嵌入式数控系统模型层验证设计策略研究

郑建华[1,2] 李 迪[1] 朱 蓉[2] 肖苏华[1] 苏兆港[1]
(华南理工大学机械工程学院 广州 510640)[1] (仲恺农业技术学院 广州 510225)[2]

摘 要 在分析传统嵌入式数控的开发方法上的不足,以及目前对其开发方法的诸多变革的基础上,提出并实现将特定领域模型驱动开发融入到嵌入式数控系统的开发,构建了嵌入式数控系统模型层验证框架。该方法为嵌入式数控系统的开发提升了相应的抽象层次,提高了模型在系统开发中的作用,并利于提高系统的可操作性、可靠性、开发效率。通过对数控系统不同工作模式在第三方工具(StateFlow)的仿真实例,详述了该方法在特定领域建模语言的构建、模型转换的实现等方面的细节,并给出了转换后的对应 StateFlow 模型图。

关键词 数控系统,特定领域,模型驱动,模型转换

Design Strategy for Embedded CNC Model Verification

ZHENG Jian-hua[1,2] LI Di[1] ZHU Rong[2] XIAO Su-hua[1] SU Zhao-gang[1]
(College of Mechanical Engineering, South China University of Technology, Guangzhou 510640, China)[1]
(ZhongKai University of Agriculture & Technology, Guangzhou 510225, China)[2]

Abstract After analyzing the shortcoming of traditional development methodology for embedded CNC and the evolution of development methodology, domain specific model driven development methodology used to embedded CNC was proposed and implemented, and an architecture for model verification of embedded CNC was constructed. The methodology heightens the abstract layer of embedded CNC development and enhances the function of model during the development life cycle, which improves the interoperation, reliability, develop efficient of the system. By a concrete simulation example of CNC working mode during the third simulation tools(StateFlow), the paper details the construction of Domain specific language and realization of model transformation, in the end the corresponding StateFlow diagram obtained by transformation is listed.

Keywords Computer numerical control, Domain specific, Model driven, Model transformation

1 前言

至1952年世界上第1台三坐标数控铣床在美国麻省理工学院诞生以来,在短短的几十年内,数控系统经历了五代发展历程,并广泛应用于各行各业,成为制造业的重要技术支柱之一。但在我国生产的数控机床中,42.4%是中档(普及型)数控机床,57.5%是经济型数控机床,而高级型数控机床只占0.1%,中高端数控系统基本为德国西门子和日本法兰克(FANUC)两家公司所垄断。近年来,国务院、国家发改委、科技部等机构连续发文,明确提出了鼓励装备制造业发展的战略举措,并把数控技术的创新和提高置于前所未有的重要地位[1,2]。广东省作为工业大省,2005年也提出将高速、精密数控机床及数控系统列为重点发展的产品,强调中高档数控系统将成为国产数控产业的发展趋势[3]。纵观国内数控技术的研究现状,某些控制理论的研究以及高性能五轴联动高端数控系统的开发已经达到或者接近了国际先进水平,但离实际应用和产业化还有相当大的距离。本文认为其主要矛盾已经不在于控制算法,而在于系统的设计策略和方法学上,特别是CNC的软件系统的设计策略及方法已经成为制约数控系统性能根本性提高的瓶颈。

2 传统嵌入式数控系统开发的不足

数控系统是一个典型的复杂实时多任务嵌入式系统,然而嵌入式数控系统的开发,大都采取以程序代码为中心的强耦合的过程式开发方法,它存在着如下问题:

(1)开发过程的各个环节关联稀疏

传统的顺序式软件设计方法延误了产品开发的进度。主要的缺点是高层次的系统分析抽象(包括系统的需求分析与描述、设计)阶段和目标环境中的开发编程阶段之间脱节,系统的开发常常是从技术细节开始,对整个系统缺乏全面而清晰的了解。而且开发初期所做的需求分析及设计,在后期的实现和测试中经常变动,甚至可能抛弃前期的设计方案。

(2)缺乏面向领域的开发语言和环境

从事嵌入式系统开发的工程师一般是各个应用领域的专家,对所要控制的目标系统有较深的理解,但往往缺乏足够的硬件和软件知识。而传统的嵌入式系统设计则要求设计者拥有深厚的硬件和软件知识,这就局限了嵌入式系统的进一步应用。

(3)系统测试滞后,系统可靠性难以保证

目前基于手工编码方式开发的嵌入式数控系统,其功能验证和性能验证要等到系统完成后在问题域中通过测试用例来验证所设计的系统在各种测试用例中的正确性。然而,由经验可知软件测试并不能保证系统的完全正确性,其只能尽最大可能地发现错误并修改。随着系统大型化、系统复杂性的增加,通过测试来保证系统可靠性是越来越难,并且越到后

郑建华 博士研究生,讲师,研究方向为计算机应用、嵌入式控制系统。

面发现错误则修复花费的代价也越大。

(4)代码可重用性差,开发效率低,成本高

采用编码的开发方式,数控系统需求的变化,需要人工参与编码的修改过程,有时微小的需求变动,需要大量的编码过程,所开发出来的代码可重用性低;另一方面,软件部分的开发和调试是基于所选择的硬件平台进行的。在嵌入式硬件设计过程中,人的经验主导了整个设计和实施过程,效率较低。

从上面可以看出,嵌入式数控系统的开发存在着很多的挑战。现有的开发方法已不能适应嵌入式数控系统的开发,因而必须在开发方法上进行创新,寻求适合嵌入式数控系统开发的新途径。

3 特定领域模型驱动数控系统开发

根据 IEEE 对开放式数控系统的定义,开放式数控系统必须是一个全模块化的软件体系结构,这在一定程度上为软件开发了提供开发方法上的思路。组件是可复用的、自包含的软件对象模块,组件之间可以在二进制级别上进行集成和重用。正是由于这个特点,目前许多学者都在研究基于组件的嵌入式数控系统开发方法。但目前通用的组件,如 COM/DCOM,JavaBean,CORBA 等都是基于统一的运行平台和结构,通过功能接口实现组件合成。这些组件标准在嵌入式实时系统中不适合使用[4,5],其并没有充分考虑到系统的非功能方面的性能。虽然业界在此方面做了许多的工作[6-9],包括构建实时组件等,但实际上只是改正上述的第四个不足,而对其它三个缺点的改进则显得不足。

对象管理小组(OMG, Object Management Group)于 2001 年正式提出的模型驱动体系结构,MDA 不是一个实现分布式系统的软件体系结构[10],而是一个用模型技术进行软件开发的方法。它通过合理的抽象建立系统模型,用图形化的建模语言(如 UML),对系统进行分析和设计乃至最终代码的生成,在这个过程中实现了系统说明和实现技术与平台独立开来,并且模型贯穿于整个软件系统生命周期的各个阶段,具有模型驱动开发方法的 5 个关键要素:全功能代码产生、模型可执行、模型←→代码关联、基于模型的自动化测试和实时框架。这样的开发设计方法在极大程度上纠正了上述 1,3,4 不足。华南理工大学针对 CNC 系统,研究了基于 MDA 的嵌入式软件设计方法[11]。该方法使用 UML 进行问题域建模,通过多次迭代开发出经过单元测试的运动控制器、辅助设备控制器、数控代码检查器、数控代码解释器等单元 UML 模型。上海交通大学的杜道山在文献[12]中采用 MDD(Model Driven Development)软件设计的思想和开放式模式设计软件的模型结构,分析设计了数控系统的软件开发途径,提出了判断引擎和模式转换规则库相结合的数控模式仲裁模块设计。

但 MDA 将 UML(Unified Modeling Language)作为它的标准建模语言,虽然 UML 在应用上取得了很大的成功[13,14],但是 1)UML 并不是真正的形式化语言,并不能保证模型的清晰明确性和无二义性,并且是正确的;2)UML 在系统的可靠性、资源占用以及硬件环境等方面没有给出充分的描述和规约,虽然提供了可扩展性机制,但还是难以胜任实时嵌入式系统或复杂系统的建模;3) UML 中的语言元素定位于从软件设计角度对事物的抽象描述,它是通用的、没有面向领域的概念,不容易为领域专家所理解和掌握,要求数控领域工程师也必须掌握该语言有一定困难。也就是说,MDA 方法不能解决前面提出的第 2 点不足。

而特定领域建模[15]是通过对特定领域的分析和抽象,得到该领域的共性和变化特征,建立该领域的构件库,最后通过领域建模和代码生成实现领域应用。特定领域建模比模型驱动更能够提高开发效率,缩短开发周期。在这过程中生产的特定领域建模语言 DSML 将直接面对领域开发工程师,是对特定领域的抽象,而不是对代码的抽象,它符合特定领域的使用习惯,具有特定领域的语义,容易被领域工程师所掌握。故我们应该通过元模型语言对实体、关系、属性和约束的定义和描述,建立数控领域的建模语言,并规定数控领域模型和运行平台的指令集之间的映射,这样数控开发工程人员就可以利用自己的领域知识,根据自己的实际需要搭建系统的模型,而不需要太多的软件开发基础。很显然,该方法融合了 MDA 的优点,并且拥有自身的面向领域的优点,非常适合嵌入式数控系统的开发,恰好也修正了前面所提到的目前开发方式的不足。

4 嵌入式数控系统模型层验证框架

从软件开发的发展过程来看,可以明显地看出测试与开发流程融合的趋势。特别是上世纪 90 年代以后,软件的规模和复杂程度迅速提高,这种形式上的融合也迅速走向更深层次,更具实际意义。具体地说,这种融合就是整个软件开发活动对测试的依赖性。有资料表明,在维护阶段发现错误并改正,其花费的费用将比在开发阶段的费用高出 40～60 倍[16]。对于嵌入式系统来说,其开发涉及到硬件、软件及其综合,工作量大,难度高。不仅要考虑功能性的流程设计,更要注重任务进行时间行为的正确性设计,特别是对一个硬实时系统必须满足死线,否则就发生灾难,这种灾难是极大的并且可能危害人的生命。在这种情况下,尽早进行系统验证,在系统开发周期的早期发现问题,其意义是显而易见的,可以大大地减少由于设计问题造成的系统重新开发的工作量,从而降低项目的成本,加快项目的进度。因此对于实时嵌入式系统的功能性、实时性、可靠性等特性的验证变得非常重要。

为了保证嵌入式系统各项功能和性能,目前系统验证一种趋势就是把模拟仿真与形式化、半形式化的方法结合起来形成一种混合验证的方法,即使用两种建模语言。如文献[17]提出的使用 UML 进行系统分析和设计,然后将此模型转换为形式化语言的表示形式,并用形式化语言进行验证,并借助于第三方的工具进行系统验证。

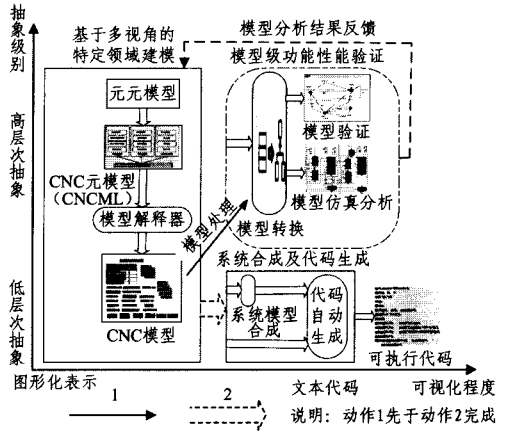

图 1 嵌入式数控系统模型层验证框架

根据前面分析,结合数控系统模型验证的要求,建立特定领域模型驱动开发数控系统模型验证框架如图 1。图中表明

的系统的开发过程是一个迭代过程，首先由数控领域建模语言建立用户模型，然后通过模型转换技术转换到另一种语言描述的模型，并进行相关的仿真和验证。结果反馈给用户模型，通过修改用户模型，并在此仿真，最终保证系统模型层的可靠性、正确性。而目前对许多的形式化语言都有比较好的仿真工具（如 Matlab 和 Uppaal[18]）的支持，故在进行模型转换时只要将其转换到第三方建模工具模型即可。而当系统验证完成后可以直接利用系统合成框图里的代码生成技术得到基于平台的可执行代码。

在这个过程中存在两个关键因素：1）数控领域建模语言的建立，特别是在描述数控系统在功能及性能方面。2）从数控领域模型到第三方仿真工具模型的转换。下面通过实例来描述这两方面的内容，以及表明该框架的可行性。

5 数控系统逻辑功能 StateFlow 验证

数控系统的性能包括系统的功能性能和非功能性能。功能性能是指系统的逻辑性能，即系统行为的正确性。逻辑性能体现了系统是否满足设定的功能需求，各个组成部分是否正确地执行任务，其间的相互关系是否正确。比如数控车床具有 6 种工作模式，每种工作模式下又有 6 种工作状态，一共 36 种工作状态，如何保证这些工作状态之间的切换是正确无误的呢？对此可以采用 Matlab 中的 StateFlow 或者 UPPAAL 来验证。Matlab 中的 StateFlow 是有限状态自动机的图形工具，它可以用于解决复杂的逻辑问题，用户可以通过图形化工具实现在不同状态之间的转换。在 StateFlow 的仿真窗口中，允许用户建立有限个状态以及状态转移的条件与事件，从而可以绘制出有限状态自动机系统，这样就可以实现对系统的仿真。故本部分建立数控系统的工作模式建模语言，以及实现领域模型到 StateFlow 模型的转换。

5.1 特定领域建模语言（DSML）的角色

根据图 1 的开发框图，本系统的关键是为领域开发者提供一套 CNC 领域建模语言和模型验证集成工具集。图 2 表明了建模语言开发者和领域开发者之间的关系。图中左边矩形中描述了建模语言开发者利用元元模型创建元模型，而该元模型则描绘了领域的元素，经解释后即为领域开发者眼中的领域建模语言，其与目前的各种语言一样，具有自己的语法和语义。

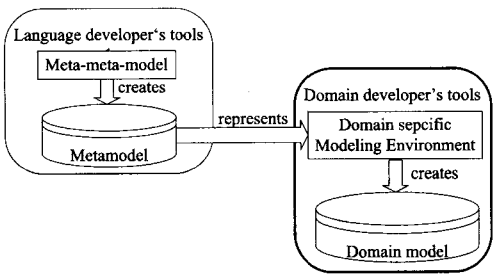

图 2 建模语言开发者和领域开发者之间的关系

5.2 数控系统工作模式建模语言（元模型）

本实例以车床为例，其有自动、编辑、录入、手动、手轮、回零 6 种工作模式，而每种模式下面又可以切换 6 个状态：位置、程序、参数、诊断、图形、偏置。这 6 种模式以及模式中的 6 种状态应该可以自由切换。本实例采用 ISIS 的 GME[19] 建模工具，构建数控车床的工作模式元模型（如图 3）。

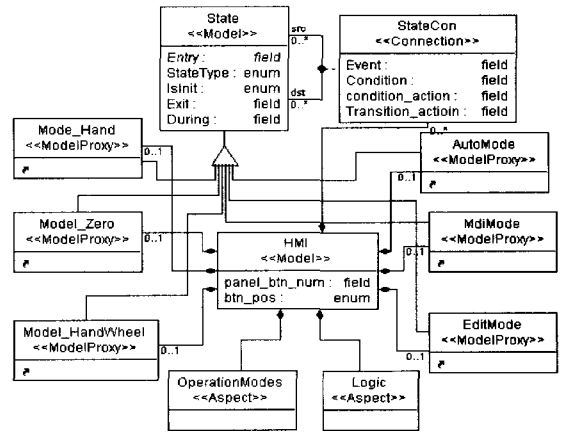

图 3 数控系统工作模式建模语言（部分）

该图形的符号采用 UML 类图的表示形式，并可以采用 OCL2.0 约束语言来进行系统的良式规则定义，如图 4 所示表示了 HMI 模型元素只能在系统建模中出现一次。

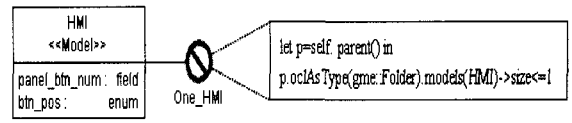

图 4 HMI 唯一性约束 OCL 表示

另外，从图 3 中可以看出，一个数控车床的人机界面是含有 6 个 ModelProxy，分别代表自动、编辑、录入、手动、手轮、回零 6 种工作模式。而这里的 ModelProxy 是 GME 一种表示代理机制实现方式，表示在其他视图或者层次有该种对象的实际定义。

为了表达各工作模式之间的转换关系，本元模型引入了有限状态机的概念。一般而言，一个 FSM 是一个标准的五元组：

$FMS=(Q,E,\delta,q_0,F)$ 其中，Q 为有穷状态集；E 为有穷事件集；δ 为从 $Q\times E$ 到 Q 上的映射或转移函数；$q_0\in Q$，为初始状态；$F\subseteq Q$ 为终止状态集。

图 3 中 State 和 StateCon，以及其含有的属性共同构成了一个 FSM 应用的 5 个组成部分，图中的 State 构成了状态机的有穷状态集以及终止状态集。而 StateCon 表示两个状态之间的迁移，并且通过其参数 Event 来描述系统的有穷事件集，而同时迁移的触发条件等则由 StateCon 中的 Condition，Condition_Action，Transition_action 来描述。State 中的 IsInit 枚举类型属性则表明了该状态是否为初始状态，并通过其他属性指出了该状态的其他状况。为了建立与车床各个工作模式之间的关系，本元模型采用了面向对象中的继承机制，使自动、编辑、录入、手动、手轮、回零 6 种工作模式都继承至 State，使得其都具有一个状态的特性。

但是，为了降低用户建模的复杂性，提高系统设计的清晰程度，本元模型还体现了多视角设计（Multi_Aspect）的理念。在图 3 中共构建了两个视图，即图中的 OperationModes 视图和 Logic 视图，通过视图的设定，可以定义在元模型某个模块中那些模块是可见的，隐藏了其他方面的细节，从而分解设计空间，使用户可以专注于一个方面问题的考虑，降低用户建模的复杂性。在该元模型中 Logic 视图则正好表达了系统进行逻辑功能建模的概念，表示了系统的状态及其转换。而在 OperationModes 则显示的是系统真正执行的操作或行为。

如图 5,6 中表示了在这两个视图下的建模。

对于各个工作模式下的不同工作状态的子元模型图可以根据上述的建模规则来构建。图 7 表示了自动模式下的不同工作状态元模型图(部分)。

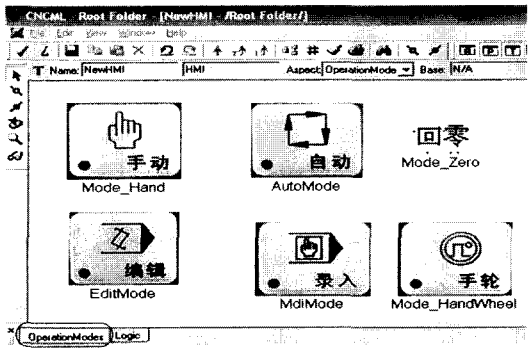

图 5 OperationModes 视图

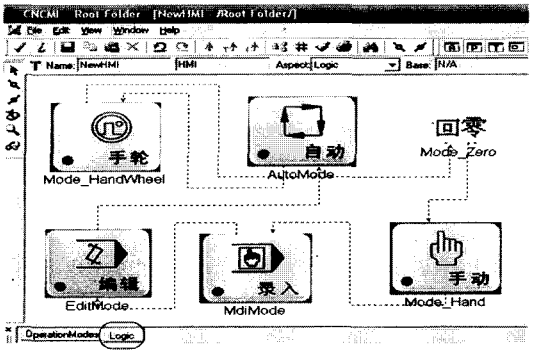

图 6 Logic 视图

5.3 模型转换

5.3.1 模型转换的定性

在基于模型驱动开发的系统开发周期中,模型已经从传统的文档角色上升到系统主要部件的角色。系统的实体或者系统的架构都是作为模型或者模型的元素而存在,系统模型中包含足够的细节用以保证能够从这些模型中生成整个系统的实现。在这整个基于模型为中心的周期中,各个不同的周期模型有不同的表现形式,比如系统需求分析阶段的平台无关模型,到后来的平台相关模型,以致最后的代码都不约而同地依靠模型转换技术,这是基于模型驱动开发的关键问题[20]。根据其定义,模型转换根据转换定义和转换规则从源模型到目标模型自动生成的过程。根据模型转换的特点我们可以将其分为内生转换和外生转换。内生性转换是指用相同语言描述的模型间的转换,外生性转换则指的是不同的模型

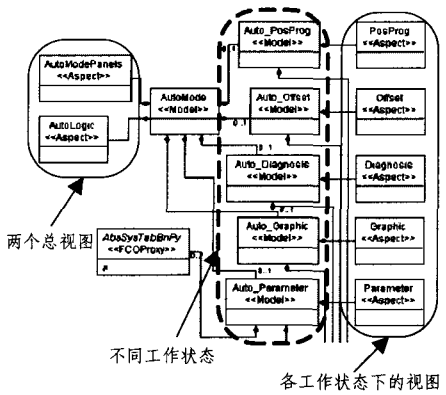

图 7 AutoMode 模式元模型(部分)

描述语言之间的转换。同时我们也可以根据模型转换的抽象水平的转变来进行模型转换类型的划分,可以分为:水平转换和垂直转换。水平转换中,源和目标模型是相同抽象级别的。垂直转换则恰好相反。实际上水平转换和垂直转换与上面提到的内外生转换并不矛盾。两种划分刚好成正交的关系,在一定程度上起着补充的作用。显然,根据以上的定义,本文研究的是将进行的模型层的系统第三方工具验证,属于同一层次不同建模语言描述的模型,故根据上述定义,是一种典型的外生水平转换。

5.3.2 模型转换方法分析

目前支持模型转换的方式有很多,典型的有直接操作转换,第二是利用中间媒介,第三是基于转换语言。其中直接操作转换法指的是程序员使用可以访问和操作模型的 API 对源模型进行转换,从而得到目标模型的方法,由于该方法简单、直接、易用,用户可直接对每个元素进行操作,因而受到用户广泛使用。

GME 提供了基于 C++和 Java 的模型访问接口,这些接口封装了底层 COM 调用的细节,使开发者不再关注底层的实现机制,编程者可以直接利用模型元素的接口[21]对元素的属性进行获取、修改,以及对元素进行更新、删除、添加操作,从而得到目标模型。

在实现该模型转换工具的过程中,本文借鉴了高级语言中编译器的概念,故在此称其为模型解释器。高级语言的编译器的任务是把编程语言的语法和语义映射为能够在特定硬件平台上执行的机器代码。而模型解释器的作用则是把图形化的领域建模语言的语法和语义映射为另一种目标环境所要求的格式。

类似编译器的谓词分析,如果忽略解释器目标语言或模型语言的细节,解释器的主要工作可以抽象为由一个"输入模型树"导出一个"输出模型树"。这里的"输入模型树"对应于输入的模型语言,"输出模型树"则对应于输出的目标语言。其中模型树的定义如下:

模型树是 n ($\geqslant 0$)个模型元素节点的有限集合。在任意一棵非空模型树中,有且仅有一个特定的模型元素节点,称之为模型树根节点(root Element);当 $n>1$ 时,其余节点可分为 $m(m>0)$个互不相交的有限集 $T_1, T_2, \cdots, T_m$,其中每一个集合本身又是一棵树,并且称为根的子树。

这样,一个解释器可以通过以下的步骤完成转换功能。

· 构建输入模型树:把图形化的、有包容关系和继承关系的模型元素,以树状的形式构建出来,图 6 的 XML 构造情况如图 8 所示。

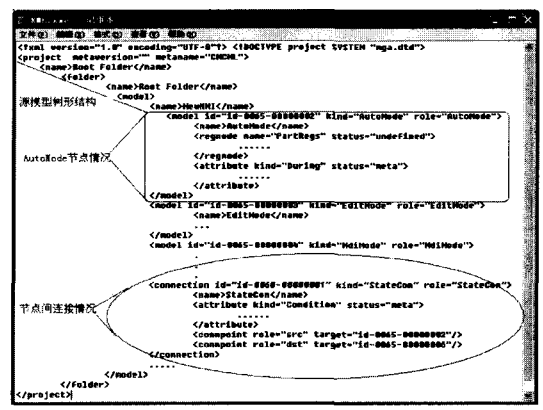

图 8 图 6 用户模型对应的 XML 表示形式(部分)

· 遍历输入模型树:通过访问输入模型树的不同节点,识别节点属性和位置,并把这些信息存储到特定的数据结构中。

• 生成输出模型树：根据输入模型树的遍历结果，判断输入模型树的节点与目标输出模型树的节点之间的映射关系，生成输出树的内容。

上述步骤中的源和目标节点之间的映射关系主要是根据源和目标模型的描述要求和特定来确定，一般是在元模型间进行映射。

5.3.3 模型转换结果

依据上述分析，在实际生成的过程中主要做好对图 6 中 GME 模型树的搜索和生成工作。在生成过程中所依赖的映射关系是 4.2 节中对图 3 元模型的描述和解释。转换后的 StateFlow 图如图 9 所示。只要 Matlab 环境中做一定的设置就可以进行数控车床工作模式的仿真，仿真结果将直接作为系统设计的依据。

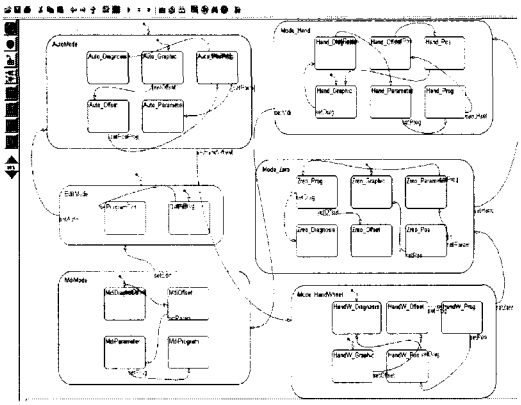

图 9 图 6 对应的转换后的 StateFlow 图形

结束语 本文总结了嵌入式数控系统开发方法的各种不足之处，经综合分析提出采用特定领域模型驱动开发方法来克服传统开发手段中的种种弊端。在此基础上提出嵌入式数控系统模型层验证框架，该框架突出了模型验证及领域建模语言在基于模型驱动开发中的重要地位。并通过基于 StateFlow 数控系统逻辑功能验证实例，描述了在该框架下系统建模语言的构建、模型转换技术的具体实现手段。特定领域模型驱动开发为嵌入式数控系统的开发提升了相应的抽象层次，改进了现有开发质量和设计效率，从方法理论上为嵌入式数控系统提供创新。但目前该方法还有很多值得改进的地方，比如数控系统的各种非功能特性如何用图形化元建模语言来表示、具有特定语义数控领域平台的模型库的建设。另外，目前实现的模型转换是采用文本的方式直接转换，是否可以采用基于图文法的方式，从而提高模型转换的可操作性，这些将是下一步研究的重点内容。

参 考 文 献

[1] 国家中长期科学和技术发展规划纲要. http://www.gov.cn/jrzg/2006-02/09/content_183787.htm

[2] 国务院关于加快振兴装备制造业的若干意见. http://www.gov.cn/jrzg/2006-06/29/content_322286.htm

[3] 广东省工业产业结构调整实施方案(修订版)

[4] Rastofer U, Bellosa F. Component-based software engineering for distributed embedded real-time systems // Software, IEEE Proceedings-[see also Software Engineering, IEEE Proceedings. 2001, 148(3): 99-103

[5] Karsai G, Gray J. Component generation technology for semantic tool integration // Aerospace Conference Proceedings. 2000 IEEE, 2000, 4: 491-499

[6] 林湁，等. 基于实时组件的可重构数控系统研究. 小型微型计算机系统，2004，25(7)：1151-1154

[7] 魏仁选，周祖德，陈幼平，等. 可重用面向对象数控软件及其开发环境研究. 华中理工大学学报，1999，2(3)：19-21

[8] Wang Shige, Shin Kang G. Architecture for Embedded Software Integration with Reusable Components

[9] Wang Shige, Shin Kang G. Constructing Reconfigurable Software for Machine Control Systems. IEEE Transactions on Robotics and Automation, 2002, 18(4)

[10] MDA Guide. Version 1.1.1. Document Number: OMG/2112-16-11. OMG

[11] 高军礼. 基于模型驱动开发方法的开放式结构计算机数控系统的研究[D]. 博士学位论文. 华南理工大学

[12] 杜道山，李从心. 模型驱动在数控系统开发中的应用研究[J]. 青岛大学学报(工程技术版)，2005，20 (3)：53-59

[13] Selic B, Leo M. Using Models in Real-time Software Design [J]. IEEE Control Systems Magazine, 2003, 23(3): 31-42

[14] Martin G, Lavagno L, Louis-Guerin J. Embedded UML: a merger of real-time UML and co-design[C] // Proceedings of the 9th International Symposium on Hardware/software Codesign (CODES01). Copenhagen, 2001: 23-28

[15] Domain-Specific Modeling, 10 times faster than UML. http://www.metacase.com/[EB/OL]

[16] 郑人杰，殷人昆，陶永雷. 实用软件工程[M]. 清华大学出版社，1997

[17] 程国达，彭澄廉. 嵌入式系统描述与验证环境的实现[J]. 计算机辅助设计与图形学学报，2004，16(1)：109-115

[18] UPPAAL. http://www.uppaal.com/[EB/OL]

[19] Sztipanovits J, Karsai G. Model-integrated Computing[J]. IEEE Computer, 1997, 30(4): 110-112

[20] Sendall S, et al. Model transformation: The heart and soul of Model-driven Software Development. IEEE Software, 2003: 42-45

[21] Karsai G, Gray J. Component Generation Technology for Semantic Tool Integration[C]. IEEE Aerospace Conference Proceedings, 2000, 4: 491-499

(上接第 246 页)

nent-based software (accepted) // AICCSA ACS/IEEE International Conference on Computer Systems and Applications. Beirut, Lebanon, June 2001

[8] Wu Ye, Chen Mei-Hwa, Offutt J. UML-based Integration Testing for Component-based Software // Proceedings of 2nd International Conference on COTS-Based Software Systems. Ottawa, Canada, Feb. 2003

[9] Jorgensen PC. 软件测试[M]. 韩柯，等译. 北京：机械工业出版社，2003

[10] 聂长海，徐宝文. 一种最小测试用例集生成方法[J]. 计算机学报，2003，26(12)：12

[11] 景旭，黄东. 移动通信 HLR 软件子系统的设计与研究[D]. 南京：东南大学，2006