# 嵌入式系统软件设计仿真方法研究报告

# 1 引言

## 1.1 研究背景

1.1.1嵌入式系统软件介绍

从20世纪70年代单片机的出现到现在各式各样的嵌入式微处理器、微控制器的大规模应用,嵌入式系统已经有了30多年的发展历史。经过这30多年的发展,嵌入式系统已经广泛应用到科学研究、工程设计、军事技术以及人们日常生活的方方面面,掀起了一场数字化技术革命。随着计算机技术和网络技术的不断发展，嵌入式系统愈来愈多地进入了人们生活的方方面面。比如，2011年全球嵌入式软件市场平均增长率为31%，中国嵌入式软件市场则超过了40%，居世界之首，2012年中国嵌入式软件产业销售收入已突破3000亿元人民币，而且在2015年底产业规模有望达到5000亿。嵌入式系统的广泛使用大大提高了人们的生活质量。

根据国际电气和电子工程师协会IEEE的定义,嵌入式系统(Embedded System)是“控制、监视或者辅助设备、机器和车间运行的装置”(devices used to control, monitor, or assist the operation of equipment, machinery or plants)。这主要是从应用上加以定义的,从中可以看出嵌入式系统是软件和硬件的综合体。而在很多时候一个嵌入式系统是否能 够很好完成使命，其关键可能不是在嵌入式系统的硬件，而是在嵌入式系统的软件。比如，1996年欧洲耗资高达7亿美元的Ariane-5 火箭发射后解体爆炸[1]。根据事后的调查报告，引起这个爆炸的原因就是惯性参考系统中的一个嵌入式软件中浮点数计算的设计缺陷。毫无疑问，一个优秀的嵌入式系统必须具有高质量的嵌入式软件。嵌入式软件作为嵌入式系统的“灵魂”，其质量对嵌入式系统可靠性保证的重要性不言而喻。但目前嵌入式软件的质量非常令人担忧，例如文献[2]显示，1990-2000年期间美国的心脏起搏器有200851台因为软件问题而被召回，占到总召回数的41%。实际上，心脏起搏器是一种功能相对单一的嵌入式系统。但受到低功耗、长寿命等条件的约束，其硬件部分设计相当复杂；相对的软件部分却不是十分复杂。但在总召回数中仍有超过4 成源于软件问题。可见，嵌入式软件的质量已经严重影响到整个嵌入式系统的质量和安全可靠性。

通过对各类嵌入式系统软件的调研分析发现，嵌入式系统软件主要面临以下三个方面的问题：1） 软件运行环境严格 。嵌入式系统软件是由硬件主导设计的，在软件实现的各个阶段都要考虑硬件环境的因素。由于嵌入式系统是应用于特定环境下执行专业领域的应用系统,所以对于不同的嵌入式系统来说，系统软件所需要考虑的硬件因素也各不相同。另外，嵌入式软件在每次运行过程中的实际环境也是不相同的，而且每次实际运行环境甚至是不可以重现的。2） 对软件质量要求苛刻。通用软件出现故障以后，可以通过软件升级来解决。但是嵌入式系统软件一旦出现故障，造成的损失是不能补救的。尤其对安全关键领域的嵌入式系统来说，任何一种情况的发生，所造成的损失都可能是毁灭性的，并且很难用金钱来计算。3）嵌入式系统软件通常都有实时性要求。20世纪90年代，随着对实时性功能急迫要求及各种应用软件的出现，导致软件规模数量不断上升，嵌入式操作系统已经开始出现新的变化，实时性变得非常突出，从而导致一场嵌入式系统研发的革命。嵌入式系统的软件设计对实时性有着很高的要求，因此，在对软件进行设计的过程中一定要充分考虑对实时性的要求。

1.1.2嵌入式系统软件的设计建模方法

随着嵌入式系统的发展，嵌入式系统结构越来越复杂、规模越来越大，对系统的开发成本、开发周期及非功能属性的要求也越来越高，早期的嵌入式系统开发方法已经不能满足当前需求。为此，业界引入了模型驱动结构方法(model driven architecture，简称MDA)[3]，系统开发被提升到更高的级别——模型级，针对特定计算平台的编码工作由机器自动完成，模型成为开发过程中的核心，从而可以在模型级别就对系统的功能属性和非功能属性进行分析。将软件需求描述与特定平台的实现分离是实现缩短软件开发时间、适应不断变化的需求、降低开发成本的有效手段。在基于MDA的软件早期设计阶段，目标系统的可选方案可以做充分的检查和比较。软件的开发风险因而可以被降低。

基于MDA的软件开发过程中，首先使用基于Meta Object Facility (MOF)[4]的建模语言，针对一个应用的业务功能构建平台独立模型PIM，该模型不随实现技术的变化而发生变化。MOF是OMG组织制定的元数据管理框架，OMG规范描述：“MOF提供一组元数据服务，这些服务可以完成基于模型的系统开发，以及系统间的交互。”在软件开发过程中，存在种类繁多的系统建模需求、各种各样的建模语言和工具、以及模型的表示方式和建模元素不一致性等问题，需要一种以一致方式描述建模语言的方法。MOF位于OMG四层元数据结构的元元层，提供了核心的基本建模构造。从而保证基于MOF定义的建模语言来描述MDA模型时，这些模型能够存储在与MOF一致的数据库中，并能被与MOF一致的工具解析和转换。

UML是典型的基于MOF的建模语言。对于很多规范来说，MDA的两层模型（PIM和PSM）都是通过UML来定义的，UML以统一的方式描述模型构造以及构造之间的关系。UML己经是事实上的软件建模标准，并且被OMG推荐用来描述MDA中的建模。然而UML无法描述系统非功能属性，而这些非功能属性在嵌入式系统软件开发过程中是非常重要的。软件系统的设计模型通常定义了系统的静态结构和动态行为，这对性能分析是不够的，还需要在设计模型中描述软件系统的某些非功能属性，为性能分析提供足够的信息。OMG最近针对嵌入式系统建模和分析的需要提出了MARTE Profile来描述嵌入式系统的非功能属性。

下面详细介绍这两种建模语言。

统一建模语言UML[5]是由Booch、Rumbaugh和Jacobson三人发起，在对象建模技术(Object Modeling Technology,OMT)和面向对象的软件工程(Object-Oriented Software Engineering, OOSE)的基础上,结合各种面向对象建模方法的优点后提出。其语义和概念由Rational等公司制定。UML是一种标准化的建模语言，它统一了建模的概念、术语和表示方法,成为软件开发人员和领域专家等共同交流的语言。UML2.0是当前最新的官方版本，为了适应MDA的需求，该版本改进了业务建模、体系建模、结构建模和行为建模4个部分。UML的设计目标是运用面向对象概念，在尽可能简单的同时对实际需要建立的系统的各个方面建模，建立起从概念模型直至可执行体之间明显的对应关系，创建一种对人和机器都适用的建模语言。

UML通过不同种类的图形从多个侧面对系统进行建模与描述，图形简单、直观且易于理解。它不但具备很强的描述能力，而且具有良好的可扩展机制，能适应不同领域系统建模的需要。UML共定义了13类图形。每一类图集合了相关的建模概念来表达一个特定的观点。这13类图划分为结构图(strueture diagram)和行为图(behavior diagram)，结构图表达系统的静态结构(static structure)，行为图表达系统的动态行为(dynamic behavior)。

随着UML2.0的提出，原有的非功能属性建模标准UML-SPT需要更新，而且UML-SPT在灵活性以及表达能力上有一定的欠缺。OMG组织为了更新现有标准，通过Profile对UML2.0 进行扩展，提出了实时和嵌入式系统的建模与分析（Modeling and Analysis of Real-Time and Embedded Systems，MARTE）标准[6]，它是一种专门针对实时嵌入式系统非功能属性建模与分析的需要而提出的一种UML Profile，为实时嵌入式系统的时间性能、调度性能分析提供了标准化的建模元素。MARTE并不提供分析设计模型的方法，而是为分析方法提供建模支持，它既可以描述系统的软件属性,也可以描述系统的硬件属性,并且与UML2和MDA标准相兼容。由于MARTE是对UML2.0的一种轻量级扩展[7]，就是使用构造型(stereotype)、标记值(tagged-value)和约束(constraint)来扩展UML的建模能力，所以可以很方便的同UML描述的设计模型相结合，表示性能相关参数。MARTE利用它的值描述语言(Value Specification Language,VSL)可以定量或定性地描述UML模型中的非功能属性。VSL允许开发者在UML模型上标记特定属性的数据类型、值、变量、约束、表达式等。

MARTE语言标准主要包括四部分：一个定义了实时嵌入式领域基本概念的核心框架；对核心框架的第一次扩展用来支持设计模型中非功能属性的建模；对核心框架的第二次扩展用来支持对设计模型的性能分析，特别是调度和时间性能分析；最后一部分包含了MARTE规范所有的附录，例如定义了一种可以在UML2中使用的值描述文本语言，以及囊括了所有MARTE规范的标准模型库。MARTE语言标准有如下三种包：1）MARTE核心包。MARTE核心包由非功能属性建模（Non-functional Properties Modeling，NFPs）、时间建模（Time Modeling，Time）、通用资源建模（Generic Resource Modeling，GRM）以及部署建模（Allocation Modeling，Alloc）四部分组成。2）MARTE建模包。MARTE建模包由通用组件模型（Generic Component Model，GCM）、高层应用建模（High-Level Application Modeling，HLAM）、软件资源模型（Software Resource Model）以及硬件资源模型（Hardware Resource Model）四部分组成。3）MARTE分析包。MARTE分析包由通用量化分析模型（Generic Quantitative Analysis Modeling，GQAM）、调度分析模型（Scheduling Analysis Modeling，SAM）以及性能分析模型（Performance Analysis Modeling，PAM）三部分组成。

嵌入式软件是实现嵌入式系统功能的关键，根据嵌入式软件的特点和实际运行过程中的具体要求，在进行嵌入式系统软件设计时，尤其需要考虑以下两方面因素：1）软件的实时性设计。嵌入式应用一般都有实时性要求，对于实时系统来说，如果逻辑和时序出现偏差将会引起系统的严重后果。根据实时系统对时间的要求不同，可以划分为两种类型的实时系统：软实时系统和硬实时系统。在嵌入式系统软件设计时，要充分考虑实时性要求。2）软件的可靠性设计。嵌入式应用一般也具有可靠性要求，除硬件必须具备的可靠性之外，软件的可靠性也必不可少。在软件设计中要充分考虑软件运行条件，减少程序错误执行的可能，在程序运行错误的情况下能够进行错误处理，并重新恢复软件的正确执行。

1.1.3嵌入式系统软件的仿真分析

软件设计人员在实现嵌入式系统软件时，除了考虑功能需求外，还必须考虑其性能需求。如果仅仅在系统（包括硬件和软件）实现后对系统进行功能和性能分析，可能会导致整个系统的重构。而且由于嵌入式系统的特殊性，甚至在软件实现后硬件系统都还未生产，这就给嵌入式系统软件的功能分析和性能分析带来不便。另外，通常软件的性能问题是由设计缺陷造成的，不是代码优化能够解决的。所以需要一种在嵌入式系统软硬件都未实现的情况下，在设计阶段对嵌入式系统软件的功能需求，尤其是性能需求进行仿真分析。软件性能工程[8]（Software Performance Engineer，SPE）由C.U. Smith 于1990 年提出，是一种构建符合性能目标的软件系统的系统化、量化方法。软件性能工程主要通过软件系统的用例以及述用例的场景分析系统性能[9, 10] 。实践软件性能工程的一般过程：设计人员首先确定关键用例和性能度量目标，然后绘制用于性能分析的性能分析模型，并根据评估结果修改性能目标或者现有设计，直到设计能够满足既定的性能需求。

嵌入式软件仿真面临的主要困难就是软件运行场景难于构造[11]。造成这一困难的根本原因是嵌入式系统软硬件结合紧密而硬件的可控性又十分有限。良好的嵌入式系统软件的仿真方法可以增强嵌入式系统的可控性，从而构造出更多软件运行场景。对嵌入式系统进行仿真的方式非常多：主要包括实物（环境）仿真、半实物（又名硬件在环，Hardware-in-Loop）仿真和全数字（又名Software-in-Loop，软件在环）仿真。实物仿真和半实物仿真都是首先构造一个嵌入式软件运行场景，并通过影响到嵌入式硬件输入手段，改变嵌入式软件的输入，而不能直接改变嵌入式软件的输入。因此，这些仿真方式对于嵌入式软件来说都是低效的。与此不同，全数字仿真平台可以使软件运行完全脱离真实硬件环境，突破各种硬件限制，构造各种可能的场景，实现运行过程中分析的自动化，并显著节约成本。同时，全数字仿真平台相对于目标机硬件环境和其他仿真方式来说，结构更简单，具有可控性强、自动化程度高、成本低等先天优势。

## 1.2 问题提出与研究目标

在新军事革命条件下，各航母国家的军事战略中仍将航母视为海军的核心力量，正在发展的航母技术包括很重要的一项就是如何改进和提高着舰引导系统在夜间和恶劣天气的作业能力。舰载机着舰引导系统是空间相对运动动力学和控制技术的一种应用。舰载机在着舰过程中，航母上的精确跟踪雷达实时测得飞机的实际位置和运动情况，并将这些参数输入计算机，得出舰载机正确的着舰位置，并将舰载机的实际位置和正确位置在计算机中进行比较，然后将修正指令发送到舰载机的终端设备，利用舰载机自动驾驶仪修正误差，完成准确着舰。目前基于雷达技术的舰载机着舰远程引导体制存在测控过程复杂、定位精度有限以及无法实施多机同时测控等缺陷。在这种情况下，着舰引导系统的嵌入式软件的任务将变得非常重要。

由于嵌入式软件和硬件结合十分紧密，嵌入式软件很难脱离实际硬件环境运行，这造成软件设计与分析对硬件因素依赖程度很高。对于舰载机着舰引导系统来说，除了1.1.1节中提到的嵌入式系统软件共有的问题外，还有其自身的特点：比如价钱昂贵，在软件设计阶段硬件还没有实现，而且硬件设备可能是由多家共同生产。因此，如果软件设计与分析不能减少对硬件设备的依赖程度，将在一定程度上制约嵌入式软件质量的保证。

本项目从嵌入式系统软件的设计分析着手，在设计阶段利用UML/MARTE工具对嵌入式系统设计建模方法，并利用嵌入式软件设计仿真方法帮助开发人员通过仿真验证的方式完成软件概要设计和软件详细设计。主要解决以下两个关键问题：1）利用UML系列语言定制一套适合描述嵌入式系统的语言来分层次、多视角的描述系统完成软件设计。利用UML/MARTE建模来覆盖完整的功能，同时完备的表达系统实时性的影响因素；2）研究嵌入式系统软件设计仿真方法，并利用仿真工具验证软件功能的完备性和系统的实时性。

# 2. 研究内容与拟采取的研究方案

## 2.1 研究内容与技术路线

为实现在独立于硬件环境下进行舰载机着舰引导系统的软件设计，本项目在虚拟软件运行环境基础上，建立一套嵌入式系统软件设计和仿真验证方法，并完成相关工具的开发，如图1。主要包括基于虚拟硬件等效模型的软件设计技术研究和基于虚拟硬件等效模型的软件仿真验证技术研究。主要分为以下两部分研究内容：

图1 研究内容

1. 嵌入式系统软件设计技术研究

为了便于在嵌入式系统软件设计的早期阶段对设计进行验证，本项目在虚拟硬件环境基础上采用基于模型的软件设计，拟基于UML系列语言定制一套适合描述嵌入式系统的语言来分层次、多视角的描述系统完成软件设计。软件设计的建模过程要覆盖软件需求的完整功能，同时能够完备地表达嵌入式系统中影响实时性的因素。另外，在将硬件虚拟化时，系统的各部分之间通过接口等通道进行通信，为了提高软件设计过程中各部分的兼容性，本项目拟提出一套接口匹配规范，用于约束硬件设备等效模型与软件模块之间，软件组件与组件之间的交互。

本部分的研究内容包括以下几个研究点：①通过硬件虚拟化、硬件软件化技术，建立硬件设备等效模型以及部分软件组件，来代替嵌入式系统软件的真实运行环境；②基于MARTE扩展UML，针对嵌入式系统软件特点，建立一套多粒度、多视图的软件系统设计建模方法，并定制一套适合描述嵌入式系统的语言，通过基于UML/MARTE的时间特征以及资源分配的定义，来对系统功能完备性和时间约束可满足性进行动态语义描述，完成软件设计；③为了提高软件设计过程中各虚拟部分的兼容性，提出一套接口匹配规范，用于约束硬件、设备等效模型与软件模块之间，软件组件与组件之间的交互。该规范标明交互的前提以及双方交互后的输出。

1. 嵌入式系统软件设计仿真验证技术研究

嵌入式系统软件仿真验证的主要目标是嵌入式系统软件设计中功能的完备性和系统实时性，针对此验证目标本项目拟对嵌入式系统软件设计仿真方法进行研究：在仿真过程中，利用具有明确语义的形式化语言去描述系统行为，确定行为分支条件；在时间属性仿真过程中，在行为语义之上加入时间约束描述机制，主要包括进程/组件的周期特性、调度特性、执行时间特性以及通信延迟等因素，然后通过引擎的执行以获取系统事件执行之后的时间特征。

本部分的研究内容包括以下几个研究点：①定义行为的形式化语义，定制软件功能的完备性规则。②为了实现系统时间属性的仿真，根据软件行为的形式化语义，定义执行时间序列和时间累加规则。③定制针对嵌入式系统软件的功能完备性和系统实时性性能指标，来量化软件设计建模过程中的功能和性能效果。④基于嵌入式软件设计方案的仿真及验证技术，进而完成嵌入式系统软件方案设计仿真工具的开发。

## 2.2 技术路线与关键技术

本项目首先对嵌入式系统中的硬件进行虚拟建模，并建立模型之间统一的接口规范；在虚拟硬件模型基础上，针对嵌入式系统软件的功能需求和实时性需求，通过MARTE扩展UML建模语言来设计建模；在仿真分析阶段，将得到的模型转换为仿真指令和相关的配置文件，根据预先定义的软件运行过程中行为的形式化语义描述方法，在仿真引擎上对这些仿真指令解释执行，并对得到的功能和实时性能指标进行分析，进而来辅助/更改设计模型。本项目的关键技术主要包括嵌入式系统软件设计建模方法的构造，软件行为的形式化语义描述以及仿真引擎的对输入模型的解释执行两个方面，具体技术路线如图2所示。

图2 技术路线图

1. 软件设计建模方法

本项目考虑的嵌入式系统软件的设计建模主要包括功能和实时性能两个方面，因此需要从功能和性能两个视图进行设计建模。首先根据嵌入式系统特征和应用需求，定义一组功能和实时性能度量指标，为度量系统设计好坏的量化依据，然后重点分析影响实时性能度量目标的因素，确定面向实时性嵌入式系统软件的建模内容，并通过选取UML建模元素、借鉴MARTE语言扩展UML，设计嵌入式系统软件设计分析的建模语言。

为了在设计阶段分析嵌入式系统软件的时间性能，需要在设计模型中添加与实时性性能相关的参数，嵌入式系统中时间性能差异主要来自两个方面：一是取决于系统设计对于有限软硬件资源的使用、争夺和调度，二是取决于外部负载，在不同的外部负载下，系统的响应时间往往差异很大，这是因为不同的外部负载所引起的系统内部的资源争夺程度不同。因此需要从资源、外部负载以及操作语义三个方面对系统设计模型（包括系统静态结构和动态行为两个方面）进行扩展。然后针对嵌入式系统给出实时性相关参数所包含的具体内容，作为扩展UML的依据，通过借鉴MARTE语言对实时性相关参数的建模要素，以及MARTE通用资源包在不同粒度上建模的方法，给出利用UML的Profile 扩展机制定义的实时性性能标签和操作语义，提出一种面向实时性分析的嵌入式系统软件设计建模语言。该软件设计建模语言同时还定义了一套嵌入式系统功能/实时性属性的度量指标，用于量化软件设计模型的好坏。

1. 软件行为的形式化语义描述及仿真实现

根据通过（1）建模方法得到的设计模型作为输入，设计并实现描述系统行为和时间需求的功能/性能仿真指令，然后设计并实现仿真引擎来实现仿真指令的解释执行与分析。

嵌入式系统软件设计模型作为设计仿真的输入，首先需要转换为仿真引擎的输入指令，从顺序图、活动图描述的系统行为以及功能/实时性能相关参数的角度出发，通过分支、顺序和循环控制结构指令描述系统的执行流程，通过具体的指令属性描述功能/实时性能相关参数，从而建立一套嵌入式系统软件行为的形式化语义描述方法。该方法设计了行为和实时性能仿真指令来描述系统设计的动态行为及其资源需求，并定义了配置文件的格式。其中配置文件是对外部负载和资源特性的描述，而描述系统及资源需求的建模元素源自UML顺序图、活动图及其扩展。然后，需要对仿真指令的实现进行设计，这些仿真指令主要有原子指令、复合指令、分支指令、循环指令、同步指令和异步指令等。最后，构造一个仿真引擎来实现对这些仿真指令的执行和解释，通过设计模型中建立的功能/实时性度量指标，来分析软件设计模型的运行情况。

为了适应分布嵌入式系统软件仿真的要求，本项目还提出在嵌入式系统软件仿真平台的基础上，扩充实时网络仿真功能，这对嵌入式系统软件的实时性特征分析具有非常重要的意义。该部分将一个独立嵌入式系统抽象为一个网络节点，主要研究跨节点的仿真技术，其目标是对包含多个网络节点的整个分布嵌入式系统进行仿真，重点解决分布嵌入式系统具有一些单个嵌入式系统不具备的特殊问题，如缺乏全局时钟、网络不稳定等。很多分布嵌入式系统的正常运转不仅依赖于逻辑时间的准确性，还依赖于绝对时间测量的准确性，因此建立全局统一的仿真时钟在嵌入式系统软件设计仿真实时性分析过程中是非常关键的。

# 3.总结

本项目以舰载机着舰引导系统为背景，基于虚拟硬件等效模型，针对嵌入式系统软件的功能和实时性两方面，建立一套嵌入式系统软件设计建模和仿真验证方法。在方案设计阶段，该嵌入式系统软件设计方法能够帮助开发人员通过仿真验证的方式完成软件概要设计和软件详细设计，并根据设计生成相应的模型代码供虚拟调试环境使用。此外，在软件开发阶段，该方法有助于指导开发计算机在硬件模型库的基础上搭建虚拟测试环境，并通过调试工具对模拟平台进行调试优化；最后在系统级设计验证阶段，该方法通过分布式仿真验证环境对目标虚拟平台进行系统级验证。因此，在整个嵌入式系统软件生命周期过程中，本项目建立的方法都具有重要的意义。

# 参考文献

[1] Lions J L, et al. Ariane 5 flight 501 failure[EB/OL],1996.

[2] WH M, MO S, WG S,et al. Recalls and safety alerts involving pacemakers and implantable cardioverter-defibrillator generators[J].JAMA,2001,286(7):793–799.

[3] OMG. Model driven architecture (MDA). 2010. http://www.omg.org/mda/

[4] Poernomo I. The meta-object facility typed[C]//Proceedings of the 2006 ACM symposium on Applied computing. ACM, 2006: 1845-1849.

[5] Fuentes-Fernández L, Vallecillo-Moreno A. An introduction to UML profiles[J]. UML and Model Engineering, 2004, 2.

[6]OMG UML profile for MARTE, beta2. <http://www.omgmarte.org/Specification.htm.> 2008.

[7]S.Bemardi, J.Merseguer, D.C.Petriu. A dependability profile within MARTE[J]. Software and Systems Modeling. DOI:10.1007/s10270-009-0128-1.

[8] Smith C U. Performance Engineering ofSoftware Systems[J]. Addison-Wesley, 1990, 1: 990.

[9] Woodside M, Franks G, Petriu D C. The future of software performance engineering[C]//Future of Software Engineering, 2007. FOSE'07. IEEE, 2007: 171-187.

[10] Bittner K. Use case modeling[M]. Addison-Wesley Longman Publishing Co., Inc., 2002.

[11] 刘辉. 分布嵌入式软件仿真测试关键技术研究[D].博士学位论文,2013