一种基于CBSE的嵌入式实时软件建模方法研究*)

夏 苑[1,2] 张为群[1,2]
(西南师范大学计算机与信息科学学院 重庆 400715)[1]
(重庆市智能软件与软件工程重点实验室 重庆 400715)[2]

摘 要 基于构件的软件工程(CBSE)是众多嵌入式软件开发方法中的一种崭新方法。本文提出一种RT-UML*/LTLC的双语言框架来对基于CBSE的嵌入式实时软件的建模,既能充分地刻画出嵌入式软件实时,资源严格有限等特点,又能较好地体现出构件的特征,能更容易地过渡到基于构件的嵌入式软件开发方法的后继阶段。

关键词 嵌入式实时软件,CBSE,建模,RT-UML*,LTLC

Research for Modeling of the Component-Based Software Engineering for Real-Time Embedded Software

XIA Yuan[1,2] ZHANG Wei-Qun[1,2]
(College of Computer and Information Science, Southwest China Normal University, Chongqing 400715)[1]
(Chongqing Intelligent Software and Softwre Engineering Laboratory, Chongqing 400715)[2]

Abstract CBSE(Component-Based Software Engineering) is a new method of the many development of the embedded software. The paper gives a modelling approach of embedded real-time software, using RT-UML* and LTLC. The model describes adequately embedded software properties such as real time and resource rigorous limitation, and is good at take the component feature, accordingly can transfer easily to subsequent phase of the development method of embedded real-time software.

Keywords Embedded real-time software, CBSE, Model, RT-UML*, LTLC

1 引言

嵌入式系统(Embedded System),是指以应用为中心,以计算机技术为基础,软硬件可裁减,适应应用环境的专用计算机系统,一般具有实时性、专用性和资源严格有限性等特点。为了突出大多数嵌入式系统的"实时"反应的特性,又称其为嵌入式实时系统(Real-Time Embedded system, RTE)。

嵌入式软件方法学是把一般软件设计方法学延伸到嵌入式中的结果。嵌入式软件方法学走过了从结构化方法,面向对象方法(OOSE),基于构件方法(CBSE)这三大阶段,经历了单任务系统到多任务系统的转变。

2 基于构件的嵌入式软件开发方法

基于构件开发(CBD)或基于构件的软件工程(CBSE)是一种软件开发新范型。用基于构件的软件方法学来构造嵌入式软件是一种新的尝试,它能将基于构件的软件开发的诸多优点(如:改进软件质量,缩短上市时间,解决日益增长的软件复杂性)带入嵌入式软件开发中。

特别是对于嵌入式实时软件,涉及到大量和时间相关的算法,如果把它们构件化,并按照特定的模式和框架在软件开发中重用它们,其意义不言而喻。而且,基于构件进行软件开发,还有利于软件开发的自动化。

基于构件的嵌入式实时软件可分为领域工程和应用工程两个方面。本文立足于应用工程。应用工程解决的核心问题是如何利用领域工程的构件和构架,进行应用程序的生成,这个过程称为基于构件的嵌入式软件开发过程。它可以分为以下几步:系统建模、构件抽取、构件组装、构件的适应性修改、集成测试与系统发布。其中,基于构件的嵌入式实时软件的建模方法是本文的研究内容。

3 基于CBSE的嵌入式实时软件的建模方法

模型为项目的参与者提供了统一的语言,方便了项目参与者在系统的某些方面的交流。可能不需要整个系统的模型集,只需关注建模中系统最复杂和最关键的部分,因为这部分最容易产生模糊性和不确定性。

3.1 RT-UML*

统一建模语言UML(Unified Modeling Language)已成为建模的标准语言,能对软件密集型系统的制品进行可视化、详述、构造和文档化。RT-UML(实时UML)是在基本UML基础上引入了起源于ROOM(real-time object oriented modeling)的便于设计复杂嵌入式系统的结构,主要通过交互图和状态图对时间限制进行建模。

本文对RT-UML进行改进(RT-UML*),来对基于CBSE的嵌入式实时软件进行建模。其改进之处主要在以下几个方面:

1. 给类的操作增加两个修饰符:Period和deadline,表明操作是以时间驱动的。Period:表明此操作能以一个周期自动地执行,周期等于操作参数列表的第一个参数P的值(图1(a));deadline:表明此操作是到达期限后自动执行,期限等于操作列表参数的第一个参数D的值(图1(b))。

*)本论文为软件非功能性需求的多模型测试与外部度量(重庆市科委),软件过程度量与质量度量综合平台建设(重庆市信息产业发展基金)联合资助下的成果。**夏 苑** 硕士,主要研究领域为嵌入式软件,形式化软件工程,软件建模;**张为群** 教授,主要研究领域为形式化软件工程,软件测试。

这样改进后的类具有了时间的特征。

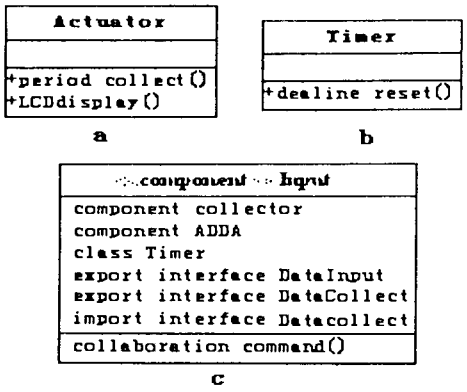


图 1

2. 对 UML 的元模型进行扩展，由构造型引入一个“逻辑构件”的元类。“逻辑构件”允许嵌套分层(图 1(c))。

①名称是“逻辑构件”的名称；

②类名是存在于构件中的一个或多个类的名称，用修饰符 class 表示；

③构件名是嵌套中的构件的名称，用修饰符 component 表示；

④接口名是和此构件相联系的接口的名称，其中，用修饰符 export interface 表示该构件实现的接口，用修饰符 import interface 表示该构件使用的接口；

⑤协作是这些类或构件之间的交互，用修饰符 collaboration 表示。

“逻辑构件”与 UML 中本来存在的构件不同。UML 中原来存在的构件表示的是存在于比特世界中的物理抽象，它不存在于概念世界中。引入“逻辑构件”后，能更好地从抽象级别上对构件进行建模，能更清晰，准确地表现构件中类及类之间的关系以及构件之间的层次关系。

3. RT-UML 中的状态图能较好地描述一般的反应式嵌入系统，但对实时嵌入式系统来说，它在对时间的描述上仍有着很大的不足。

本文对 UML 的状态图进行改进，把时间因子加入进状态中，使之能更好地描述时间限制问题(图 3)。

我们用六元组$(N,\sum,n_0,\delta,Inv,g)$来表示状态图，其中：$N$ 为节点的有限集；$\sum$为动作集，其中元素用 $\alpha,\beta\cdots$来表示；n_0 ($n_0\in N$)为初始结点；δ 为转换函数；Inv(invariant)：$N\rightarrow Constr$ 给出结点上的时间不变式；g 为监护条件 *guard* 的集合。

我们以一个结点 n 和时钟变量的一个计值 $\rho(\rho>0)$，得到一个二元组$<n,\rho>$ 为系统的一个状态，状态图的行为通过状态转换得到体现。

其中，状态转换可以分为两种，一种为执行一条边上的转移动作后转换至下一状态，称为跳跃转换；另一种为选择留在本结点进行等待，让所有时钟变量的值在该结点按相同速率增加，称为延迟转换，其状态变换只体现在 ρ 的变化上，但延时的长度要受到制约，即延时后的时钟变量 ρ 值必须满足该结点上的时间不变式 Inv。

3.2 用 RT-UML* 进行建模

我们把改进后的 RT-UML 称为 RT-UML*，用它来对基于构件的嵌入式实时系统进行建模。在模型中我们把构件作为基本元素，就像面向对象建模时 UML 中的类一样。

1)构件图 捕获系统的静态视图。构件图由构件的层次关系组成。最高一层“逻辑构件”就是整个系统，最低一层“逻辑构件”是叶子构件，它由一些类，接口及它们之间的协作构成。

接口是一个用来描述一个构件所提供的服务的操作集合。构件的服务一般只能通过其接口来访问，几乎所有流行的基于构件的标准(如 COM+，CORBA 和 JavaBeans)都以接口作为把构件绑定在一起的粘合剂。接口可以用构造型或者棒棒糖型来表示。

图 1(c)表示的构件可以展示为一个构件图，如图 2 所示。

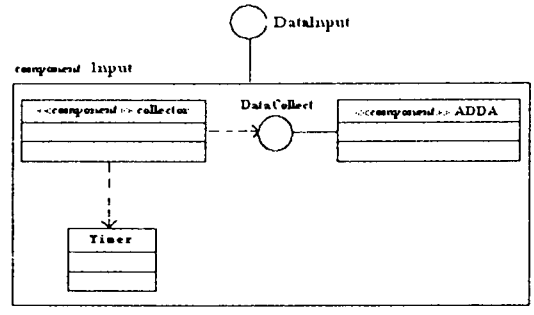


图 2

2)交互图 捕获系统的动态视图。交互图包括顺序图和协作图，由于嵌入式系统的时间特性，这里我们主要使用更能体现时间特性的顺序图来对嵌入式实时软件进行建模。建模的基本元素仍然是构件。当然，对叶子构件而言，组成它的类之间也可以用顺序图来表示它们之间的协作。

3)实施图 显示系统的设备和处理器视图。在进行建模的时候，我们用实施图描述出系统的资源情况(包括 CPU，内存等)。图 3 为铁路道口控制系统的实施图。

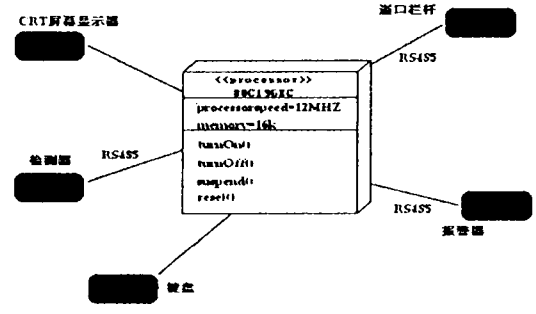


图 3

4)状态图 捕获系统的动态行为视图，主要是在时间限制建模方面。

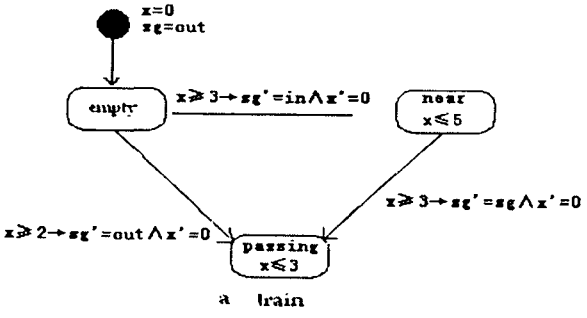
嵌入式实时系统的时间限制主要反应在：① 外部刺激到达系统的时间限制；② 系统响应时间的限制。

下面以铁路道口控制系统来说明如何用 RT-UML* 的状态图来进行时间限制的建模。

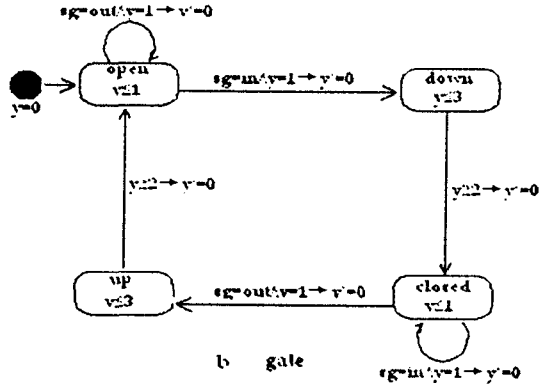
由探测器检测火车的临近。当探测器检测到火车经过时，发出通知 near，火车发出进站信号‘in’，在 near 区段火车运行 3～5 分钟便进入 passing 区段。在 passing 区段火车运行 2～3 分钟离开道口，发出出站信号‘out’。前后两次探测器检测到火车临近的时间至少要 3 分钟。道口起初是打开的(open 状态)，并每隔一分钟检查一次看是否有火车进站的信

号，若无则继续等待，若有则开始下降道口上面的栏杆(这一阶段道口处于 down 状态)，道口彻底关闭后处于 closed 状态。然后道口便开始每隔一分钟检查一次看火车是否仍在车站内，若是则继续等待，否则便开始升起栏杆，恢复到 open 状态并开始等待下一个'in'信号。门的关闭和打开均需要 2～3 分钟。

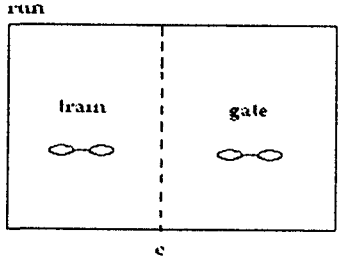
图 4(a)和图 4(b)表示的是火车和道口的状态图，其中 x 和 y 是时钟变量，用来表示时间约束，布尔型变量 sg 用来表示信号'in'和'out'，sg=in 表示火车在站内，sg=out 表示火车在站外。



a train



b gate



c

图 4

图 4(c)表示状态 *train* 和 *gate* 为并发状态，并且都为状态 run 的子状态，这时图 4(a)和图 4(b)就成为两个子机。

RT-UML* 的状态图既继承了 RT-UML 状态图的嵌套分层(子机)、并发等优点，又引入了时间因子的概念及表示方法，能较好地描述实时系统的时间限制。

可以看出，RT-UML* 既能很好地体现出构件的特征，又能很好地体现出实时的特点，能较好地对基于构件的嵌入式实时软件建模。

4 带有时钟变量的线性时序逻辑 LTLC

嵌入式实时系统经常是一些高安全关键性系统，例如，军事控制、航天航空、医疗设备等等。在对这些系统进行建模时，对系统性质的描述和证明就显得极为重要。我们用 RT-UML* 的状态图作为描述嵌入式实时系统动态行为的主要手段(特别是时间限制方面)，但它属于一种操作性规约语言，不便于性质的描述。于是本文将利用 LTLC 来描述嵌入式实时系统的性质(见文[5])。

LTLC(linear temporal logic with clocks)是由李广元、唐稚松等提出的，它是线性时序逻辑在实时情况下的一个推广。LTLC 既能作为规范语言用来表示实时系统的性质，也能作为系统描述语言来表示实时系统的实现模型。LTLC 的这个特点将有助于实时系统的性质验证和实时系统的逐步求精。

关于 LTLC 的语法和语义请参见文[5]，这里不再叙述。

4.1 嵌入式实时系统的 LTLC 描述

实时嵌入式系统是一种带有时间约束的计算系统，它的许多动作的完成与时间相关。

我们以时间模块(timed modules)来模拟实时系统。每个时间模块主要由有限个转换(transition)组成。控制变量的值通过转换的作用来改变。每个时间模块有两种类型的转换：跳跃转换和延迟转换。

跳跃转换通常被写成形如：'vertex∧guard→new_vertex∧assignment'的卫士命令。

延迟转换通常被写成如下形式：

'vertex→invariant'

一般而言，一个时间模块具有如下的形式：

```
module      module_name
external    {variable_name:type}*
controlled  {variable_name:type}*
init        init_cond
jump        {vertex∧guard→new_vertex∧assignment}*
delay       {vertex→invariant}*
```

其中，type 表示变量的类型，可以是 boolean(布尔型)和 clock(时钟型)；init_cond 是模块的初始条件，它是一个 LTLC/R 当前的状态公式，用于表示模块在初始时刻 t=0 时其控制变量所应满足的条件。

下面通过铁路道口控制系统来说明怎么用 LTLC/R 模拟嵌入式实时系统。

图 3 的模型可以用 LTLC/R 表示如下的两个时间模块，其中的布尔型变量 p 表示图 4(a) *train* 的状态，布尔型变量 q 表示图 4(b) *gate* 的状态。

```
module      Train
controlled  p:{empty,near,passing};
            sg:{in,out}
            x:clock
init        p=empty∧sg=out∧x=0
jump
p=empty∧x≥3→(p'=near∧sg'=in∧x'=0);
p=near∧x≥3→(p'=passing∧sg'=sg∧x'=0);
p=passing∧x≥2→(p'=empty∧sg'=out∧x'=0);
delay       p=empty→true;
            p=near→x≤5;
            p=passing→x≤3;
```

```
module      Gate
external    sg:{in,out}
controlled
q:{open,closed,up,down};
            y:clock
init        q=open∧y=0
jump
q=open∧sg=out∧y=1→(q'=q∧y'=0);
q=open∧sg=in∧y=1→(q'=down∧y'=0);
q=down∧y≥2→(q'=closed∧y'=0);
q=closed∧sg=in∧y=1→(q'=q∧y'=0);
q=closed∧sg=out∧y=1→(q'=up∧y'=0);
q=up∧y≥2→(q'=open∧y'=0);
delay       q=open→y≤1;
            q=down→y≤3;
            q=closed→y≤1;
            q=up→y≤3;
```

注：以上转换中出现的符号→并不是 LTLC 中的逻辑联

(下转封四)

(上接第 215 页)

结词,所以上面给出的时间模块现在还不是 LTLC 中的逻辑公式,下面给出时间模块对应的 LTLC 公式,并利用 LTLC 给出时间模块的语义。

对于跳跃转换 α: vertex ∧ guard→new_vertex ∧ assignment,称 LTLC 公式 vertex ∧ guard ∧ new_vertex ∧ assignment 为转换 α 所对应的时序逻辑公式,记作 TLF(α)。

对于延迟转换 β: vertex→invariant,称 LTLC 公式 vertex ∧ invariant 为延迟转换 β 所对应的时序逻辑公式,记作 TLF(β)。

定义 1 设 M 是一个时间模块,$\alpha_0,\alpha_1,\cdots,\alpha_{n-1}$ 是 M 的所有跳跃转换,$\beta_0,\beta_1,\cdots,\beta_{l-1}$ 是 M 的所有延迟转换。称 LTLC 公式 $init\_cond \wedge (o(V_c=V_{c'} \vee \forall_{j<n} TLF(\alpha_j))) \wedge o\forall_{k<l} TLF(\beta_k)$ 为时间模块 M 所对应的时序逻辑公式。

其中,Vc 是 M 中所有控制变量的集合。以 $TLF(M)$ 来表示这个对应与模块 M 的时序逻辑公式。

时间模块 Train 和 Gate 所对应的时序逻辑公式:

TLF(Train)=((p=0∧sg=0∧x=0)∧°((p=0∧x≥3∧p′=1∧sg′=1∧x′=0)∨(p=1∧x≥3∧p′=2∧sg′=sg∧x′=0)∨(p=2∧x≥2∧p′=0∧sg′=0∧x′=0)∨(p′=p∧sg′=sg∧x′=x))∧°((p=0∨(p=1∧x≤5)∨(p=2∧x≤3))。

TLF(Gate)=((q=0∧y=0)∧°((q=0∧sg=0∧y=1∧q′=q∧y′=0)∨(q=0∧sg=1∧y=1∧q′=1∧y′=0)∨(q=1∧y≥2∧q′=2∧y′=0)∨(q=2∧sg=1∧y=1∧q′=q∧y′=0)∨(q=2∧sg=0∧y=1∧q′=3∧y′=0)∨(q=3∧y≥2∧q′=0∧y′=0)∨(q′=q∧y′=y))∧°((q=0∧y≤1)∨(q=1∧y≤3)∨(p=2∧y≤1)∨(q=3∧y≤3))。

将时间模块等同于它所对应的时序逻辑公式 $TLF(M)$,并将 $TLF(M)$ 的语义模型作为 M 的语义模型。

4.2 系统的性质描述

定理 1 设 M 是一个时间模块,φ 是一个 LTLC 公式。如果 $TLF(M)\models\varphi$,则 φ 是 M 的一个性质。

定理 2 对于公式 φ,若 $TLF(M1)\wedge TLF(M2)\wedge\cdots\wedge TLF(Mn)\models\varphi$,则 φ 是复合模块[$M1\parallel M2\parallel\cdots\parallel Mn$]的一个性质。

由定理 1 和定理 2 可知,如果 φ 是 $TLF(M1)\wedge TLF(M2)\wedge\cdots\wedge TLF(Mn)$ 的逻辑结论,则复合模块具有性质 φ。

把铁路道口控制系统看作是两个模块 Train 和 Gate 组成的复合模块,由这个复合模块逻辑推出的公式就是此实时系统的性质。例如,如果公式°($p=2\rightarrow q=2$)(也就是 p=passing→q=close,它表示当火车处在 passing 状态的时,道口一定是关闭的)能由复合模块[Train ‖ Gate]逻辑推出,那么°($p=2\rightarrow q=2$)就是该实时系统的性质,这是一个安全性公式。

由此,我们可以用 LTLC/R 来描述嵌入式实时系统的性质。

结论 本文针对嵌入式实时软件的特点,并结合构件的特点,用 RT-UML* 和 LTLC 对基于 CBSE 的嵌入式实时系统建立分析模型,实际上是建立了 RT-UML*/LTLC 的双语言框架模型。这个模型还需要细化,进而抽取出构件,最终进行系统的构件组装集成。

随着构件技术以及嵌入式硬件技术的迅速发展,用嵌入式构件来进行嵌入式实时软件的开发将会成为一种主流的开发方法。

参 考 文 献

1 Murali S. Compositional performance reasoning. In: Harris CC, ed. Proc. of the ICSE CBSE4. IEEE, 2001. 98~101
2 Reed R. Methodology for real time systems. Computer Networks and ISDN Systems, 1996, 28: 1685~1701
3 Douglass B P. Real-Time UML (2nd Edition). Addison-Wesley, 1999
4 Pressman R S. Software Engineering: A Practitioner's Approach, (Fifth Edition). China Machine Press, 2002
5 李广元,唐稚松. 反应系统的连续时序逻辑表示和验证. 计算机学报,2003,26(11)
6 周彦晖,张为群. 软件形式化与可视化软件模型的转换. 计算机科学,2003,30(7)
7 周彦晖,张为群. 形式化与可视化结合的 FDOOM 的软件开发方法. 计算机科学,2003,30(9)
8 舒风笛,毋国庆,李明树. 面向嵌入式实时软件的需求规约语言及检测方法. 软件学报,2004,15(11)
9 熊光泽,古幼鹏,桑楠. 嵌入式应用软件设计方法学研究综述. 计算机应用,2004,24(4)
10 杨芙清. 软件工程技术发展思索. 软件学报,2005,16(1)

计算机科学

(1974 年 1 月创刊)
第 32 卷第 8 期 (月刊)
2005 年 8 月 25 日出版

ISSN 1002-137X
CN50-1075/TP

定价: 25.00 元 国外定价: 5 美元
邮发代号: 78-68
发行范围: 国内外公开

主管单位: 国家科学技术部
主办单位: 国家科技部西南信息中心
编辑出版:《计算机科学》杂志社
重庆市渝中区胜利路 132 号 邮政编码: 400013
电话: (023) 63500828 E-mail: jsjkx@swic.ac.cn
网址: www.jsjkx.com
社 长: 牟炳林
主 编: 彭 丹
副 主 编: 朱宗元
主编助理: 徐书令
印 刷 者: 重庆科情印务有限公司
总发行处: 重庆市邮政局
订 购 处: 全国各地邮政局
国外总发行: 中国国际图书贸易总公司 (北京 399 信箱)
国外代号: 6210-MO