实时系统开发方法研究*)

The Study of Development Method for Real-time System

乔 颖 王宏安 戴国忠

(中国科学院软件研究所智能工程实验室 北京100080)

Abstract In this paper, we investigate the development method for real-time system. Several methods employed in the phase of system modeling, implementation and testing are addressed. Moreover, we indicate the direction of the study of development method for real-time system.

Keywords Real-time system, System modeling, Development method

1 引言

实时系统不仅要求所产生的结果在逻辑上是正确的,而且要求在时间上也是正确的[1,2]。它已渗透到了社会生活的各个领域,如在过程控制、敏捷制造、核反应堆、航天航空和电讯业上都发挥了重要作用。对于实时系统来说,它应具备以下几个重要特性:

·及时性:实时系统所产生的结果在时间上有着严格的要求,只有符合时间约束的结果才是正确的。在实时系统中,每个任务都有一个截止期,任务必须在这个截止期之内完成,以此来保证系统所产生的结果在时间上的正确性。根据这一特性,实时系统可分为硬实时系统和软实时系统。对于硬实时系统来说,如果所产生的结果不符合时间约束,那么,由此带来的错误将是严重的和不可恢复的;而对于软实时系统来说,虽然结果的产生不符合时间约束,但由此带来的错误是可以被接受和恢复的。

·同时性:一般来说,一个实时系统常常有多个输入源,因此,这就要求系统具有并行处理的能力,以便能同时处理来自于不同源点的输入。

·可预测性:这是指,实时系统的行为必须在一定的限度内,而这个限度是可以从系统的定义而获得的。这意味着系统对来自于外部输入的反应必须全部是可预测的,尽管在最坏的条件下,系统也要严格遵守时间约束。因此,在出现过载的时候,系统必须要能以一种可预测的方式来降级其性能。

·可靠性:可靠性一方面是指系统的正确性,即,系统所产生的结果不仅在值上是正确的,而且在时间上也是正确的。另一方面是指系统的健壮性,也就是说,虽然系统出现了错误或外部环境与定义不符,但系统仍然可以处于可预测状态,它仍可以安全地带错运行和优雅地降级。

于是,如何开发出具有以上这些特性的实时系统便成为了一个十分重要的研究课题。我们可以将实时系统的开发过程分为如下几个阶段:即,系统建模、实现和测试。其中,实时系统的建模包括了需求分析和系统设计两个部分。而实时系统的开发方法则实际上是包含了以上各个阶段在内的一整套方法体系。在本文中,我们将围绕着这几个阶段来阐述目前实时系统开发方法的研究现状。本文剩余部分的组织结构为:第二部分介绍了在实时系统建模上所采用的方法;在第三部分中,我们将对实现阶段的工作进行分析;第四部分是对测试阶段工作的介绍;第五部分则对实时系统开发方法的研究进行了总结和展望。

2. 实时系统的建模

2.1 实时系统的需求定义

在需求定义中,主要是采用适当的方法对实时系统的需求进行分析,从而产生对系统功能和非功能行为的规格定义。这一阶段对于实时系统的开发来说,是非常重要的,它将直接影响到后面几个阶段的工作质量。对于实时系统来说,它的需求分为功能需求和非功能需求两大类。功能需求是指对系统所进行的操作以及这些操作所产生的影响的定义;而非功能需求则是指诸如系统的时间特性这样的需求。

结构化分析方法(SA)一直是分析者用来获取系统需求的最有效手段,它对于分析功能和数据需求来说是一个很经典的方法。D Hatley 与 I. Pirbhai 对传统的结构化分析方法进行了修改和扩展,提出了实时结构化分析方法(SA/RT)。通过控制流图的引入,以

*)本课题得到国家自然科学基金重大项目(项目编号:69896250)和国家自然科学基金重点项目(项目编号:79931000)的资助。

及将控制流图与数据流图进行适当的整合，实时SA(SA/RT)可以较好地处理实时系统的需求分析。

在结构化的需求分析方法出现之后，面向对象的需求分析方法又被广泛应用于对实时系统的需求分析上。同时还出现了一些用于实时系统需求分析的形式化方法。例如，著名的FOREST项目，在这里，Goldsack和Finkelstein定义了一个逻辑方案来处理实时系统的需求，并提出了一种需求抽取的方法。另外，在Esprit II ICARUS项目中，为建造和抽取实时系统的需求开发出了一种面向智能体的语言，称之为ALBERT语言。

实时系统的需求分析将形成对需求的规格定义。目前，对这个规格定义的描述基本上还使用自然语言，而现在比较流行的一些形式化方法，如，VDM和Z，它们使用了集合理论和谓词逻辑，比之非形式化和结构化的方法确实有了不小的进步，但以它们目前的状况，还不能很好地处理实时系统需求的规格定义问题。

2.2 结构化设计方法和面向对象的设计方法

结构化的方法和面向对象的设计方法一直是实时系统设计中的两大主流。在结构化的设计方法中，比较经典的有：Mascot，JSD，Youdon，MOON，HOOD，DARTs，MCSE等[3]。

JSD是一种基于进程模型的设计方法。虽然JSD最初并不是用于实时应用的，但它在大规模系统的设计中已取得了成功。目前，这种设计方法已经被用在了实时领域。Mascot3是专门为设计、构造和运行实时软件而开发的。Mascot3使用了数据流网络和分级设计，模块化是这一方法的关键。它具有用于设计、构造、实现和测试的多个可识别模块。与Mascot3和JSD有所不同，HRT-HOOD直接是用于硬实时系统设计的。它将设计过程看作是一个渐增特定规约的过程。这些规约定义了系统设计的特性。而HRT-HOOD定义了体系结构设计的两个行为，即，逻辑结构设计和物理结构设计。逻辑结构设计包含了与约束无关的规约，其目的是要满足功能需求。物理结构设计则要将这些功能需求和其它约束一起进行考虑，并且包括了非功能需求。它形成了判断细节设计及实现是否满足应用的非功能需求的基础。物理结构设计是对逻辑结构的提炼，并经常是一个迭代和并发的过程。在这个过程中，两部分的模块都会被开发或修改。分析技术应在物理结构设计中被尽早地使用，因此，通常可以定义初始的资源预算，而这个预算在逻辑结构的提炼中要服从修改和校订。通过这种方式，便可以从需求到实现都保持一个可行的设计。

面向对象的设计方法始于80年代后期。目前，它已被广泛应用在分布式实时系统的设计上。它与结构化的设计方法有着很大的区别。结构化的设计方法将系统看成是一系列功能的集合，而对数据的考虑则是次要的，同时，也没有处理并发性问题。面向对象的设计方法则将对象作为最基本的单位，它所处理的是类和实例、关系和角色、操作和事件、聚合及继承，并通过实体-关系图的表现形式增强了这种设计方法的可视化程度。而对于系统的行为，大多数面向对象的设计方法都是用状态图来表示的。一个状态图与一个类相关联，它的作用是对对象实体的行为进行描述。

在面向对象的设计方法中，比较典型的是实时统一建模语言(RT-UML)[4]。在RT-UML中，使用了类似于实体-关系图的语言来描述类及对象。对于系统行为的早期设计可以用消息序列图来表述。面对于系统行为的全面定义则可以采用状态图的形式来表示。

由于对实时系统来说，时效性比正确性更为重要，而面向对象的设计方法可以很自然地捕捉到这种具有硬实时特性的系统的特征和需求，因此，在对实时系统的设计上，面向对象的设计方法比结构化的方法更具优势。其优势主要表现在：

(1)面向对象的设计方法增强了模型视图的一致性。在结构化的设计方法中，分析视图和设计视图之间的映射是比较困难的，这就很难显示代码是否实现了分析模型。而在面向对象的设计方法中，各阶段的模型视图是一致的。在分析模型中所标识的对象和类在代码中有直接的表示。这样，就不必花费精力来显示问题定义和其解决方案之间的关系。一般来说，面向对象的系统可以使用两种方法来开发。一种方法是在分析模型上加入设计概念；另一种方法则是建造一个可以直接表达设计决策的翻译器。而无论采取哪种方法，都可以将分析模型直接映射到实现。

(2)面向对象的设计方法增强了问题域的抽象程度。结构化的设计方法在对系统的抽象和封装上的局限性比较大，它将数据与操作的内在关系割裂了开来。而面向对象的设计方法则可以支持数据与操作之间的关联。由于这种关联在现实世界中是普遍存在的，因此，这便使得面向对象的设计方法对系统的抽象更加直观和有力。用户可以更加清晰地对需求进行理解。

(3)面向对象的设计方法提高了稳定性。在结构化的设计方法中，需求的变化往往会对系统的结构产生很大的影响，从而浪费开发者的大量精力。然而，在面向对象的设计方法中，由于系统的抽象是建立在现实世界基础上的，因此它的稳定性是比较强的。需求的变化只是引起模型中某些部件的增加或删除，而不会导致系统的重构。

(4)面向对象的设计方法增加了模块的可重用性。在结构化的设计方法中，对模块的重用通常会导致代

码的修改，从而限制了系统的可重用性。在面向对象的设计方法中，可以通过继承及精练这两种技术来提高模块的可重用性。继承技术在无须修改源代码的情况下对已有的构件进行扩充，从而增强了模块的可重用性。这使得开发者仅需编写有关"不同事物"的代码。精练技术与继承技术十分类似，但是，它允许对象的不完全定义，然后，通过增加丢失的部分来对这些对象的不完全定义进行精练。这样，代码可以只被编写一次，而却可被应用到多种不同的环境。

(5)面向对象的设计方法增强了可扩展性。结构化设计所采用的建模方法使得它缺少可扩展性。然而，在面向对象的设计方法中，其较好的抽象和封装可以支持构件之间的松耦合，从而减少了"病态"耦合，提高了系统的可扩展性。

(6)面向对象的设计方法能够更好地支持对可靠性和安全性的考虑。在面向对象的设计中，由于采用了较好的抽象和封装，使得不同构件间的交互限制在一些被定义好了的接口上，这样，便可以使构件之间的交互得到控制，从而提高系统的可靠性。另外，面向对象的系统还提供了一些异常处理来保证在错误及异常情况下的系统的正确性。同时，面向对象系统较高的可重用性也减少了开发新系统所带来的错误。

(7)面向对象的设计方法可以支持并发性。并发性是实时系统的一个重要特性。结构化的设计方法中没有并发的概念，而在面向对象的设计方法中，却可以支持并发性。任务及任务同步可以通过状态图中的构件、活动对象和对象消息表示出来。

2.3 形式化设计方法

形式化的设计方法是继结构化设计和面向对象设计之后的又一流行的设计方法。它在实时系统设计中的应用也十分广泛。目前，用于实时系统设计的形式化方法主要可以分成四大类，即：时序逻辑、过程代数、自动机和状态机以及 Petri 网[5]。

时序逻辑是对命题和谓词的一个扩充，它引入了一些新的操作符来表示与实时系统相关的一些特性。Lamport 等人又对时序逻辑做了扩充，使得其本身的跃迁成为了可能。随后，经过更进一步的发展，时序逻辑允许将截止期附加到操作符上。例如，RTL(Real-Time Logic)就很好地将时间与一阶谓词逻辑结合了起来。

基于过程代数的形式化方法有：Pennsyevania 大学所开发的通讯共享资源(ACSR)代数。它是一个定义有界资源实时系统的过程代数。再如，过程代数 ACSR-VP，它对 ACSR 进行了扩展，使其具有了值传递。

在 I/O 自动机的基础上，Lynch 和 Vaandrager 提出了时间 I/O 自动机。它是为实时系统中的元素所建立的一个带标识的跃迁系统模型。在时间 I/O 自动机中，除了具有普通的输入行为、输出行为及内部行为外，还具有时间段行为，在此之后，Lynch 等人又对时间自动机模型做了扩充，使得它不仅能够描述离散行为，还能够描述连续行为。这一模型被称为混合 I/O 自动机，属于自动机类的形式化方法还有时间约束反应自动机(TRA)，它采用基本的空间和时间的概念来对表示进行限制，即，它只允许对反应、自发和偶然计算的定义。而通讯时间状态机(CTSM)则是描述实时系统的另一个形式化模型。在 CTSM 中，一个系统包括了若干通过信道彼此通讯的并发进程。每一个进程都拥有具有任意域的数据变量和具有离散域的时钟变量。在这里，时钟变量是特殊的变量，它专门用来表示诸如延迟和截止期这样的时间约束。CTSM 是可合成的，两个 CTSM 进程可以合成一个新的 CTSM 进程。

Petri 网可以对实时并发系统的行为进行建模。它是由结点间的双向图构成的。它对于并发和分布式系统的形式化分析来讲，是一个十分有效的方法。但是，Petri 网的缺点是它们会产生大量的表示，因此，又引入了有色 Petri 网来对这类系统进行更精确的建模。

一个利用形式化方法进行系统设计的典型例子是 Satoshi Yamane 提出的分布式实时系统的分级设计方法[6]。该方法利用了开放式的时间自动机。这一设计将系统中的进程分为高层和低层。首先，用开放式时间自动机来定义高层进程。然后，对进程进行修改以便满足接受性。下一步，是要定义低层进程，之后，再对这些低层进程进行修改，以便使它能满足接受性。在此之后，要使用基于假设-保证模式的时间模拟关系来校验高层与低层间的一致性。如果一致性不能满足，那么，将对低层进行修改。以上过程将不断被重复，直到设计出符合定义的系统。

2.4 实时系统的其它设计方法

除以上所述的设计方法外，在实时系统的设计中还有其它的一些方法，主要有：

·*硬软件协同设计* 该设计方法就是在对实时系统进行设计时，要将硬件和软件进行混合考虑，使得两者在整个设计过程中都是紧耦合的。这种设计方法所要做的工作包括：建立硬件/软件的体系结构模型，然后，将实时系统的定义映射到所建立起来的硬件/软件体系结构上。此外，该方法还要对基于给定体系结构的设计进行分析，而根据分析结果，设计者可以对已建立起来的体系结构进行修改，从而使得最终的体系结构符合系统定义的优化设计。这种设计方法在开销和性能间做了更大的妥协和折中。

·*基于构件的设计* 在该设计方法中，使用了构件思想对实时系统进行开发。基于构件的设计最大限度

地减少了开发一个应用所要编写的新代码的数量。许多软件模块可以从已存在的应用中被重新使用。系统整合可以通过一个可支持动态重配置的实时操作系统来自动实施，这样，就无需编写"粘合"代码了。基于构件的开发对于开发可重配置实时系统来说，是一个很有效的方法。

·基于中间件的设计　这种设计方法在操作系统平台和实时应用间提供了一些通用的中间服务。中间件具有精确定义的接口，在此之上，可以开发实时应用程序。这样，通过识别中间件服务集合，使应用模块可以被重用，从而降低了系统的复杂度，并降低了修改和扩展现存软件基或开发升级系统的开销。

·支持开放式实时系统的设计　开放式实时系统的特点就是它可以进行动态的硬/软件升级。为了支持这种动态的硬/软件升级，该设计方法提供了一个功能包。这个功能包允许实时系统即使在新的硬件或软件元素中存在错误的情况下，也能进行安全的硬软件动态升级。这个包所提供的功能包括：独立于应用的运行期服务、与应用领域相关的特定知识、支持依赖性分析和跟踪的方法及工具。这种设计方法使得实时系统的开发模式从"静态设计和广泛测试"转变成为"允许安全的升级"。

3. 实时系统的实现

在实时系统的实现阶段，需要根据设计对实时系统进行编码。因此，编程语言是这一阶段的主要研究内容。对实时系统编程语言的设计有六条基本原则，即，安全性、可靠性、灵活性、简单性、可移植性和效率。目前，用于实时系统开发的编程语言主要有三类，即、汇编语言、串行语言及高级并发语言。

最初，大部分实时系统都是用嵌入式计算机的汇编语言来编码的。但是，汇编语言的一个最主要的问题就是它是面向机器的，而不是面向问题的。于是，随着计算机性能的提高、编程语言的进一步成熟和编译技术的进步，用高级语言编写实时软件的优势逐渐显露了出来。为了克服如 Fortan 这类语言的一些缺陷，人们专门为嵌入式程序设计开发了一些新的语言。如，Jovial、Coral66和 RTL/2。近几年，C 和 C++语言又十分流行。然而，所有这些语言都是串行的，而且它们为实时控制和可靠性所提供的机制还都是比较弱的。因此，这类语言经常要依赖于操作系统的支持，并需要插入汇编代码，针对这一问题，又出现了一些实时高级并发语言，比较典型的有 Modula，Modula-2，PEARL，Mesa，CHILL 和 Ada，它们都包含了嵌入式系统开发的特征。此外，还有 Occam 语言，它与 Ada，CHILL 和 Mesa 相比，其规模比较小。该语言在开发紧耦合分布嵌入式应用中发挥着重要的作用。

此外、国外在实时编程语言方面的研究也比较活跃，如美国 Boston 大学开发的嵌入式实时计算的编程语言和工具——CLEOPATRA，在这个编程语言中嵌入了约束反应自动机(TRA)模型。该编程语言使得底层计算平台(硬件和操作系统)的限制对于编程者来说是可见的，并以此提高了可预测性。CLEOPATRA 是事件驱动的，因此，很适合于嵌入式过程控制应用。它是面向对象的和可构造的，支持模块化和可重用性。同时、CLEOPATRA 是语义健壮的、它的对象可以被无二义性地变形为 TRA 操作。

RealTimeTalk(RTT)语言是由瑞典 Malardalen 大学开发的。它在代码级是基于 Smalltalk 语法的。但在 RTT 中，不但修改了 Smalltalk 的非确定性特征从而保证了应用的瞬时行为、同时还提供了一个类型推理系统来进行动态类型检查。此外，为了对那些时间特性与定义相一致的任务进行合成、美国 Maryland 大学开发了一种实时编程语言，称为 TCEL(时间约束事件语言)。该语言具有高层的时间结构，从而允许编程者在一个程序内部嵌入实时需求。

4. 实时系统的测试

由于实时系统具有很高的可靠性要求、因此对实时系统的测试就必须非常严格。实时系统的测试不仅要针对系统的功能行为还要针对系统的时序行为。而且，特别重要的一点是，不仅要在正确环境下测试系统行为的正确性，同时，还要在非正常环境下测试系统行为是否可靠。而所有的错误恢复路径和错误同时出现时所产生的影响也都应该被测试。

传统的测试方法需要有一个称之为模拟器的测试环境。模拟器实际上是一个可以模仿实时系统活动的程序。它模拟了中断的产生，并实时地实施其它 I/O 活动。通过模拟器、可以产生正常和异常的系统行为。这样、尽管最终的系统已经完成了，某些错误状态也能通过模拟器而被安全地检测出来。

目前，在这一阶段的研究也有不少。Pennsyevania 大学开发了一个实时系统的运行期监测框架。在这个框架中，提出了监测系统的一个监测和检查(MaC)体系结构、从而与系统形式化需求定义进行比较来检查系统的正确性。为了能基于高层需求描述来检查系统运行的正确性，MaC 体系结构包含了若干层次，这些层次缩短了高层与细节层次的抽象差距。

在嵌入式系统的后期实现阶段，开发者面临着调整系统时间参数的工作。这时，他们就不得不求助于很耗时的反复调试方法。为了解决这一问题，美国 Maryland 大学开发了一个称之为 AFTER 的自动工具，它

可以检测系统时间参数的异常，利用度量数据自动分析和预测嵌入式实时系统中的时间参数，从而帮助开发者用一种系统的方式来调整嵌入式实时系统。

而在性能评估方面，美国 Michigan 大学开发了一套实时系统的性能评估工具。这套评估工具包括：一个合成负载发生器(SWG)；一个实时监控器(HMON)；一个集成的可靠性评估环境(DOCTOR)，该环境包括了一个软件容错插入工具。另外，这套工具还包括了一个为错误插入自动生成错误集的工具(TEMPEST)。

总结 实时系统的开发方法是一个包含了系统建模、实现和测试在内的一整套方法体系。目前，国内外在这一方面的研究都十分活跃。可靠性校验技术是其中的一个研究热点。这些技术包括基于形式化方法和调度原理的静态分析以及基于测试、运行期监控和检查的动态分析。此外，软件工程在实时系统开发中的应用也是一个很重要的研究课题，并且，针对实时系统的开发，软件工程面临着一些新的挑战。如，对时间、可靠性等约束的考虑等。而从实时系统的发展方向来看，开放式实时系统将是下一代实时系统的主宰。因此，实时系统的开发目标将逐步集中在对开放式实时系统的开发上。例如，支持动态硬/软件升级和可重配置系统的一些开发工作，便是在这一方面所做的尝试。同时，对于开放式实时系统开发来说，如何在系统的灵活性和可预测性上保持最大的折中，也将成为一个重要的研究内容。

参 考 文 献

1 Stankovic J A. Real-time and Embedded Systems. ACM Computing Surveys, 1996, 28(1)

2 Stankovic J A, et al. Strategic Directions in Real-Time and Embedded Systems. ACM Computing Surveys, 1996, 28(4)

3 Burns A, Wellings A. Real-time Systems and Programming Languages Addison Wesley Longman, England, UK, 1997

4 Douglass B P. Real-time UML, Developing Efficient Objects for Embedded Systems. Addison-Wesley Longman, Massachusetts, 1998

5 Joseph M. Real-time Systems, Specification, Verification and Analysis. Prentice Hall, UK, 1996

6 Yamane S. Hierarchical Design Method for Real-time Distributed Systems. In: 5th Intl. Conf. on Real-time Computing Systems and Applications, 1998

(上接第75页)

义 stereotypes 去扩充 UML 的 Stereotypes。如此反复地求精，直到能使用 UML 类模型描述类及其协调关系为止。因此，RM 是能够在概念层描述 UML 对象模型的一种抽象模型。UML 的对象模型是 RM 的细化。如图12所示，我们提出的建模方案是：在用 UML 建立对象模型之前，先建立角色模型，并逐步从角色模型求精到 UML 对象模型。

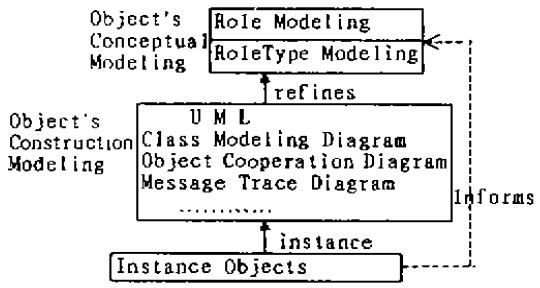

图12 使用角色模型的建模方案

小结和今后的工作 在本文中，我们讨论了对象模型化的四个视点，并强调了首先应注重对象角色或作用的观点，因为角色模型从概念上抽象地描述了对象的存在理由和协调的行为关联。本文还给出了角色模型的基本图示法。此外，我们还分析了软件模式的语义及其 RM 描述，明确了软件模式的描述由知识级和操作级描述两个部分组成，论述了角色模型是在知识级描述模式的一种有效方法。本文给出了用角色模型在知识级分别描述设计模式和分析模式实例，以及从软件模式的 RM 到 UML 类模型和消息追踪图的求精和实现过程。作为今后的研究课题，RM 的元模型(Meta-Meta RM, Meta-RM, RM, User's RM)体系及其形式化，角色模型的约束机制，基于 RM 扩充 UML 及其 Stereotypes 等都有待于进行进一步研究。

参 考 文 献

1 Gamma E. et al Design Patterns. Elemens of Reusable Object-Oriented Software. Addison-Wesley Publishers, 1995

2 Fowler M. Analysis Patterns, Reusable Object Models. Addison-Wesley Publishers, 1997

3 Buschmann F. et al. Pattern-Oriented Software Architecture. A System of Patterns. Addison-Wesley Publishers, 1996

4 OMG. Unified Modeling Language V. 1. 4 Specification (draft) OMG, 1999

5 何克清，木村高久，岡本泰次，田中茂. RM Visual Specification, Version 0. 70. Fujitsu Limited, Japan, 1999. 12

6 何克清，岡本泰次，田中茂. Extended UML's Meta-Model with RM, Version 0. 50. Fujitsu Limited, Japan, 2000. 03

7 Andersen E P. Conceptual Modeling of Objects. A Role Modeling Approach: [Dr Scient thesis 4]. Department of Informatics, University of Oslo. 1997

8 Reenskaug T. UML Collaboration and OOram semantics New version of a green paper. Nov. 1999

9 Odell J. Power Type. JOOP, May 1998