嵌入式软件建模、实现与验证:研究与进展*)

胡 军 张 岩 于笑丰 王林章 李宣东 郑国梁

(南京大学计算机软件新技术国家重点实验室 计算机科学与技术系 南京 210093)

摘 要 随着计算机硬件设备计算能力的迅速提高,嵌入式系统中软件的规模和复杂度的急剧增大,软件可靠性在嵌入式系统中的重要性占据了统治地位。本文首先概要介绍了嵌入式软件不同于传统商业软件、科学计算软件的物理性、实时性、领域性等重要特征,以及由此带来的困难和挑战。然后重点介绍目前在解决嵌入式软件系统开发过程中的问题时所采取的建模思想、实现技术和验证方法。最后对嵌入式软件及其相关技术的发展进行了展望。

关键词 嵌入式软件,嵌入式系统,嵌入式建模,实时软件

The Research and Development on Modeling, Implementation, and Verification for Embedded Software

HU Jun ZHANG Yan YU Xiao-Feng WANG Lin-Zhang LI Xuan-Dong ZHENG Guo-Liang

(State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract Embedded systems are the devices that include a programmable computer but are not itself a general-purpose computer. The reliability of modern embedded system mainly depends on the embedded softwares which are becoming more and more complicated. This paper gives an overview of the problems and challenges in the development of embedded system and software design. Firstly, embedded computing software is quite different from the mainstream business computing software and scientific computing software in many issues; such as: physicality, real-time requirement, domain-specific, reliability, etc. Then we introduce recent research effort on those areas, for example: model-based design, object-oriented programming, component-based development, formal verification technology, etc. Finally, a conclusion is drawn and a prospect is given on the cost-efficient development of reliable embedded software.

Keywords Embedded software, Embedded system, Embedded modeling, Real-time software

1 引言

嵌入式系统是指将可编程计算设施(处理器、存储器、输入输出设备等)安装在其他物理设备系统中,而这些设备并不是专门为商业计算、科学计算或者个人计算所使用的系统。IEEE(国际电气和电子工程师协会)对嵌入式系统的定义是:用于控制、监视或辅助某个设备、机器和工厂运作的装置。系统设计目的在于满足某种特定的计算需求。从20世纪70年代开始,嵌入式计算系统就已经在控制工程领域开始应用。与逻辑控制电路相比,可编程处理器加上合适的控制程序可以更好地满足控制需求。早期嵌入式控制软件的任务简单明确,由控制工程师自行设计开发。在软件的设计和实现的过程中,关注的重点是如何有效地满足控制软件的非功能性需求,基本不考虑软件工程中控制和管理系统规模和复杂性的方法和技术。

近年来,随着硬件设备计算能力的迅速提高以及嵌入式系统软件复杂度的大大增加,软件可靠性对嵌入式系统的影响占据了统治地位。例如:在航空航天、电站控制以及铁路运行管理等系统中的软件代码规模都达到了几十万行、甚至上百万行。嵌入式系统开发的成功与否极大地依赖于嵌入式软件的质量。与此同时,网络和通信的快速发展使嵌入式应用迅速地由工业控制层面扩展到消费电子产品领域,给嵌入式软件系统开发的各个方面(从系统分析、设计到实现、验证)都带来了新的困难和挑战。与商业软件相比,现代嵌入式软件不仅具有相类似的复杂功能性需求,同时各种非功能性需求构成了其最重要的特征。因此,嵌入式软件系统的开发必须要紧密结合现代软件工程中的方法和技术,如:分层抽象控制机制、构件化技术、基于模型的设计、分析与验证等等。

本文第2节简要阐述嵌入式软件的主要特征以及分类;第3节分别介绍和分析近年来嵌入式软件设计建模、实现和验证相关的方法和技术;最后对嵌入式软件未来的发展趋势作了若干分析和展望。

2 嵌入式软件的特征和分类

2.1 特征

嵌入式软件区别于非嵌入式软件主要是在于前者对非功能性需求的重视程度,下面给出嵌入式软件几个重要的特征。

物理性 嵌入式软件的计算过程参与到周围物理环境的行为过程中,它本身成为物理世界活动过程的一部分。系统其他部件通过某种形式的接口与嵌入式软件相互通信联系,只认为其是整个系统中具有某些物理属性(反应速度、能量消耗、容积大小等等)的一个处理部件。这是嵌入式软件与非嵌入式软件最大的不同所在。正是这种物理性的特点衍生出后面几种重要特征。在非嵌入式软件中,计算的逻辑正确性是占首要地位,而软件在某种特定计算平台上运行时所表现出的物理特性处于次要考虑的地位。

*)本文的研究工作受到国家自然科学基金(批准号60203009,60233020),江苏省自然科学基金(批准号BK2003408)和国家973项目(批准号2002CB312001)的资助。胡 军 博士生,主要研究方向为软件工程,嵌入式软件建模,形式化方法。张 岩 博士生,主要研究方向,网构软件建模与形式化验证。于笑丰 博士生,主要研究方向软件工程,模型驱动转换与验证。王林章 博士生,主要研究方向软件工程,模型驱动的软件测试。李宣东 教授,博士生导师,主要研究软件工程、形式化方法、模型检验。郑国梁 教授,博士生导师,主要研究软件工程、形式化方法。

实时性　在嵌入式软件中，某一种计算过程常常要求在可以预测的时间内完成。严格实时性的特征要求嵌入式软件的开发从需求分析、设计到实现、验证都必须显式地考虑实时时间约束的影响。与实时性紧密关联的是系统可靠性和安全性，尤其是在具有高安全性需求的应用领域（如航空航天、武器控制、医疗卫生等）。在以往的实时系统、反应式系统研究中人们已经积累了许多实时性处理的方法和技术，但是对于现代嵌入式软件开发技术来讲，如何有效地满足实时性需求仍然是一个很大的挑战。

资源受限性　嵌入式计算设备的空间体积、重量、能量消耗以及成本都有着严格限制，使得嵌入式软件的设计和开发被限定在有限范围内选择所使用的处理器、存储容量以及程序规模和能量消耗等，同时还要满足正常的功能性需求以及实时性要求。资源的严格受限性需要更好的资源调度管理方法，目前的许多资源调度算法并不一定适合嵌入式软件系统。

领域性　嵌入式系统通常是为了满足某些领域中特定的计算需求。系统的不同结构、行为特征常常使用不同领域的知识来建模，而不同应用领域的计算模型不尽相同。因此，嵌入式软件先天上就具有特定的领域性（domain specific），这给大规模复杂系统的集成带来很大的困难。

分布性　网络和通信技术的快速发展使得现代的嵌入式软件系统越来越多的采用分布式技术，分布式架构具有良好的健壮性、并行性、资源共享性等优点。在航空、汽车等工业控制领域的嵌入式系统常常需要提供时钟同步、实时触发、可靠的消息传递、高容错性的分布式实时架构；无线通信的嵌入式传感器网络则更需要实时处理、能源优化、安全稳定；各种与 Internet 相连的信息家电的出现代表着一个更为开放的分布式网络。

2.2　分类

嵌入式软件表现形式纷繁复杂，难以给一个比较清楚的类别划分。但从所提供信息的不同层次来看，嵌入式软件系统大致可以向用户提供两类信息：一类是物理层次的信息，如：机械信号、光信号、化学信号、声音信号等等；另一类是非物理层次的符号信息，即人们在生活中进行交流时所使用的信息，如：语言、文字、图形等。由此可以将大多数嵌入式软件划分为提供这两种不同信息的类别。第一类系统主要是在控制工程领域中，它们处理的是不需要直接给人们使用的物理形式的信息，如：家用电器的控制设备，工厂车间、交通运输等使用的自动控制设施等等。人们要理解这些系统人机接口所提供的信息，还需要一定形式的转换条件，如：必须是受过相关训练的工程师和技术人员才能准确地理解系统提供的信息的含义。因此，虽然在个人计算机出现以前，日常生活中早已存在各种以单片机形式出现的嵌入式设备，但常常不会引起太多的注意。第二类嵌入式系统则是近几年以消费电子设备以及信息家电的形式出现提供可以直接理解和处理的符号信息。其主要目的不是进行过程控制，而是为了满足在大量异构环境下信息及时获取与处理的需求，应用层面延伸到生活各个角落。符号信息的处理需要强大的软件支持，需要更多的计算资源，硬件的高速发展提供了这种可能和背景。此类软件的市场需求潜力巨大，更新换代周期短。所需的开发技术和工具同第一类软件有较大的不同；后者更多地、也更易于采用现代软件开发技术以及各种工具、方法，同时，其硬件/软件平台、开发环境可以做到相当的标准化，以降低开发成本，提高效率。

3　嵌入式软件的建模、实现及验证

近年来，嵌入式软件已经成学术界和工业界共同关注的热点论题。如：美国计算机学会的嵌入式系统特别兴趣研究组（ACM-SIGBED）从 2002 开始出版 ACM 嵌入式计算系统学报（Transactions in Embedded Computing System），每一期都讨论嵌入式计算系统中的一个专门议题。国际嵌入式软件会议（EMSOFT）从 2001 年开始由美国 DARPA（Defense Advanced Research Projects Agency）信息技术处和 NSF（National Science Foundation）联合主办，主要议题是现代嵌入式软件系统设计和实现的相关理论、方法和技术。之后 2003 第三届开始，由 ACM-SIGBED 承办。其他还有：嵌入式系统编译器、架构和合成的国际会议（CASES）。由欧洲设计和自动化学会（EDAA）主办的嵌入式系统软件和编译器国际会议（SCOPES），等等。

目前，针对嵌入式软件的研究工作主要存在两种不同的切入点。一种侧重于工程实用，由于现在嵌入式软件的开发方法、工具落后于主流商业软件的开发技术，这部分研究工作注重于如何将后者应用到嵌入式领域，并且形成相应的工程技术和工具。另外一种则认为随着嵌入式软件规模和复杂性的不断增大，嵌入式软件的建模方法、实现技术以及软件可靠性和安全性验证等方面都遇到了新的困难和挑战，需要有新的思想和方法。但人们都有一个共识，即嵌入式软件系统的设计与开发必须要充分利用现代软件工程的思想和技术，如：模型驱动，架构设计，构件式开发等。本节的内容主要是对目前嵌入式软件建模方法、实现技术以及形式化验证技术三个方面的研究工作进行相关的介绍和分析。

3.1　建模

软件模型在整个系统开发工程中占据着重要地位。使用模型可以提高开发者对整个系统的观察深度和控制复杂度的能力，给不同的开发阶段提供全局统一的视图和指导，提高软件质量、生产率和可靠性。建模也是进行形式化分析和验证的基础。目前，对于嵌入式软件系统的建模，存在几种不同的建模思想和方法。

首先，在嵌入式控制工程领域，Simulink[1]被广泛使用来对动态、非线性实时系统进行建模设计。Simulink 是 Mathworks 公司开发的非常成熟的商业建模工具，具有图形化的交互式开发环境。其目的是建立严格的数学模型，对所设计的物理系统动态行为进行仿真，并在此基础上对设计的软件系统行为进行建模和分析，支持自动生成软件代码、测试方案等。但是 Simulink 是根据控制工程中数据流转换的规律来建模、仿真，与计算机科学中使用的建模语言和实现语言相比较，还不具备精确形式化定义的语义、强类型系统等特征，因此，在系统模型和实现之间还需要进行相应的一些转换工作。如：Paul Caspi 等人对 Simulink 模型中的类型和时间描述进行形式化的定义，设计了一种自底向上的分层转换方法，将离散时间的 Simulink 模型转换为由 Lustre 同步语言实现的系统[2]。此外，Simulink 模型的性质验证是通过仿真过程来完成的，一些应用于计算机程序设计语言的形式化程序分析和模型检验相关的技术和工具，无法有效地应用于针对 Simulink 模型的性质验证。

第二类方法是使用与商业软件类似的建模方式，使用标准建模语言 UML[3]（或者类 UML 的建模语言）对系统不同视图分别建立模型：结构模型、行为模型、状态模型等，然后进

行相关的模型分析和转换。如:在 Manfred Broy 等设计的一个基于模型的嵌入式软件系统工程化开发方法中[4],其使用的类 UML 的形式化模型包括:Data type declaration, Component box diagrams, System structure diagrams, Extended event trace, State transition diagrams 等等,并支持增量式建模过程和外接形式化验证工具。Andrzej Wasowski 等[5]构造了一个基于形式化状态图模型到资源受限微控制器系统实现的软件合成工具。法国的 Hugues Martin 等直接使用 UML 用例图、类图、消息图和状态图对 JavaCard Applets 的业务需求和安全需求分别建模[6],并利用自动测试工具 TGV 从 UML 的模型自动生成测试用的 Java 程序,等。但是,UML 本身其实并没有包含对时间、资源等性质的描述机制,因此必须对其进行相应的扩展,相关的工作有:embedded UML[7], UML Profile for Schedulability, Performance and Time[8], real-time UML[9]等。

第三类建模思想是基于混成自动机(Hybrid Automata)。嵌入式系统中既有离散状态变化,又有连续时间行为,用混成自动机来建模是非常自然的想法。相关的研究工作有:R. Alur 等设计了一种形式化层次混合建模语言 CHARON[10],其中使用 Agent 来表达系统结构层次及其并行性;用混成自动机(Mode)来建模嵌入式系统的行为语义层次。从形式上看,Mode 类似于 UML 中的状态图。CHARON 的重要特征在于组合式语义(Compositional Semantics),是其系统进行精化设计的基础。在 Tunc Simsek 等提出的一种分布式分层控制的设计方法中[11],使用混成自动机作为中间过渡层的模型,将连续世界和离散世界沟通起来。其目的是组织那些在空间上分布的嵌入式设备共同完成一些不同的系统级复杂任务;核心思想就是任务分解,通过分层控制、相互协作完成任务。最底层是物理设备,它们之间通过信号交互;高层是抽象物体,通过符号的消息传递进行交互。因此,混成自动机构造的准确程度如何,直接影响到高层控制的能力和范围。Cadence Berkeley Laboratories 的 Jerry R. Burch 等人对混成嵌入式系统建立了一个基于完全形式化的模型[12]。他们使用抽象代数初步定义了一种框架,在这个框架范围内,可以形式化地定义各种计算模型,如:metric time, non-metric time, Pre-post Time 等,各个模型之间的交互,以及进行模型相互比较。这种完全形式化的框架和模型可以为进一步设计嵌入式软件集成化的开发环境提供理论基础。

第四类建模方法是基于嵌入式软件的领域性和构件式开发技术。也是目前关注的重点,其基本思想是:将嵌入式系统中所涉及到的各种计算方式分别建模,整个系统的结构和行为由属于不同计算模型的构件及其交互来表达,以控制系统开发的复杂度和提高可靠性。如:具有代表性的是美国加州大学的 Ptolemy[13]和 Kansas University 设计的 Rosetta[14]。在 Ptolemy 中,Edward A. Lee 等根据人们在设计建模过程中对并发和时间的不同处理方式划分出在嵌入式计算中常用的若干模型:Differential Equations、Discrete Event、Finite State Machines、Synchronous/Reactive model、Cycle driven model、Timed-CSP、Timed-PN 等。这些计算模型用接口自动机(interface automata)[15]形式化地表达出来,称为 Type Signature Domains。同时提供了一个具有多态机制的基于构件的系统级类型系统,不仅可以检查构件接口之间的数据是否兼容,还可以检查构件接口之间的动态交互方式是否兼容。这样,当多个构件集成时,通过接口自动机的组合运算可以有效地检查构件接口之间的静态数据和动态交互。Rosetta 则是让参与嵌入式系统设计的各个领域专家尽量使用自己熟悉的语义来建模和设计自己的构件;然后通过 Rosetta 提供的交互机制:组合运算和投影运算来对不同领域的语义模型之间的交互进行建模。在 Rosetta 中所划分的不同层次的计算模型与 Ptolemy 中略有不同,其已经定义的几类领域模型的基本集包括:Discrete time domain、Infinite-state domain、Finite-state domain、Time domain、Frequency domain 等以及在各个基本集之间的交互组合与投影的关系。每一个模型都是基于一个特定的领域,称之为 facet。模型和模型之间进行分层定义;可以由不同的模型来混合定义另一个模型。不同的领域模型其实反映了系统不同的侧面视角。与 Rosetta 相似的还有 Carnegie Mellon University 设计的 Time Weaver[16]。

其它类似的采用构件式建模方法的支撑工具和开发环境还有:PECOS(Pervasive Component System)[17]是由德国、瑞士和荷兰等国家的企业和研究所合作的一个设计构件式嵌入式软件系统的项目计划。主要关注的是在控制工程领域的应用。PECOS 设计的建模语言 COCO 可以有效地描述系统的功能和非功能性质,并且支持从 COCO 到实现级语言 Java、C++等的代码生成。Vanderbilt University 设计的 MIC(Model Integrated Computing)中[18],没有使用统一形式的建模语言,而是在维持原有各个领域建模语言的基础上,引入元语言建模机制来解决多领域计算模型构件之间的交互问题。通过定义抽象语法、具体语法、解释层的三元组 $L=\langle A, C, I\rangle$ 形式化地建立了类似于 UML 的四层元语言建模架构;其元建模语言采用的则是 UML 类图和 OCL 语言。Virginia University 设计的 VEST[19]则侧重于系统软硬件构件之间非功能性性质的依赖关系的检查:如:相互调用图、接口及其参数类型的需求、构件的前驱需求、独占资源的需求、版本的兼容等等。Gregor Gössler 等[20]设计的 Metropolics 将计算、通信和交互分别使用进程、媒体和调度者等实体来建模,从而支持在不同的抽象层次上的分解和组合。整个系统框架包括一个编译器的前端,将元模型的规约说明转化成中间表达形式,和一系列的合成、精化、分析和验证的工具集。另外,工业界的工程人员们提出的 Model Compiler[20]和 DESERT(DEsign Space ExploRation Tool)[22]工具包也是同样支持构件式模型设计和验证。其基本思想来源于汽车装配流水线,构件模型库的粒度较 Rosetta 和 Ptolemy 要小,主要是在工业界已经成熟使用的各种控制子例程。

3.2 实现

从系统模型到具体的实现是一个从抽象到现实多层次的转换过程,下面主要从嵌入式软件实现语言,系统架构及其相关技术方面做一些简要的介绍和讨论。

在系统实现语言方面,目前许多嵌入式控制软件仍然优先选择标准 C、Ada,或者是同步编程语言:Lustre、Esterel、Signal 等。与主流的面向对象程序设计语言相比较而言,它们更能有效地控制和使用各种系统资源、确定任务的运行时限、描述系统的并行行为。C 语言是一类非常优秀的通用程序设计语言[23],它把高级语言的最佳成分与汇编语言的控制和灵活性结合了起来。在嵌入式软件系统实现中得到极为广泛的应用。Ada 是美国国防部官方认可的为实时软件系统开发而设计的语言和相应的支持环境[24],现在已经发展到 Ada 95,在航空航天等安全关键控制系统中有着广泛的使用。尽管在军事系统以外的领域中没有得到推广应用,由于美国国

防部是世界上最大的实时系统软件的定购商，因此 Ada 语言就成为在军事相关领域中嵌入式实时软件系统事实上的工业标准。Lustre[25] 和 Signal[26] 等都是一种基于数据流的同步程序设计语言，将整个系统看作实时交互的节点集，实际上构成了一种时间演算系统，可以较好刻画系统的动态行为，在许多安全关键系统中已得到成功的应用。但是，另一方面，使用此类语言使得系统实现工作与硬件紧密相关。当系统复杂度提高时，其可靠性、开发效率，维护成本等难以得到有效的控制和保证。其软件可移植性较差。

近几年由于智能手持设备、信息家电、高性能微处理器智能卡等市场的迅速兴起，Java 技术在嵌入式领域的应用成为热点，得到工业界的广泛支持和推广，并且提供了相应的规范实现和应用开发支持环境。在有限存储、显示和处理能力的嵌入式应用领域，主流的 Java 技术是 J2ME(Java 2 MicroEdition)规范[27]和 JavaCard 规范[28](如图 1 和图 2 所示)。

• J2ME/CLDC(Connected, Limited Device Configuration)：针对具备间断网络通讯能力的个人移动信息设备，虚拟机使用完全重新设计的 KVM。由不同的 Profile(如：MIDP，PDAP 等)提供针对设备特殊功能的 API 和扩展类库，并且允许规范的实现可以以此为最小实现集合进行扩展。

• J2ME/CDC(Connected Device Configuration)：针对有固定的不间断网络连接的共享连接信息设备，如：机顶盒、网络电视等具有 2M 内存和 32 位处理器的嵌入式设备。虚拟机仍然使用的是 JVM。

• JavaCard：针对具有 8/16/32 位处理器，最小配置为 1k 的 RAM，8k 的 EEPROM 和 16k 的 ROM 的智能卡应用。除了少部分的 Java 核心类库，其它类库都是专为智能卡而设计，字节码格式也是特别设计的。但所有的对象都是静态分配，不支持碎片回收。运行支持环境分为四个层次：操作系统及基本函数层；JavaCard 虚拟机层；JavaCard 框架层；厂商自定义类层；Javacard Apple 应用程序层。最大程度地支持了一卡多用。

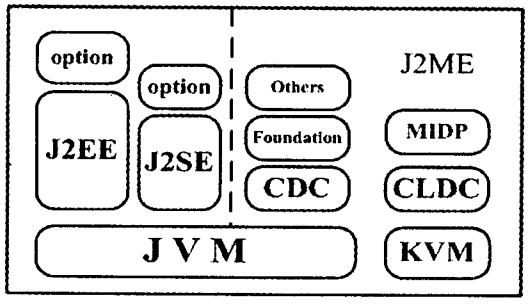

图 1　J2EE、J2SE 和 J2ME 的比较

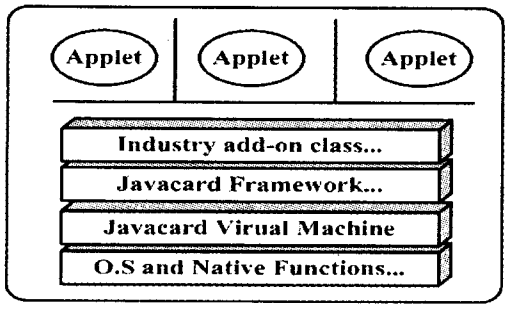

图 2　JavaCard 的系统架构

事实上，Java 已不单单是一种面向对象的程序设计语言了，已经成为一种重要的现代软件开发技术和软件运行支持环境。在传统的实时控制工程领域，人们也开始考虑使用 Java 技术，扩展 Java 对运行性能、实时要求、资源受限等的支持。如：Sun 公司的 Real-Time Java Specification[29] 是 J2SE 语言的超集，扩展了为实时系统所设计的一些机制，如：同步和共享资源管理、异步事件处理和控制转移、物理内存存取等。它除了保留传统 Java 中的堆分配、碎片回收内存管理模型之外，还支持永久内存(immortal memory)和区域内存管理(scoped memory)两种内存模型；但是这样也使得 RTJS 的实现工作相当复杂。MIT 的 William S. Beebee 等在 RTJS 规范的实现中发现[30]，为了满足实时任务的严格实时性要求，最关键的一点就是做到保证使用非堆内存的实时线程与碎片回收管理线程在运行时候任何阶段都不能交互影响。Angelo Corsaro 等也完成了另一个 RTJS 的实现[31]：jRate(Java Real-Time Extension)，并且对运行时刻区域内存中的对象引用检查提出了一个更为有效的算法，通过引用关系的父子树将引用安全检查转化为子类型匹配检查，使得算法的复杂度从线性降为常数级。但是到目前为止，RTJS 在嵌入式实时系统领域还没有得到广泛的应用。

与此同时，也出现了一些更为简化的 Java 支持平台，希望在大量的低端嵌入式设备中(如：16 位或者 8 位的微控制器，只有 512bytes RAM 和 4kB ROM 等)应用 Java 技术。相关工作如：simple Real-time Java[32] 是澳大利亚的 RTJ Computing Pty. Ltd. 开发的一个实时 Java 的支持环境，其主要目的是在小内存低端嵌入式设备上使用 Java。但是它与 RTJS 所要求的规范语义有所不同，simpleRTJ 实际上是一个小型的 Java 操作系统，完全提供内存分配、堆管理、线程管理、软件定时器等功能，同时还有相应的开发工具链支持。JEPE(Java Execution Platform for Embedded Systems)[33] 是 Mjalner Informatics 设计的在低端嵌入式设备上运行的可剪裁 Java 字节码支持环境。其支持的 JEPE 是功能有限的 Java 语言，对标准 Java 语言规范的语义也进行了调整，以支持内存资源非常有限的运行环境，如：数据类型中整型调整为 16 位，长整型为 32 位，暂时不支持多线程和某些标准类库等。Nik Shaylor 等[34] 设计了一个符合 CLDC 语义的虚拟机实现架构 Squawk，准备用于下一代的 Javacard 系统。Squawk 设计的字节码结构和标准结构不一样，有利于减小内存的需要和验证，并且所有的核心解释器和碎片回收器都是用 Java 实现的，可以自动转化为 C 代码，编译后直接在硬件平台上运行。Pieter H. Hartel 等[35] 则对 Javacard 规范和主流的 Java 规范(J2SE 和 J2EE)之间的不同之处从程序设计和模型设计两个角度进行了详细的比较和分析。

Java 在嵌入式系统中的应用也使得碎片回收技术引起人们的非常关注，从软件工程角度来看，面向对象语言和碎片自动回收技术提供了相当好的软件可重用性、存储系统使用可靠性，从高级语言这个层次上保证了软件系统工作的安全性、可靠性；但是在嵌入式实时环境中，如何构造合适的碎片回收算法仍然是一个很大的挑战[36,37]。

除了这些工业界使用较为通用的语言外，其他还有一些由大学和研究所设计的试验性的系统实现语言。如：Berkeley 的 Thomas A. Henzinger 等为硬实时嵌入式控制软件系统设计了一种平台无关的程序设计语言 Giotto[38]，将软件功能以及实时性要求的实现与具体硬件平台计算能力、调度管

理划分开来的构想，其严格形式化的语法和语义有利于编译器从软件到不同硬件平台的映射的自动化构造。Euclid[39]是University of Toronto设计的程序设计语言，同其他的实时语言类似，具备强类型检查、能够声明内存地址变量、扩展的自定义异常管理机制等特点，并且可以合理准确地预测程序的最差执行时间，对实时任务的可调度性做出准确的分析。当然，获得这些能力的同时也排除了其他的一些在通用程序设计语言所具有的强有力的表达机制，如：动态数据结构、递归框架结构等。另外，值得注意的是基于λ-演算的函数式程序设计语言在嵌入式实时系统中的应用。Yale University的Walid Taha等[40]认为：函数严格的Totality和Uniqueness数学性质正好准确地刻画了实时软件的特征，除去一些系统运行性能以外，目前实时软件的所有计算特性都可以用函数准确地定义。但是目前主流的函数式程序设计语言，如：Scheme、SML、Ocaml、Haskell等还不是纯Totality函数，使其应用于嵌入式实时软件还有待进一步研究。

在系统架构方面，由于目前系统中需要处理的硬、软件资源越来越多，大部分嵌入式软件系统都采用了基于某种嵌入式操作系统的应用开发和实现模式来处理。底层由嵌入式操作系统或者虚拟机运行环境来提供包括任务管理、进程通信、中断管理、同步与互斥、内存管理等在内的传统操作系统所具有的功能以及嵌入式系统所特别需求的实时性要求。这种分层抽象控制的实现机制简化了上层嵌入式应用开发的复杂度，也使得主流的面向对象高级编程语言(Java，C++等)得以广泛的使用，大大提高了开发效率和软件的可移植性。典型的嵌入式操作系统有：PSOS，QNX，VxWorks，uCOS，Embedded Linux等等。

对于分布式、构件化的嵌入式实时系统，要求有一种能够为上层应用提供时钟同步、实时触发、可靠的消息传递、高容错性的分布式实时架构。这种架构往往是以总线及总线协议的形式出现；如：SAFEbus[41]是Honeywell公司为载人航空航天器设计的分布式、高容错性、实时总线架构；SPIDER[42]是美国航空航天局(NASA)的Langley研究中心设计的分布式星形实时总线；FLEXRay[43]是BMW，Daimler Chrysler，Motorola和Philips为现代汽车动力传动系统和底盘控制系统设计的总线架构；Technical University of Vienna的Hermann Kopetz等人设计的TTA(Time-Triggered Architecture)[44]是既可用于航空领域，又可用于汽车领域的高可靠性的实时总线架构。David E. Culler等人设计的微型嵌入式系统[45]则代表了新一代嵌入式传感器网络迅速发展的趋势：节点完全分布式、无线通信和自组织网络拓扑的构架；而且要满足系统的能量优化控制、实时信息处理和通信、高可靠性和健壮性等等需求。

在嵌入式系统中能量的管理和优化非常重要。在大多数系统里，最消耗能量的是计算子系统，包括处理器和内存系统。可以在不同层次上对系统能量进行管理和优化，相关的工作有：使用低能耗的数字逻辑设备；采用可调电压的新型微处理器架构；在操作系统、运行环境层使用能量优化调度算法；在编译器层采用相关能量优化技术等等。从软件系统的实现来看，编译器层的优化对高层嵌入式应用屏蔽了底层硬件的细节，提供了良好的工具支撑。但是一些普遍使用的编译优化技术在嵌入式系统资源受限的环境下并不一定能够达到原有的效果。如：循环展开，函数内嵌等优化技术会明显增加运行代码的长度和程序的能量消耗，这将要求提高整个系统的运算、存储能力和能量供应，这在一些嵌入式应用中也许是不能接受的。编译器前端的词法分析、语法分析等技术已经是非常成熟，所以目前这方面的大部分研究工作都是关注于编译过程后端针对某个特定平台的能量优化。如：L. N. Chakrapani等[46]对各种编译优化技术在StrongARM类的嵌入式处理器平台下与能量消耗的关系进行了量化评价，并且对基于能量优化的编译技术进行了概略的分类。STMicroelectronics所开发的FlexCC[47]是一个模块化的可重定目标的编译器框架，其目的是：在设计新的ASIP(Application Specified Integrated Circuit)的同时，通过重用FlexCC提供的框架来设计相应的高效编译器。后端优化专门针对ASIP处理器的架构特性，如：寄存器的分配、软件流水线、硬件执行循环等。Northwest University的Joseph等[48]详细分析了在手持式嵌入设备的内存子系统中基于循环展开、函数内嵌的编译优化技术在程序运行性能与内存能量消耗之间的关系，认为：指令的存取数量与代码长度的乘积可以用来较好地衡量内存的能量消耗。并给出了两种在指定的内存能量限制条件下确定合适的优化策略的算法。

此外，现代嵌入式处理器芯片大多采用多级流水线、高速缓存等技术来大幅度提高运算速度，这使得要准确地预测某个计算任务的最差执行时间(Worst Case Execution Time)非常困难。而WCET是安全关键控制系统中极其重要的参数。在这里出现了两种完全不同的处理思想：一种认为精确的WCET值是不可能获得的[49]，并且现在的硬件资源越来越宽松，为保证软件的可靠性，在实现时应该将高速缓存禁用，并限制使用中断。虽然这样会造成一定的资源浪费，但不影响大局。另一种思想则认为可以通过设计有效的模型和工具来准确测量WCET[50]；实验数据表明后者是可行的，但是这种模型和工具本身相当复杂，因为它必须要能够准确分析特定的硬件平台。

3.3 验证

基于模型的开发技术优点之一就是可以利用形式化方法进行软件正确性和可靠性验证。形式化验证不同与传统的嵌入式系统的调试和测试技术。它采用基于数学的语言，技术，工具来对复杂的计算机系统进行相关性质的验证。虽然形式化方法不能完全确保系统的可靠性，但可以通过揭示系统的不一致性、歧义性和不完备性来增加对系统的理解程度，从而提高对系统可靠性的可信度。对于嵌入式软件的功能性和非功能性的特殊需求，人们认为形式化验证方法会有更好的应用前景。常见的形式化验证方法包括：推理验证、模型检验和静态程序分析。

推理验证是构建一个形式化的公理系统，包括推理规则；使用严格的推导证明来验证软件的正确性。证明推导过程的一部分可以做到自动化，但在多数情况下还是需要人工的参与。因此，虽然推理验证有着非常成功的嵌入式软件系统验证的例子，由于其人力成本高昂，目前还不适合于大范围的推广应用。如：巴黎市无人驾驶地铁14号线的控制软件的设计和验证中使用了交互式形式化推理验证，系统总共87000行Ada代码，效果非常好，但是软件开发率也高达600人/年[51]。

模型检验是以有限自动机为形式化模型，通过穷尽系统状态空间进行相关性质的检验。目前，模型检验在工业界的软件验证应用中还未到实用阶段，但近几年已经嵌入式系统的软硬件协同设计中[52]得到广泛的重视。模型检验的工作

主要在于设计合适的算法来解决搜索状态空间过大的问题，相关的工作有：Matthew B. Dwyer 等[53]对基于事件驱动的构件式分布式实时嵌入式系统的设计定义了一类系统状态空间，其具有 quasi-cyclic 结构的模型，将全局的状态搜索分解为许多可以并行执行的不相关的子搜索，但仍然还是保持了原来状态系统的相关性质，从而减小了进行模型检验时状态空间的大小，但是这同时也带来了计算时间复杂度的提高。Kim Guldstrand Larsen 等[54]通过构造实时任务的时间自动机模型，对嵌入式系统所应用的特定调度策略的正确性和资源需求进行形式化的验证，相应的支撑工具集是 Uppaal[55]。此外，模型检验还常常作为测试方法的辅助检验工具，来查找使用非形式化验证手段发现不了的错误。

静态程序分析是基于抽象解释的形式化方法，可以看作是编程语言标准语义的抽象。目前使用的通用自动化分析是建立在近似的基础上，可能会有些错误无法检测到；要做到精确的验证，还需要人工进一步的参与。相关的工作如：Yamine Aït Ameur 等[56]对实时系统的 Lustre 程序进行抽象解释，并使用 Java 构造了一个工具分析器对程序的错误进行形式化的分析；其结果已经成功地应用于欧洲 Airbus 的飞行控制系统的设计中。阿里安 5 型火箭飞行控制软件的一个浮点数转换溢出导致导航设备失效的检测也是使用静态程序分析方法的成功范例[51]。

在智能卡应用领域，一方面由于开放平台(Open Platform)标准的形成和 Javacard 智能卡的广泛使用，要求卡平台的提供商和卡应用程序的提供商都需要对其产品进行极高的安全评价等级的验证，另一方面智能卡上应用软件的规模相比较商业软件要小得多，使得非常适合基于数学的形式化规约和验证方法的应用，这已经引起人们的极大的兴趣。如：德国的 Bernhard Beckert 等在 KeY Project[57]中定义了一个形式化描述 Javacard 程序性质的动态逻辑演算系统，并设计了一个相应的交互式证明器。这种动态逻辑实际上是从 Hoare Logic 扩展而来并加上了模态词的谓词逻辑，可以用来描述类似于 Javacard 等较简单的面向对象编程语言的一些性质，对于较复杂的语言特性，如：动态绑定、异常处理等尚有困难。法国的 Pierre Bieber 等[58]开发的 PACAP 原型工具可以根据给定的 Javacard 的配置条件，在一个新的应用 Javacard Applet 从网络上动态下载到智能卡中之前，使用基于模型检验的 SMV model checker 静态的检查 Javacard Applets 之间的数据流是否满足智能卡安全策略的要求。Australia National University 的 Rajeev Goré 等[59]则使用了多模态的命题逻辑来描述智能卡中的安全策略模型和 Javacard Applet 的安全属性，并且设计了一个推理验证器 CardKt。与 PACAP 不同的是，CardKt 可以即时地从网络上下载到智能卡中运行，对相关的 Applets 验证完毕后，即可从卡中删除。荷兰的 Joachim van den Berg 等人对 Javacard API 的实现进行形式化的验证[60]，他们首先使用 JML(Java Modeling Language)对非形式化的 Javacard API 规范进行了精确的形式化描述，然后利用转换工具 LOOP[61]将 Javacard API 的实现转换成可以描述其在推理验证工具 PVS 中的语义的结构，最后使用 PVS 来证明这些结构能不能满足 JML 的规约要求。其最终目的是想要得到一个完全经过形式化验证的 Javacard API，从而为验证上层 Javacard Applet 的性质奠定一个牢固的基础。目前已经对 Application Identifier Class 和 JCSystem Class 进行了形式化的规约和验证。

从目前研究现状来看，形式化软件验证技术在工业界尚未得到广泛应用，传统非形式化验证技术，如：测试、模拟等的重要地位目前仍然不可替代。然而毫无疑问的是：在未来的嵌入式软件系统的开发中，形式化软件规约和验证技术将逐渐成为最重要的方法之一。

结束语 现代嵌入式计算系统具有许多与传统商业软件、科学计算软件不同的重要特征，在计算机科学领域中带来许多新的问题和挑战。本文对近年来嵌入式软件领域中设计建模方法、实现技术以及形式化验证方法等方面的研究现状和进展进行了较为详细的介绍和分析，在此基础上我们对进一步的发展做出如下的展望。

首先，基于模型和构件式开发方法将在嵌入式软件的设计、实现和验证中被广泛采用。结合构件式开发技术一方面有利于构件重用，降低开发成本，提高软件生产率，另一方面，基于模型和构件的技术可以有效地降低嵌入式系统的复杂性，比如采用组合式形式化语义可以降低验证的复杂性。由于嵌入式软件的领域性，比较可行的方法是面向不同领域建立标准模型，并且开发相应的嵌入式构件库。

其次，在嵌入式计算环境中，底层平台(包括硬件和系统软件)控制了上层应用的运行速度、反应时间，并影响了软件并发任务粒度的划分；在分布式计算环境下，平台对软件的影响更为显著，如：系统容错问题，通信延迟问题都是无法通过类似于虚拟机的机制来完全屏蔽的。如同在建筑工程领域中必须要考虑所需材料的质量、尺寸、弹性等属性一样，在嵌入式软件的工程开发中，必须要综合考虑实现软件功能的物理载体的属性，如：处理器运行速度、存储容量大小、网络通信效率、电源的能量供应等等。采用类似于服务质量(QoS)的模型机制将有助于这方面的进一步研究。

第三，形式化方法在嵌入式软件系统建模与验证方面将扮演越来越重要的角色。形式化建模、分析与验证可以有效地支持嵌入式系统资源的动态分配，做到充分利用现代高性能处理器的架构性能，同时仍然保证系统运行参数的可预测性。进一步提高形式化方法在嵌入式软件领域应用的适应性和实用性是进一步研究工作的重要内容。

最后，目前嵌入式软件方面的研究大部分关注于如何解决在设计和开发的某一阶段中出现的关于实时性、资源受限等方面的问题，缺乏针对建立一个完整的、符合软件工程规范的开发过程相关的研究工作，尤其是针对嵌入式软件生命周期中维护阶段的分析和研究工作非常缺乏。在消费电子领域，产品更新换代的短周期和跨平台 Java 技术的使用使得软件的维护工作大大简化，但是对于工业领域中的大型嵌入式软件系统，如何安全可靠地维护和升级仍然是非常困难、同时又是迫切需要解决的问题。

参 考 文 献

1 http://www.mathworks.com/products/simulink

2 Caspi P, Curic A, Maignan A, et al. From simulink to SCADE/lustre to TTA: a layered approach for distributed embedded applications. In: Proc. of the 2003 Conf. on Languages, Compilers, and Tools for Embedded Systems (LCTES'03), San Diego, California, USA, 2003. 153～162

3 Booch G, Jacobson I, Rumbaugh J. Uni. ed Modeling Language User Guide. Addison Wesley, 1998

4 Broy M, Slotosch O. From Requirements to Validated Embedded Systems. In: Embedded Software: First International Workshop, EMSOFT 2001. LNCS 2211, 2001

5 Wasowski A. On efficient program synthesis from statecharts. In: Proc. of the 2003 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'03). San Diego, California, USA, 2003. 163～170
6 Martin H, du Bousquet L. Automatic Test Generation for JavaCard Applets. Java on Smart Cards: Programming and Security, First International Workshop, JavaCard 2000, LNCS 2041,2001
7 Martin G, Lavagno L, Louis-Guerin J. Embedded UML: a merger of real-time UML and co-design. 9th International Workshop on Hardware/Software Co-Design (CODES'01), Estes Park, Colorado,2001
8 www. omg. org/docs/formal/03-09-01. pdf
9 Douglass B P. Real-Time UML: Developing Efficient Objects for Embedded Systems (2nd Edition). Addison-Wesley Pub Co; 2nd edition,1999
10 Alur R, Dang T, Esposito J M, et al. Hierarchical Hybrid Modeling of Embedded Systems. In: Embedded Software: First Intl. Workshop, EMSOFT 2001, LNCS 2211, 2001
11 Simsek T, Varaiya P. Design of Autonomous, Distributed Systems. In: Embedded Software: First Intl. Workshop, EMSOFT 2001, LNCS 2211,2001
12 Burch J R, Passerone R, Vincentelli A L S. Using Multiple Levels of Abstractions in Embedded Software Design. In: Embedded Software: First International Workshop, EMSOFT 2001, LNCS 2211,2001
13 Lee E A. Overview of Ptolemy Project : Technical Memorandum UCB/ERL M01/11, March 2001
14 Alexander P, Kong C. Heterogeneous Modeling Support for Embedded Systems Design. In: Embedded Software: First International Workshop, EMSOFT 2001, LNCS 2211,2001
15 de Alfaro L, Henzinger T A. Interface Automata. 9th ACM SIGSOFT Intl. Symposium on the Foundations of Software Engineering (ESEC/FSE 01), 2001
16 de Niz D, Rajkumar R. Time weaver: a software-through-models framework for embedded real-time systems. In: Proc. of the 2003 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'03). San Diego, California, USA, 2003. 133～143
17 Genβler T, Christoph A, Winter M, et al. Components for embedded software: the PECOS approach. International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), Greenoble, France, 2002
18 Sztipanovits J, Karsai G. Model-Integrated Computing. IEEE Computer, 1997. 110～112
19 Stankovic J A. VEST - A Toolset for Constructing and Analyzing Component Based Embedded Systems. In: Embedded Software: First International Workshop, EMSOFT 2001, LNCS 2211,2001
20 Gössler G, Sangiovanni-Vincentelli A L. Compositional Modeling in Metropolis. Embedded Software, Second International Conference, EMSOFT 2002, LNCS 2491,2002. 93～107
21 Butts K, Bostic D, Chutinan A, et al. Usage Scenarios for an Automated Model Compiler. In: Embedded Software: First International Workshop, EMSOFT 2001. LNCS 2211,2001
22 Neema S, Sztipanovits J, Karsai G, et al. Constraint-Based Design-Space Exploration and Model Synthesis. Embedded Software, Third International Conference, EMSOFT 2003, Philadelphia, PA, USA, Lecture Notes in Computer Science 2855 Springer, Oct. 2003. 290～305
23 Kernighan B, Ritchige D. The C Programming Language. Prentice-hall, 1987
24 Ada Programming Language Reference Manual. ANSI, 1983
25 Halbwachs N, Caspi P, Raymond P, et al. The synchronous dataflow programming language Lustre. In: Proc. IEEE, 1991,79: 1305～1320
26 Benveniste A, Guernic P L, Jacquemot C. Synchronous programming with events and relations: The Signal language and its semantics. Science of Computer Programming, 1991,16:103～149
27 http://java. sun. com/j2me/
28 http://java. sun. com/products/javacard/
29 http://www. rtj. org/
30 Beebee W S, Rinard M C. An Implementation of Scoped Memory for Real-Time Java. In: Embedded Software: First Intl. Workshop, EMSOFT 2001. LNCS 2211,2001
31 Corsaro A, Cytron R. Efficient memory-reference checks for real-time java. In: Proc. of the 2003 Conf. on Languages, Compilers, and Tools for Embedded Systems (LCTES'03). San Diego, California, USA, June, 2003. 51～58
32 http://www. rtjcom. com/home. html
33 Schultz U P, Burgaard K, Christensen F G, et al. Compiling java for low-end embedded systems. In: Proc. of the 2003 Conf. on Languages, Compilers, and Tools for Embedded Systems (LCTES'03). San Diego, California, USA, June 2003. 42～50
34 Shaylor N, Simon D N, Bush W R. A java virtual machine architecture for very small devices. In: Proceedings of the 2003 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'03). San Diego, California, USA, June, 2003. 34～41
35 Hartel P H, de Jong E. A Programming and a Modelling Perspective on the Evaluation of Java Card Implementations. Java on Smart Cards: Programming and Security, First Intl. Workshop, JavaCard 2000, LNCS 2041,2001
36 Chen G, Shetty R, Kandemir M T, et al. Tuning garbage collection for reducing memory system energy in an embedded java environment. ACM Transactions in Embedded Computing Systems (ATECS)27-55 Volume 1, Number 1, November 2002
37 Bacon D F, Cheng P, Rajan V T. Controlling fragmentation and space consumption in the metronome, a real-time garbage collector for Java. In: Proc. of the 2003 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'03). San Diego, California, USA, June 2003. 81～92
38 Henzinger T A, Horowitz B, Kirsch C M. Giotto: A Time-Triggered Language for Embedded Programming. In: Embedded Software: First International Workshop, EMSOFT 2001. LNCS 2211,2001
39 Kirshna C M, Shin K G. Real-time Systems. The McGraw-Hill Companies, Inc
40 Taha W, Hudak P, Wan Z Y. Directions in Functional Programming for Real(-Time) Applications. In: Embedded Software: First International Workshop, EMSOFT 2001. LNCS 2211,2001
41 Hoyme K, Driscoll K. SAFEbusTM. In: 11th AIAA/IEEE Digital Avionics Systems Conference, Seattle, WA, October 1992. 68～73
42 Miner P S. Analysis of the SPIDER fault-tolerance protocols. In: C. Michael Holloway, ed. LFM 2000: Fifth NASA Langley Formal MethodsWorkshop, NASA Langley Research Center, Hampton, VA, June 2000
43 Berwanger J, et al. FlexRay-the communication system for advanced automotive control systems. In: SAE 2001World Congress, Society of Automotive Engineers, Detroit, MI, April 2001. Paper number 2001-01-0676
44 Kopetz H. The Temporal Specification of Interfaces in Distributed Real-Time Systems. In Embedded Software: First International Workshop, EMSOFT 2001, LNCS 2211,2001
45 Culler D E, Hill J, Buonadonna P, et al. A Network-Centric Approach to Embedded Software for Tiny Devices. In: Embedded Software: First International Workshop, EMSOFT 2001, LNCS 2211,2001
46 Chakrapani L N, Korkmaz P, Mooney V J, et al. The emerging power crisis in embedded processors: what can a poor compiler do? 176-180 CASES 2001 Atlanta, Georgia, USA. ACM, 2001
47 Bertin V, Daveau J, Guillaume P, et. al. FlexCC2: An Optimizing Retargetable C Compiler for DSP Processors. Embedded Software, Second International Conference, EMSOFT 2002,

LNCS 2491 ,2002
48 Zambreno J, Kandemir M T, Choudhary A N. Enhancing Compiler Techniques for Memory Energy Optimizations. Embedded Software, Second International Conference, EMSOFT 2002, LNCS 2491 ,2002
49 Wirth N. Embedded Systems and Real-Time Programming. In: Embedded Software: First International Workshop, EMSOFT 2001, LNCS 2211,2001
50 Engblom J, Jonsson B. Processor Pipelines and Their Properties for Static WCET Analysis. Embedded Software, Second International Conference, EMSOFT 2002, LNCS 2491 ,2002
51 Cousot P, Cousot R. Verification of Embedded Software: Problems and Perspectives. In:Embedded Software: First International Workshop, EMSOFT 2001, LNCS 2211,2001
52 http://www.irisa.fr/manifestations/2003/MEMOCODE/
53 Dwyer M B, Deng Xianghua, et al. Space Reductions for Model Checking Quasi-Cyclic Systems. In:Embedded Software, Third Intl. Conf., EMSOFT 2003, Philadelphia, PA, USA, 2003. Lecture Notes in Computer Science 2855 Springer 173～189
54 Larsen K G. Resource-Efficient Scheduling for Real Time Systems. In: Embedded Software, Third International Conference, EMSOFT 2003, Philadelphia, PA, USA, 2003. Lecture Notes in Computer Science 2855 Springer 16～19
55 http://www.uppaal.com/
56 Ameur Y A, Bel G, Boniol F, et al. Robustness analysis of avionics embedded systems. In: Proc. of the 2002 Joint Conf. on Languages, Compilers, and Tools for Embedded Systems & Software and Compilers for Embedded Systems (LCTES'02-SCOPES'02), Berlin, Germany, June 2002. 123～132
57 Ahrendt W, Baar T, Beckert B, et al. The KeY approach: Integrating object oriented design and formal veri. cation. In:Proc. Logics in Arti. cial Intelligence (JELIA), Malaga, Spain, LNCS 1919. Springer, 2000
58 Bieber P, Cazin J, Marouani A E, et al. The PACAP Prototype: A Tool for Detecting Java Card Illegal Flow. Java on Smart Cards: Programming and Security, First International Workshop, JavaCard 2000, LNCS 2041,2001
59 Goré R, Nguyen L D. CardKt: Automated Multi-modal Deduction on Java Cards for Multi-application Security. Java on Smart Cards: Programming and Security, First International Workshop, JavaCard 2000, LNCS 2041,2001
60 Leavens G T, Baker A L, Ruby C. JML: A notation for detailed design. In:H. Kilov and B. Rumpe, eds. Behavioral Speci. cations of Business and Systems, Kluwer, 1999. 175～188
61 van den Berg J, Jacobs B. The LOOP compiler for Java and JML: [Techn. Rep. CSI-R0019]. Comput. Sci. Inst., Univ. of Nijmegen. TACAS'01. ,2000

第一届全国数据库应用技术研讨会

征文通知

由中国计算机学会数据库专业委员会主办、山东大学计算机科学与技术学院承办、山东地纬计算机软件有限公司协办的第一届中国数据库应用技术研讨会(CDAT2006)将于2006年10月在济南举行。作为对NDBC的强力补充，本次会议将展示全国数据库应用的最新技术和成果，为数据库研究者、开发者和用户提供一个数据库应用技术论坛，探讨数据库应用技术所面临的关键问题和发展方向。

CDAT2006的议题涉及数据库应用及应用平台的多个方面，届时国内外著名专家将到会作专题报告。同时邀请世界著名的数据库厂商就最新的应用工具和平台技术进行交流，满足国内在该领域日益增长的技术和应用需求。

CDAT2006诚邀各行业数据库工作者踊跃投稿！经评审录用稿件将在本次大会的论文集(《计算机科学》专刊)发表，优秀稿件将在《山东大学学报》正刊或推荐到《计算机科学》正刊发表。

1.征文范围 会议的主要方向包括(不限于此):

大规模数据库应用;数据库存储技术及实现;数据库备份技术;数据库平台和深度挖掘;数据库容灾技术;XML 数据库实现技术;查询处理技术;事务(日志)管理;分析型数据库系统实现技术;数据挖掘的实用技术;数据仓库的实用技术;中间件和应用服务器技术及应用;多媒体数据库技术及应用;生物信息系统应用。

2.投稿要求

作者投往本届大会的稿件必须是未发表的技术成果、工作经验。论文应包括题目、摘要、关键字、正文和参考文献;作者信息包括论文题目、作者全名、所属单位、电子邮件、通信地址、电话和传真。稿件以Word格式提交。

3.重要日期 征稿截止时间2006年4月25日 论文录用通知时间:2006年6月25日

来稿请寄:250100 济南市山东大学计算机科学与技术学院 洪晓光 教授 收

专用信箱:cdat2006@sdu.edu.cn

会议网址:http://www.sdu.edu.cn/cdat2006

联 系 人:宋婷婷 洪晓光 电话:0531-88169988转 传真:0531-88113508