部行为，因此，MSC 描述的实例间通信要与 SDL 体系结构相匹配，该对应关系为：

·垂直条为 SDL 体系结构所描述的实体实例；

·MSC 消息对应于 SDL 互联图中的信号；

·消息传送要为 SDL 模型中两个对话实体间的通信信道所支持。

测试用例还应该对体系结构方面的内容进行扩展，如：性能、强壮性和安全性等，用来提高最终系统的质量。

6 系统实现

在目标化阶段，开发人员把软件设计内容转换为可执行的代码，并综合考虑操作环境和 RTOS(Real-time Operating System)等内容。系统应用的建立过程为：

(1)实现并行对象。所有的 SDL 项目都应转化为可执行代码，把 SDL 信号转化为 RTOS 消息，把进程转化为 RTOS 任务。该过程可以借助工具软件来实现。

(2)实现被动对象。由软件开发人员根据应用情况进行设计。

(3)集成并行对象和被动对象。并行对象由外部事件、内部事件或定时事件激活，然后调用由被动对象所提供的服务。

(4)重用并集成已存在的子系统。

(5)调试应用系统，使用 RTOS 提供的调试器，对生成的应用进行交叉调试，发现并纠正程序中存在的错误。

7 测试

在 SDL 和 MSC 分析中设计的测试用例并没考虑实际的物理体系结构，属于抽象用例，为实施测试，应

把它们转化为具体用例，其转化过程为：

(1)由 MSC 产生具体用例 把 MSC 中系统与环境的交互转化为目标平台上的测试行为

(2)由 SDL 描述产生具体用例，首先分析 SDL 进程，细化交互序列框架，然后把进程中的交互行为转化为具体的输入和输出数据

产生实际的测试用例后，对系统进行测试，该测试完成后，还应该由系统用户进行接收测试，该测试不考虑系统体系结构，侧重于实际应用提供的功能，以及系统的质量和性能，接收测试主要依靠需求分析阶段建立的用户实例模型。

8 例子

在 ATM(Automatic Teller Machine)中，顾客输入密码和现金支取数额后，由 ATM 读取卡上的内容，并把相应信息传送到银行 如果信息与银行帐号一致，则传送信息到 ATM，由 ATM 输出现金，然后顾客取出卡和现金，下面给出了 ATM 的类图，MSC 方案体系结构，MSC 图，SDL 体系结构，互联图和进程图 根据进程图，即可生成系统的最终应用 生成最终应用可以通过 CASE 工具软件实现，也可以由软件开发人员根据进程图来编程实现

·ATM 类图(需求分析)，如图4所示。

·MSC 方案体系结构(需求分析)，如图5所示

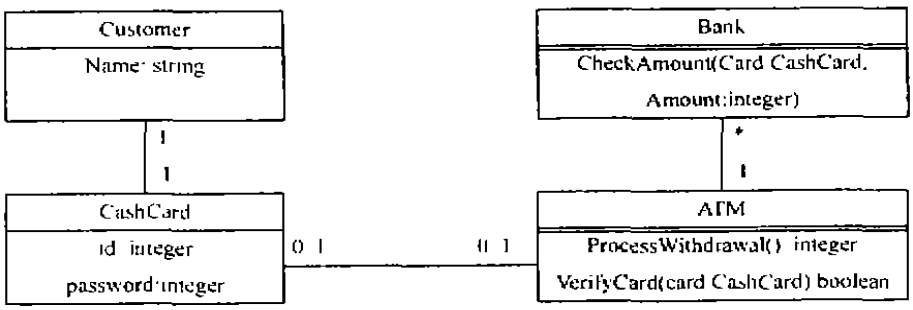


图4 ATM 类图

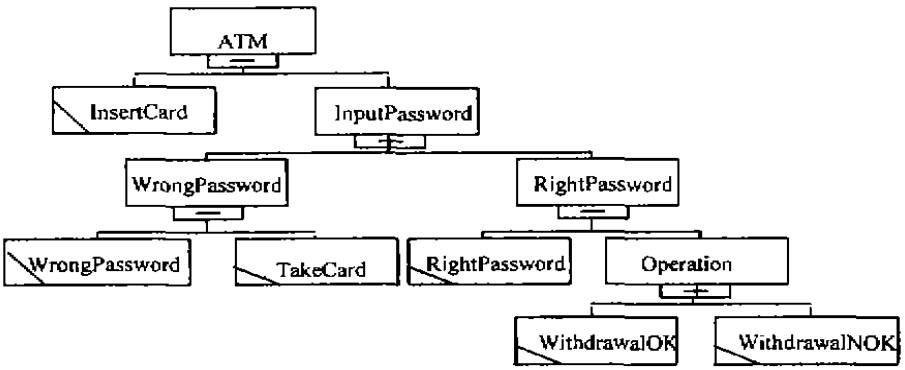


图5 ATM 的 MSC 方案体系结构

·方案 WithdrawalOK 的 MSC 图(需求分析)，如图6所示，图中，rs 表示 response(响应)，rq 表示 request(请求)，cf 表示 confirm(确认)。

·ATM 系统的 SDL 体系结构(体系结构设计)，如图7所示。ATM 系统被分为 BlockATM 和 BlockBank 两块，块 BlockBank 又被分为 Supervisor 和 Transaction 两个进程。Supervisor 用来处理 ATM 传过来的顾客信息，然后把信息转交给 Transaction，由 Transaction 对现金支取数额进行验证，并把结果送到 ATM 端。

·ATM 系统互联图和 BlockBank 块互联图(体系结构设计)，如图8所示。

·进程 Supervisor 和 Transaction 的进程图(详细设计)，如图9所示。

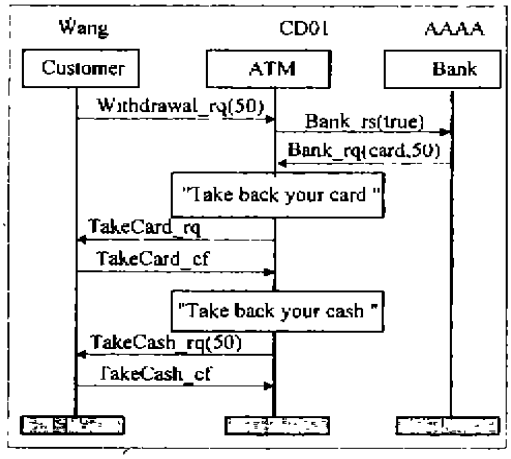


图6 方案 WithdrawalOK 的 MSC 图

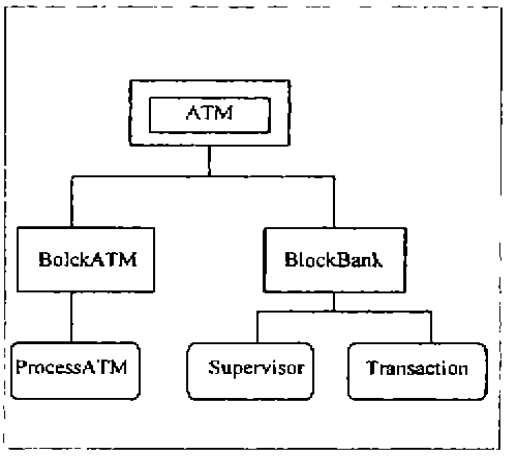
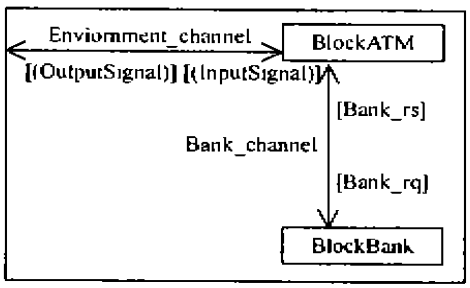


图7 ATM 系统的 SDL 体系结构



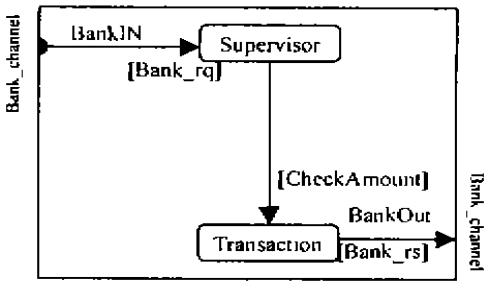


图8 ATM 系统互联图和 BlockBank 块互联图

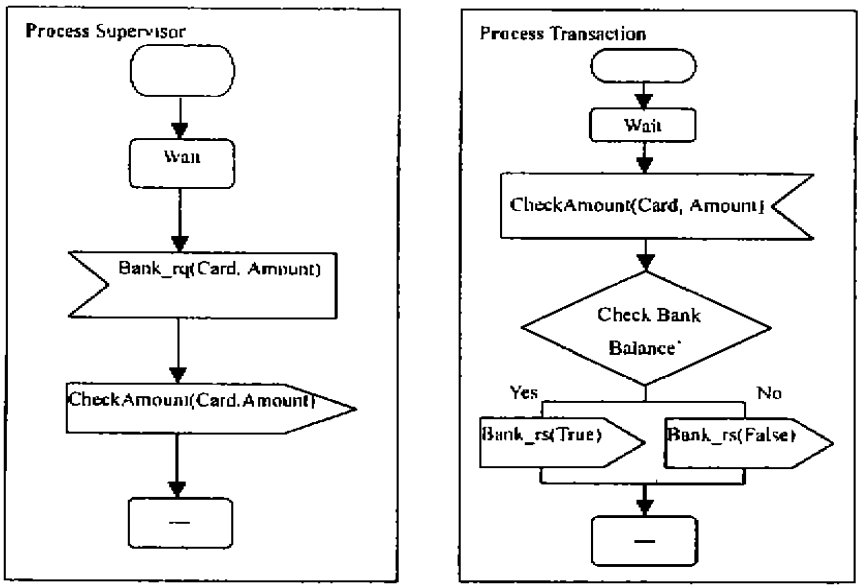


图9 进程 Supervisor 和 Transaction 的进程图

（下转第18页）

器误认，必须将 Script 和 Style 置于 CDATA 中。下面给出示例：

```
<script>
  <![CDATA[
  …unescaped script content…
  ]]>
</script>
```

使用正确的元素和名称空间 XHTML 标准中规定，文件的根元素标记必须是<html>，而且要用 xmlns 这个特别的属性来注明 XHTML 专属的名称空间，如：

<html xmlns=http://www.w3.org/TR/html1/">

严格声明 head 和 title 在 HTML 中<head>和<title>可以省略，但在 XHTML 中，一律得按规则加在适当的位置。另外，<title>必须是<head>中第一个出现的元素。

输出类型 经过验证的 XHTML 文件是一个真正的 XML 文件，因此理论上讲可以为任何兼容的 XML 浏览器或软件所理解。当然也可以输出成 HTML 文件，在 HTML 浏览器中显示。要输出成 HTML 类型，必须在 meta http-equiv="Content-Type"的类型设为"text/html"；同样，要输出成 XML 类型，必须在 meta http-equiv="Content-Type"的类型设为"text/xml"。

4 XHTML 及 Web 浏览语言展望

为了很好地发挥 XHTML 的优点，满足各种不同的需要，还有许多工作要做。目前，W3C 正在紧锣密鼓地进行 XHTML 相关标准的制定工作。有关的标准有：HTML 的模块化、子集与扩展性、文档配置文件(Document Profiles)。届时，随着 XHTML1.1 规范的推出，XHTML 文档的编辑者和浏览器能按实际需要去套用不同组合的 DTD。那么，今后 Web 浏览将全采用什么语言呢？在这里，我们不得不提到 XML。XML 是一种目前在业内广为流行的新兴标记语言，且大有成为 Web 浏览技术上的明日之星的趋势，但是不是说 XML 马上就可成为取代 HTML 的后继者呢？不是的，XML 的确有许多优势(详细论述参见参考文献中提到的文章)，但它仍存在一些短时间内无法克服的缺点。XML 还有很长的路要走，而 XHTML 做为对 HTML 的扩展，其优势使其成为 HTML 的理想继承人，现在已经有针对 XHTML 应用的支持，MathML 和 SVG 就是很好的例证。因此，我们预测在不久的将来，XHTML 将会成为浏览器中的核心页面语言，直至 XML 技术的真正成熟之日才会逐渐退出历史舞台。关于 XHTML 和 Web 浏览技术，我们将在今后的工作中进行相应的跟踪研究。

参 考 文 献

1 W3C 的第一个 XHTML 草案. http://www.w3.org/TR/1998/WD-html-in-xml-19981205
2 W3C 的第二个 XHTML 草案. http://www.w3.org/TR/1999/WD-html-in-xml-19990224
3 W3C 的第三个 XHTML 草案. http://www.w3.org/TR/1999/WD-html-in-xml-19990304
4 W3C 的第四个 XHTML 草案. http://www.w3.org/TR/1999/xhtml1-19990505
5 W3C 的第一个 XHTML 建议标准. http://www.w3.org/TR/1999/PR-xhtml1-19990824
6 Hu Lao. Two-Tigers Workshop. No-crap XML，1999.10.14
7 Bray T，et al. Extensible Markup Language (XML) 1.0 Specification. 1998. http://www.w3.org/TR/REC-xml
8 DOM (Core) Level 1 Specification Recommendation. http://www.w3c.org/TR/REC-DOM-Level-1
9 CSS1 (Cascading Style Sheet 1). http://www.w3c.org/TR/css1.html
10 CSS2(Cascading Style Sheet 2) http://www.w3c.org/TR/css2.html
11 Bray T Using XML to Build the Annotated XML Specification. http://www.xml.com/xml/xmlannotation.html.
12 Murata Madoto(Fuji Xerox Information Systems). DTD Transformation by Patterns and Contextual Conditions In:SGML/XML'97 Conference Proceedings. 1997.153～169
13 Jelliffe R. Using XSL as a Validation Languange. http://www.open-oasis.org/xml/xslAsValidator 19990124.html

(上接第46页)

参 考 文 献

1 Gomaa H. Software Design Methods for the Design of Large-Scale Real-Time System. The Journal of Systems and Softwre，1994，25：127～146
2 Selic B，Gullekson G，Ward P. Real-time Object-Oriented Modeling. John Wiley & Sons，Inc.，1994
3 邵维忠，梅宏. 统一建模语言 UML 述评. 计算机研究与发展，1999(4)
4 Nissanke N. Realtime Systems，Prentice Hall，1997
5 Gheorghe S，McGee J. Using Object-oriented modeling to design complex real-time embedded systems. Real-Time Engineering，1995，1(4)：11～25
6 Jacobson I. Formalizing use-case modeling. JOOP，1995，8(3)：10～14
7 Jacobson I，et al. Object-Oriented Software Engineering，A Use Case Driven Approach. ACM Press，Addison-Wesley，1992
8 Olsen A，et al. Systems Engineering Using SDL-92. North Holland，1994