基于环境的多形态时间需求建模方法

陈小红[1),2),3)] 刘 静[1)]

[1)](华东师范大学上海市高可信计算重点实验室 上海 200062)

[2)](北京大学高可信软件技术教育部重点实验室 北京 100871)

[3)](浙江师范大学计算机软件与理论省级重中之重学科 浙江 金华 321004)

摘 要 时间需求作为嵌入式系统的关键要素,其重要性越来越突出.但是目前的时间需求存在着多样的环境时间描述与单一的软件时间描述融合问题.文中在基于环境的功能性需求的基础上,提出基于环境的多形态时间需求建模方法,试图在需求层次上为该问题提供解决方案.文中构建了支持这个方法的多形态时间需求模型,在环境因素的基础上增加了多形态时间描述,并提出该需求模型制导的需求建模过程,帮助需求分析员在功能性需求的基础上建模时间需求.文中还给出了从多样的环境时间描述得到软件时间需求规约的步骤,指导需求分析人员抽取时间需求规约.

关键词 时间需求;多形态时间;基于环境建模;嵌入式系统

中图法分类号 TP311 **DOI 号** 10.3724/SP.J.1016.2013.00088

Modeling Software Timing Requirements: An Environment Based Approach

CHEN Xiao-Hong[1),2),3)] LIU Jing[1)]

[1)](*Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai* 200062)

[2)](*Key Laboratory of High Confidence Software Technologies (Peking University) of Ministry of Education, Beijing* 100871)

[3)](*Top Key Discipline of Computer Software and Theory in Zhejiang Provincial Colleges, Zhejiang Normal University, Jinhua, Zhejiang* 321004)

Abstract Timing requirement as a key element in the embedded systems is becoming more and more important. However, at present, there is an integration problem between multiple environment time description and single software time description in timing requirements. On the basis of the environment based functional requirements, this paper proposes a multiform time requirements modeling approach trying to provide a solution to the integration problem at the requirement level. A multiform time requirement model is built by adding multiform time description to environment elements for supporting this approach. A timing requirements modeling process guided by the requirement model is also presented for helping timing requirements description on the basis of functional requirements. Steps from timing requirements to timing requirement specification are also given. Following these steps, requirements analysts can obtain the timing requirements specification easily.

Keywords timing requirements; multiform time; environment based modeling; embedded systems

---

收稿日期:2012-07-01;最终修改稿收到日期:2012-11-27.本课题得到国家自然科学青年基金(61202104,60903021)、国家自然科学重点基金(61170084)、国家"九七三"重点基础研究发展规划项目基金(2009CB320702)、国家"八六三"高技术研究发展计划项目基金(2011AA010101)、教育部博士点基金(20120076120016)、创新群体(61021004)以及浙江师范大学计算机软件与理论省级重中之重学科开放基金(ZSDZZZZXK30)资助.**陈小红**,女,1982 年生,博士,讲师,主要研究方向为需求工程和软件工程.E-mail:xhchen@sei.ecnu.edu.cn.

刘 静,女,1965 年生,博士,教授,博士生导师,主要研究领域为模型驱动软件开发和软件工程.

1 引　言

嵌入式系统是近年来的研究热点，新近提出的物联网[1]和信息物理融合系统(Cyber-Physical Systems，CPS)[2]都建立在其基础之上. 在嵌入式系统中，实时是关键要素之一. 实时需求的研究工作主要集中在实时系统中，方法主要采用时态逻辑和时态逻辑的扩展[3-4]对时间点和时间段做定性与定量的描述与分析. 现在的需求工程方法，比如面向目标的方法[5]和面向主体和意图的方法[6]，也采用时态逻辑或其扩展对时间点和时间段进行描述. 我们以前的工作也针对嵌入式系统，从环境建模的角度对时间点和时间段进行刻画[7].

而物联网和 CPS 等的提出，对时间需求又提出了新的挑战. 以 CPS 为例，CPS 是一类在环境感知的基础上，深度融合了计算、通信和控制能力的可控、可信、可扩展的网络化物理设备系统[8]. 简单来说，CPS 系统就是开放的嵌入式系统加上网络和控制功能，其面临的环境多样，包括自然环境、建筑、机器，当然也包括人类自身. 这样的环境随时间连续变化，在时间描述上可能多样，比如本文案例中使用汽车曲轴转过的角度描述时间. 而对 CPS 软件来说，其时间描述单一，因此就存在着多样的环境时间描述与单一的 CPS 软件时间描述之间的融合问题.

多样的时间描述问题也出现在了软件工程其它领域. 在同步语言[9]中，为了解决这个问题它引入了多形态时间. 时间单位不仅仅限于物理时间的秒、微秒等. 在同步程序中，可以出现类似“这个任务必须在 10ms 之内完成”的语句，也可以出现“这个车必须在 50 m 之内停住”的语句. 10 ms 和 50 m 都是表示时间的截止，这就是多形态时间. 法国国家信息与自动化研究所 INRIA 的 Andre 等人将多形态时间引入到系统建模中去，为实时嵌入式系统构建了规范 MARTE(Modeling and Analysis for Real-Time Embedded System)[10]，将时间定义为一组时钟，每个时钟都有自己的时间单位. 在此基础上，他们还创建了时钟约束语言 CCSL(Clock Constraint Specification Language)[11]，用于实现时间的同步.

本文将多形态时间的概念引入到需求建模中来，尝试在需求层次上解决多样的环境时间描述与单一的软件时间描述的融合问题. 我们以前的工作提出了基于环境建模的功能性需求建模方法[12]. 该工作将环境看作是一组与软件交互的实体(又称问题领域)，将需求看作是预期问题领域的变化，即期望的环境变化. 我们认为，问题领域是现实世界的一部分，有自己的生命. 从直觉上讲，每个问题领域的时间都可以建模成(或者参考)一个时钟，该时钟可以有自己的时间单位. 因此，结合我们前期的工作[12]和 MARTE/CCSL[10]，本文提出构建基于环境的多形态时间需求建模方法，跟其它需求工程方法相比，其主要特点有：(1) 它是基于环境的建模，建立在现实世界问题领域的基础上，有着坚实的现实基础，也建立在功能性需求之上；(2) 它支持多形态时间建模，适合描述多样的环境时间和单一的软件时间.

为了支持基于环境的多形态时间需求建模方法，本文首先构建了基于环境的多形态时间需求模型，在环境因素的基础上增加了多形态时间描述，由此在基于环境变化描述的功能性需求的基础上描述多样的软件环境时间约束和软件时间约束，包括最基本的时间点和时间段的描述，并提出该需求模型制导下的建模过程，以指导需求分析人员使用该需求模型进行时间需求建模. 最后采用该方法对汽车领域进行了应用，验证了本文方法的有效性.

本文第 2 节给出多形态时间需求模型；第 3 节给出该时间需求模型制导下的时间需求建模过程；第 4 节采用汽车领域的具体案例来验证方法的可行性；最后，第 5、6 节讨论相关工作并总结全文.

2 多形态时间需求模型

在我们以前的功能性需求模型的基础上[12]，本文进一步将跟功能相关的概念与多形态时间的概念关联起来，扩展该需求模型，得到多形态时间需求的概念模型，如图 1 所示.

图 1 白色矩形框中的概念及其关联来自我们以前的工作，总结了问题框架方法[13]的概念和关联，表示了基于环境软件问题的基本概念及关联：

(1) 软件问题(Problem)处于一组现实世界的问题领域(Problem Domain)当中，其目的是开发一个软件机器(Machine)，以满足客户的需求(Requirement)，其中软件机器的环境就表示为一组问题领域的集合，需求就表示为问题领域的预期变化.

(2) 在软件机器和问题领域存在共享的现象(Phenomenon)，这就是在机器和问题领域之间的交互(Interaction). 需求通过软件机器与环境的交互实现.

(3) 每个交互有一个发起方，一个接收方，发起方和接收方必须一方是机器，另一方是问题领域. 每个交互必须还有内容，那就是现象，现象可以分为事

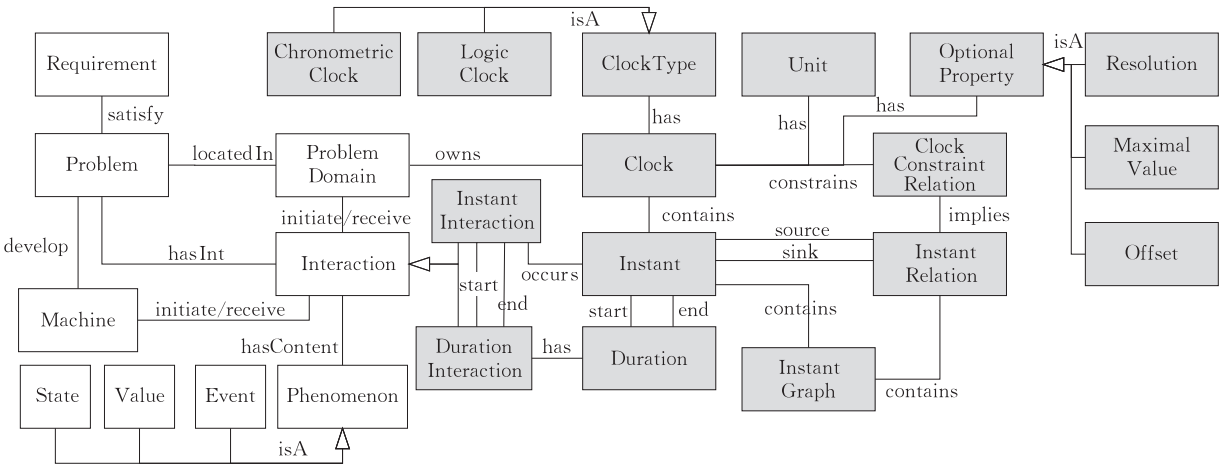


图 1 多形态时钟需求概念模型

件(Event)、状态(State)和值(Value).

由上述描述可以看出,基于环境的功能性需求模型的核心概念是问题领域和交互,它们都有环境因素信息.围绕这些环境因素,进行与时间相关的扩展,得到图 1 灰色矩形框中表示的概念.这些概念大部分来自 MARTE,根据需求阶段的特征,又重新进行了定义.

首先考虑扩展交互.原来工作假设交互瞬间发生,不考虑持续时间.但在实际情况中,有些交互的发生需要花费一定时间.根据其持续与否,将交互分为瞬时交互(Instant Interaction)和持续交互(Duration Interaction)两种.持续交互也可以用两个瞬时交互表示,一个表示持续交互的开始,一个表示持续交互的结束.每个持续交互都有持续时间,这个持续时间可以由开始瞬时交互和结束瞬时交互之间的时间差(Duration)来表示.

与 MARTE 类似,我们也将时间(Time)看作时钟(Clock)的集合.这些时钟本质上是连续的(dense),因为在现实世界中,每种物品都有连续的生命周期.一个时钟可以是精密时钟(Chronometric Clock),也可以是逻辑时钟(Logic Clock),每个时钟有自己的时钟类型(ClockType)、时间单位(Unit)和一组可选性质(Optional Property).不同类型的时钟,时间单位选择不同.精密时钟的时间单位是常见单位,比如秒或者其它衍生的单位.大多数逻辑时钟使用一个普遍的时间单位 tick,也可以使用像处理周期这样的单位,或者使用物理数量,像本文案例中使用的角度.时钟的可选性质包括分辨率(Resolution)、初始值(Offset)和最大值(Maximal Value).

每个时钟都包含一组有序的时间点(Instant).时间点和时间点之间存在着各种时间点关系(Instant Relation).时间点和时间点之间的时间差距称为时间差或时间段(Duration).为了显式地表示时间点及其关系,定义了时间点图(Instant Graph).不同的时钟由于其时间点之间存在的关系,使得它们之间也存在着约束关系,这种关系我们称时钟约束关系(Clock Constraint Relation).

这两组概念(图 1 中的白色和灰色矩形框)通过如下关系关联在一起:每个问题领域都有(或者参考)某个时钟;每个问题领域的每个交互(特指瞬时交互,若为持续交互就转换成瞬时交互)发生的时刻都是该时钟的一个时间点.

表 1 给出了上述概念的具体含义及其层次.

表 1 需求模型中概念的层次结构及含义

概念	定义
问题	要由软件开发完成的任务
机器	在软件开发中要开发的系统
问题领域	与软件系统交互的现实世界实体
需求	需要满足的功能
交互	在机器和问题领域之间共享的可观察现象
瞬时交互	交互的发生瞬间完成
持续交互	交互的发生需要花费一定时间
现象	一个现象可能是状态、事件或值
事件	在特定时刻发生、出现的个体
状态	变量和值之间的关系,可以随时间改变
值	无形的个体,是不会改变
时钟类型	时钟的种类
精密时钟	指向物理时间的时钟
逻辑时钟	关注时间点之间的顺序,而忽视时间点之间的物理时间间隔的时钟
时钟	用于记录、保持时间的模型
时间单位	测量时间所用的基本单位
时间点	特定的时刻
时间段	时间区间
时间点关系	时间点之间的关系
时钟约束关系	时钟之间的关系
时间点图	显式表示时间点及时间点关系的图
可选特性	可能具有的能够刻画时钟的特性
分辨率	时钟的粒度
最大值	时钟时间点能达到的最大值
初始值	时钟的开始值

基于上述的概念模型，本文给出一些关键概念的形式化定义，包括时钟、时间点图、时间差、一些时钟约束关系和交互等. 由于时钟的多形态特性，本文在 MARTE 时钟模型的五元组定义基础上，增加了可选属性来描述多形态时钟，得到时钟定义，如定义 1 所示.

定义 1.　时钟(Clock).

$$\text{Clock} := \langle I, <, D, \lambda, U, O \rangle,$$

其中，

$I=\{instant_1, instant_2, \cdots, instant_n\}$ 是时间点 $instant_i(1\leq i\leq n)$ 的集合. 这些时间点都是时钟的里程碑，它们记录了在其生命周期中有意义的时刻. 这些时间点有两种记录方式，一种是用交互发生的时刻来记录($@t_{\text{interaction}}$)，另一种是其时钟变量的赋值.

$<$是定义在 I 上的类序关系，命名为严格先于.

$D=\{d_1, d_2, \cdots, d_n\}$ 是标签的集合，D 中的每一个元素 $d_i(1\leq i\leq n)$ 表示一个有意义的交互.

$\lambda: I\rightarrow D$ 是一个标签函数.

U 是时间单位符号.

O 是一组可选属性的集合，包括分辨率、最大值和初始值.

时间点图是直观地表达时钟内或时钟之间时间点及时间点之间关系的图. 该图可以形式化地定义为一个 2 元组，如定义 2 所示.

定义 2.　时间点图(Instant Graph).

$$\text{InstantGraph} := \langle Ins, Rels \rangle,$$

其中，

$Ins=\{instant_1, instant_2, \cdots, instant_n\}$ 是一组时间点的集合.

$Rels\subseteq Ins\times Ins$ 是 Ins 集合中时间点之间的二元关系集合. 二元关系可以分为 3 种[10]：先于(Precedence，$\leq$)、同时发生(Coincident，$\equiv$)和严格先于(Strict Precedence，$<$)，其中先于关系包含同时发生的情况，而严格先于关系不允许同时发生. 这 3 种关系的图形表示如图 2 所示.

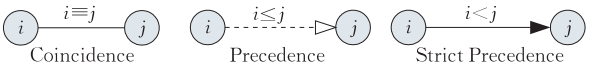


图 2　时间点关系示意图

时钟约束关系可以靠时间点之间的关系来定义. 文献[11]给出了很多种关系，下面介绍本文用到的过滤(filterBy)关系，它描述了两个时钟时间点之间的“过滤”对应关系，即一个时钟的时间点都是依靠另一个时钟的时间点通过某种形式的过滤得到. 其具体定义见定义 3.

定义 3.　过滤(filteredBy).

$$A=B \text{ filteredBy}(\omega),$$

其中 A 和 B 都是时钟，ω 是一个二进制数. A 的时间点由 B 的时间点通过 ω 过滤器构造出来.

关于时间点与时间点之间的时间差，与 MARTE 类似，本文用时差算子(Duration operator，—)来描述. 与原定义中的时间单位秒(s)不同，本文采用多形态时间单位. 其具体定义见定义 4.

定义 4.　时差算子(Duration operator，—).

$$a-b<t \text{ unit} <=> b<a<b+t \text{ unit},$$

其中，a 和 b 都是时间点，“+”是延迟算子，unit 为任意时间单位.

交互作为一个非常重要的概念，从它出发的关联有 3 个，因此，我们将交互定义为一个四元组(其中一元为交互名称)，其具体定义见定义 5.

定义 5.　交互(Interaction).

$$\text{Interaction} := \langle Int, Sender, Receiver, Content \rangle,$$

其中，

Int 是交互的名称.

$Sender$ 是发起方.

$Receiver$ 是接收方.

$Content$ 是发送方和接收方共享的现象.

3　多形态时间需求建模过程

遵循图 1 给出的多形态时间需求模型，我们定义了一个多形态时间需求建模过程，如图 3 所示. 它包含 4 大步骤，即为每个问题领域进行时间建模，确定时钟的约束关系，构建时间点之间的定量关系和抽取时间需求规约. 下面介绍具体步骤.

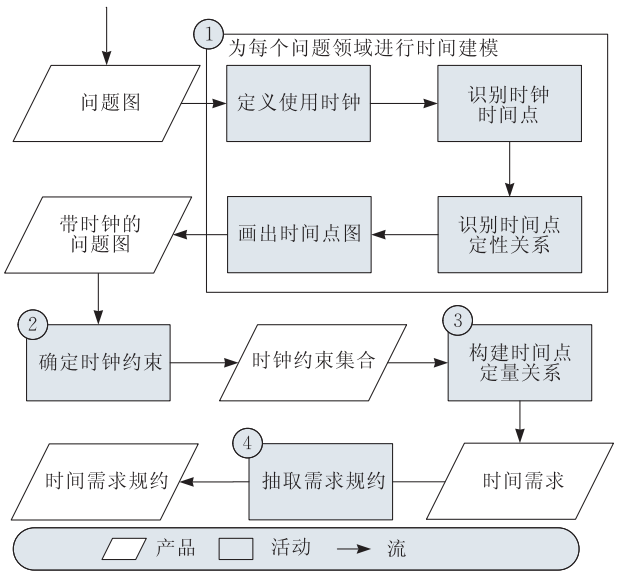


图 3　时间需求建模过程

3.1 问题领域的时间建模

本步骤的输入是问题图[13],包含所有的问题领域和交互,输出为带有时钟的问题图.根据时钟的定义,我们设计如下4个子步骤来完成:

(1)定义使用时钟

这一步首先检查是否有现成的时钟可用,若无,考虑从如下几个方面定义新的时钟:时钟类型(是精密时钟还是逻辑时钟)、时间单位和可选特性(包括分辨率、最大值、初始值)等.

对于问题图中的任一问题领域,其任一连续属性都可以考虑作为一个逻辑时钟,比如本文案例中使用曲轴角度作为一个时钟,其角度的任一取值都是该时钟的一个时间点.

(2)识别时钟时间点集合

由时钟的定义知,上一步定义的时钟,其能识别的时间点是与这个问题领域相关的瞬时交互发生的时刻,或者该时钟对应连续属性的一个取值.

需要注意的是,所有的时间点都为瞬间发生,应该对应到瞬时交互,所以要先对问题图中的持续交互作预处理,将它们转换为瞬时交互.由于一个持续交互可以分为很多个持续交互,每个持续交互又可以由开始瞬时交互和结束瞬时交互表示,因此,一个持续交互可以转换为多个瞬时交互.

(3)识别时间点定性关系

对于(2)中识别出的时间点集合,需要确定它们之间的关系,这些关系包括第2节中介绍的先于(≤)、严格先于(<)和同时发生(≡).这3种关系都具有传递性,也就是说满足如下性质:

$$aRb, bRc \rightarrow aRc,$$

其中 a,b 为时间点,R可以为<,≤或者≡.据此,确定时间点之间的关系可以分为两步:

① 确定具有明确关系的时间点.

② 根据传递性采用推理机推理未明确的时间点之间的关系.

(4)画出时间点图

经过上述3个步骤,得到时钟的6元组定义,将时钟的时间点和时间点之间的关系体现在图形上,就得到了其时间点图.

对于问题图中的任一问题领域,重复步骤(1)~(4),可以得到该问题领域使用(或者参考)的时钟,我们在问题图中引入一个时钟符号(),其中的文字表示时钟名字.将该符号放在问题领域之上,表示该领域使用的是该时钟,由此得到带有时钟的问题图.

3.2 时钟约束关系确定

上一步得到了多个不同时钟,这些时钟并不是孤立存在的,它们共存在于一个系统之中,因此,它们必然存在着某种关系.根据文献[11],其时钟约束关系可以分为两种:

(1)基于同时发生关系(≡)的时钟关系,它可以有很多种,比较常用的有:

等价关系 Equal(A,B),时钟 A 与 B 是等价的.

子时钟 A isFinerThan B,表示 B 是 A 的一个子时钟.它又包含好多种,前一章介绍的 filteredBy 算子也是其中的一种.

(2)基于优先关系(≤)的时钟关系,它可以有很多种,比较常用的有:

周期性 B isPeriodicOn A,表示时钟 B 是 A 的某些时间点的周期性表示.

弱子时钟是子时钟的变体,将子时钟时间点之间≡关系换为≤即可.

在文献[11]还提出很多不同的时钟约束关系,详细内容请参阅文献[11].

对于任意两个时钟,先看它们之间满足什么关系,然后采用上述关系进行描述.通过这一步,我们得到时钟约束关系集合.

3.3 时间点之间的定量约束

除了定性关系之外,还要确定时间点之间的定量关系,也就是时间差,这个时间差的时间单位为其参考时钟的时间单位.

时间差我们采用时差算子(—)来表示,相关约束有3种表达方式:

(1)$a-b<(\leq)$ t unit 表示时间点 a 必须在时间点 b 之后 t 单位时间内发生.

(2)$a-b=t$ unit 表示时间点 a 在时间点 b 之后 t 单位时间点上发生.

(3)$a-b>t$ unit 表示时间点 a 必须在时间点 b 之后 t 单位时间之后发生.

3.4 需求规约抽取

通过前面3步,我们得到时间需求描述,包括软件与其交互环境发生什么样的交互,这些交互之间存在什么样的时间约束.一般来说,需求描述了对环境现象的期望关系,而软件需求工程的目的是要得到需求规约,用来描述软件在环境中的期望行为.为了成为一条需求规约,一个需求必须遵守至少如下3条规则[14]:

(1)需求提到的环境现象都要和机器共享,即规约全部存在于机器和环境的接口.

(2)所有要约束的现象都要直接由机器控制.

(3)所有期望对交互的约束必须用之前发生过的交互表述.

从问题图上可以看出，我们的交互在软件和环境的接口上，且由软件接收或发起，因此，条件(1)和(2)肯定可以满足. 为了满足(3)，还必须要知道这些交互的先后顺序及相关约束.

本文需求规约里面涉及到持续交互，为了清晰起见，我们将需求规约分为两层表示：第 1 层是整体规约，表明在问题图中所有交互(包括持续交互和瞬时交互)的关系，第 2 层是对第 1 层中涉及到的持续交互的规约表示.

整体规约采用顺序流来表示，顺序流由交互关系确定，交互关系有两种，一种是瞬时交互与瞬时交互的关系，跟时间点关系相同，第二种是瞬时交互与持续交互的关系，持续交互可以简化为开始瞬时交互，这样就将持续交互与瞬时交互关系转化为瞬时交互与瞬时交互间关系了. 据此，将整体规约定义为一个二元组，其具体定义见定义 6.

定义 6.　整体规约(Overall Spec).

$$\text{OverallSpec} := \langle IS, < \rangle,$$

其中，

$IS=\{int_1, int_2, \cdots, int_n\}$是所有交互的集合.

$<$是定义在 IS 上的严格先于关系集.

这样的整体规约也可以用一个图形来表示，圆圈表示交互，箭头表示交互发生的严格先于关系，阴影圆圈表示开始，双圈(中间的圆圈有阴影)表示结束.

整体规约可以采用如下步骤获取：

(1) 获取所有的上层交互，包括在问题图中的所有交互.

(2) 对于瞬时交互，根据 3.1～3.3 节得到的对应时间点关系直接排序.

(3) 对于持续交互，在 3.1～3.3 节得到的关系基础上，使用它的开始瞬时交互替换该持续交互，然后根据传递性、时钟约束关系推导出该持续交互与其它瞬时交互对应时间点的关系.

关于持续交互规约，有两层意思需要表达. 首先要表达持续交互的整个流程，包括从开始瞬时交互到结束瞬时交互的所有交互及其关系，它不因有无软件观察而开始或结束，一直按照自己的规律运行. 另一个需要表达的是软件需要观察到的从开始交互到结束交互过程. 由此，用一个四元组来定义持续交互规约，其具体定义见定义 7.

定义 7.　持续交互规约(DurationInstantspec).

$$\text{DurationInstantspec} := \langle IS, <, OI, OE \rangle,$$

其中，

$IS=\{int_1, int_2, \cdots, int_n\}$是这个持续交互的所有瞬时交互 $int_i(1\leq i\leq n)$的集合.

$<$是定义在 IS 上的严格先于关系集.

OI 是软件所要观察的开始瞬时交互.

OE 是软件所要观察的结束瞬时交互.

这样的持续交互规约也可以用一个图形来表示，其图示如图 4 所示. 圆圈表示交互，箭头表示交互流，阴影圆圈表示观察开始交互，双圈(中间的圆圈有阴影)表示观察结束交互. 图 4 的含义如下：a 为一个持续交互，它由瞬时交互 b 开始，经 c 到 d 结束，软件关注的是开始瞬时交互 b，当观察到瞬时交互 c 发生后，软件将不再关注交互 a 了. 因此，c 作为一个观察结束瞬时交互存在于 a 中.

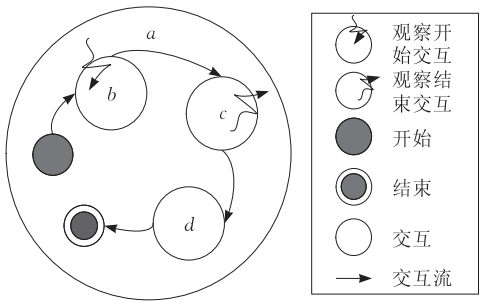


图 4　持续交互规范示意图

根据上面的定义知，可以分两步取得持续交互的规约：

(1) 确定持续交互的所有的瞬时交互及其顺序关系.

(2) 确定软件观察的开始瞬时交互和结束瞬时交互. 一般来说，观察开始瞬时交互就是该持续交互的开始瞬时交互.

4　案例研究

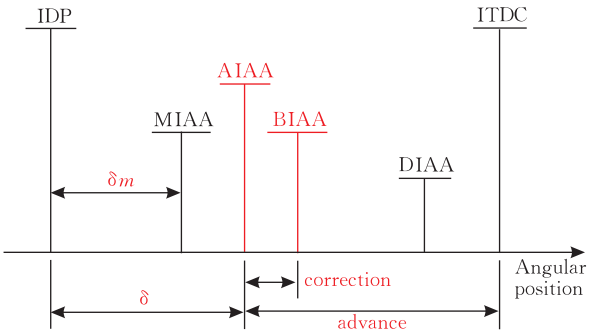
4.1　汽车应用领域问题描述

本文使用汽车领域中的点火系统和爆震控制系统来进行方法验证. 点火系统使用 4 冲程发动机，它包括 4 个阶段：进气、压缩、燃烧膨胀和排气. 这些阶段都是曲轴驱动的，可以由曲轴转动的角度来衡量. 在发动机点火时，由于点燃火花塞需要一定的延迟才能在燃烧室产生火花，点火花必须在理论点火点(TIDC)之前产生. 在压缩阶段，在点火确定点(Ignition Decision Point，IDP)之后，点火系统要决定产生火花的最佳角度. 具体的过程涉及到很多点火点，如图 5 所示. 产生电火花的简化流程如下：

根据当前空气/汽油比率和当前发动机的速度，查询点火控制曲线图(MAP)得到基本点火提前角

(BIAA). 然后修正 BIAA 数据,得到修正点火提前角(CIAA). 修正涉及到很多因素,比如,缸温、缸压、爆震等. 本文为了简化,只考虑爆震因素. 修正会带来两个结果:CIAA 和爆震窗(Knock Acquisition Window,KAW)信息. 这组行为序列是由点火确定点(IDP)触发的.

图 5 中还有最大点火提前角(MIAA),它对应最坏的情况(最大的引擎速度和最大允许提前角).



ITDC: Ignition Top Dead Center
IDP: Ignition DeCision Point
AIAA: Actual Ignition Advance Angle
MIAA: Maximal Ignition Advance Angle
DIAA: Default Ignition Advance Angle
BIAA: Basic Ignition Advance Angle

图 5 点火参数[11]

爆震指得是在发动机燃烧阶段在设计的点火角之前,由于种种原因导致燃气混合物自行点火燃烧,此时,由于燃烧所产生的巨大冲击力与活塞运动的方向相反而引起发动机震动现象. 这种现象会导致发动机动力下降、油耗增加、噪音加大等,可能会对气缸造成不可逆转的损害. 爆震控制系统检测和修正这种现象. 它包含一个或多个爆震传感器和一个控制器,获得和计算修正. 爆震信号的获得在爆震窗(KAW-Knock Acquisition Window)期间执行. 开始点(KAW-Start)和持续时间(KAW-Duration)可能有变化,变化依赖于上一次燃烧阶段测量得到的爆震强度(Knock Intensity,KI). KI 值也将用于调整点火提前角.

爆震过程由压缩阶段的 ITDP 触发,其具体过程如下:根据上次爆震强度 KI 修正之后得到本次 KAW,等待延迟的 KAW-Start,然后执行爆震信号获取(KSA)活动(由 KWB 和 KWE 来表示其开始和结束),KSA 获取填充到一个 Buffer 里面,然后再由 Filtering 读取,Filtering 的最后结果就是 KI.

更多关于汽车点火系统和爆震控制系统的细节,请参见文献[15].

4.2 时间需求建模

4.2.1 问题图及交互预处理

根据上述的汽车点火系统(Ignition System,IS)的陈述,得到其问题图,如图 6 所示. 其含义为:

IS 为要创建的系统;

IS 将要与气缸(Cylinder,CR)、爆震控制系统(Knock System,KS)、凸轮轴(Camshaft,CAS)、曲轴(Crankshaft,CRS)和修正(Correction,CN)这些问题领域进行交互;

IS 的需求就是要确定点火的最佳时机(determine the best spark timing);

IS 与问题领域的交互(包括需求引用和约束),如表 2 所示. 将其中的持续交互转换为瞬时交互,如表 3 所示.

表 2 汽车点火系统的交互及其类型

交互名	发起方	接收方	内容	交互类型
int_1	CR	IS	Speed,AF	Instant
int_2	IS	CR	spark	Instant
int_3	KS	IS	KI	Instant
int_4	CRS	IS	angle	Duration
int_5	CAS	IS	angle	Duration
int_6	CN	IS	KAW	Instant
int_7	CR	IS	Cycle	Duration

表 3 点火系统持续交互转换为瞬时交互

Duration Interaction	Instant interaction	Content
int_4	int_8	OTDC (Overlap Top Dead Center)
	int_9	FBDC(First Bottom Dead Center)
	int_{10}	ITDC(Ignition Top Dead Center)
	int_{11}	SBDC(Second Bottom Dead Center)
	int_{12}	IDP(Ignition Decision Point)
int_5	int_{13}	0°
	int_{14}	90°
	int_{15}	180°
	int_{16}	270°
	int_{17}	332.5°
int_7	int_{18}	Intake open (IO)
	int_{19}	Intake closes(IC)
	int_{20}	Compression opens(CO)
	int_{21}	Compression closes(CC)
	int_{22}	Combustion opens(CP)
	int_{23}	Combustion closes(CL)
	int_{24}	Exhaust opens(EO)
	int_{25}	Exhaust closes(EC)

图 6 中的问题领域爆震控制系统(Knock System,KS)指得是图 7 中要开发的爆震控制系统. 图 7 是爆震控制系统的问题图,其中,

KS 为要创建的系统;

KS 将要与气缸(Cylinder,CR)、爆震传感器(Knock Sensor,KE)、曲轴(Crankshaft,CRS)和修正(Correction,CN)这些问题领域进行交互;

KS 的需求就是要提供爆震强度(provide knock intensity).

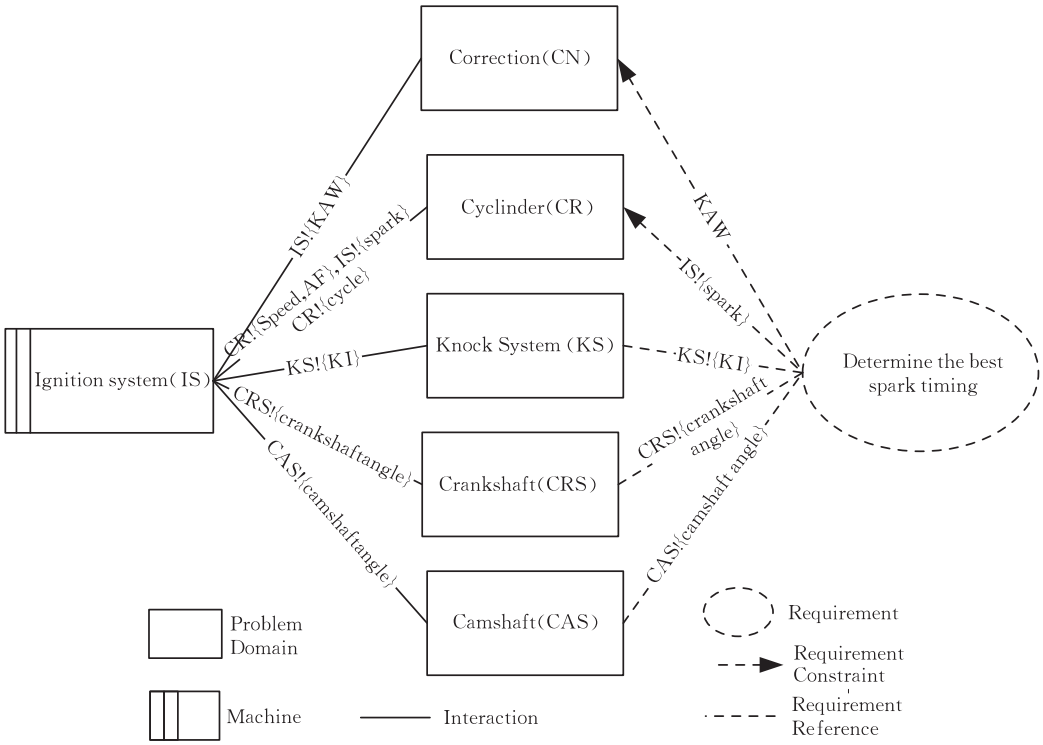


图 6　汽车点火系统问题图

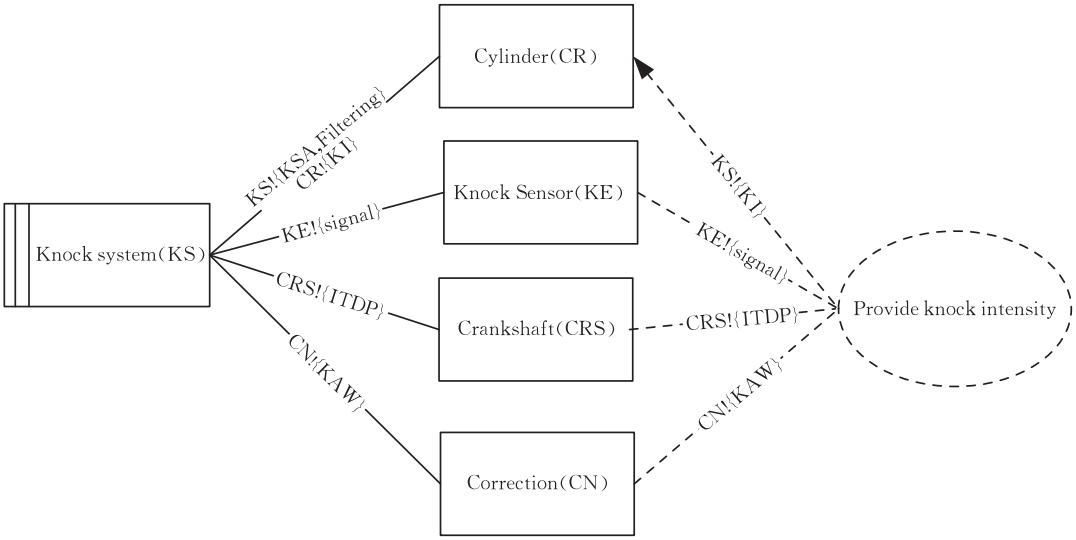


图 7　爆震系统问题图

KS 与问题领域的交互(包括需求引用和约束),如表 4 所示,将其中的持续交互转换为瞬时交互,如表 5 所示.

表 4　爆震控制系统中的交互及其类型

交互名	发起方	接收方	内容	交互类型
int_{26}	KS	CR	KSA	Instant
int_{27}	KS	CR	Filtering	Duration
int_{28}	KS	CR	KI	Instant
int_{29}	KE	KS	signal	Instant
int_{10}	CRS	IS	ITDC	Instant
int_{30}	KS	CN	KAW	Duration
int_{31}	CR	KS	KI	Instant

表 5　爆震控制系统持续交互转换为瞬时交互

DurationInteraction	Instant Interaction	Content
int_{27}	int_{32}	Filtering.start
	int_{33}	Filtering.end
int_{30}	int_{34}	KWS
	int_{35}	KWE

4.2.2　问题领域的时间建模

对于问题图中的每个问题领域,都要对其进行时间建模.接下来的步骤(1)～(3)详细建模了点火系统的问题领域,(4)中建模了爆震控制系统的问题

领域.在开始之前,首先为物理时间定义一个理想时钟 idealClk,它是一个精密时钟,可以被任意时钟引用.

(1)凸轮轴(Camshaft)

凸轮轴是直接驱动摇臂和气门的推杆装置,通过它可以控制节气门,由此控制空气进入发动机.在4冲程发动机中,一个循环它转过360°,粗略来讲,从0°到90°为发动机进气阶段,从90°到180°为压缩阶段,从180°到270°为燃烧阶段,从270°到360°为排气阶段.

凸轮轴的角度可以定义为一个逻辑时钟 camClk,其每个角度的位置可以作为这个时钟的一个时间点.在时钟 camClk 中,其时间单位可以定义为°CAM,分辨率为(比如)0.5,初始值为0,最大值为360,所有的这些值的单位都是°CAM.

这个时钟的时间点可以由时钟变量的取值来标识.根据上述描述知道,需要标识的时间点有0°CAM,90°CAM,180°CAM,270°CAM,360°CAM,其中0°CAM和360°CAM重叠.也可以由对应的交互发生的时间点来表示:$@t_{int_{13}}$,$@t_{int_{14}}$,$@t_{int_{15}}$,$@t_{int_{16}}$,$@t'_{int_{13}}$,其中$@t_{int_{13}}$表示int_{13}发生的时间点,$@t'_{int_{13}}$表示新一轮int_{13}发生的时间点,其它的表示类似.

时钟 camClk 的时间点图如图8所示.

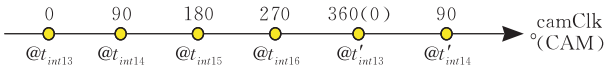


图8 时钟 camClk 的时间点图

(2)曲轴(Crankshaft)

曲轴是引擎的主要旋转机件,装上连杆后,可将连杆的上下(往复)运动变成循环(旋转)运动,曲轴的旋转是发动机的动力源.在4冲程发动机中,一个循环它转过2圈,即720°.

曲轴转动的角度可以定义为一个逻辑时钟 crkClk,其每个角度的位置可以作为这个时钟的一个时间点.在时钟 crkClk 中,其时间单位可以定义为°CRK,分辨率为(比如)0.5,初始值为0,最大值为720,所有的这些值的单位都是°CRK.

这个时钟的时间点可以由时钟变量的取值来标识.根据上述描述知道,需要标识的时间点有0°CRK,180°CRK,360°CRK,540°CRK,720°CRK,其中0°CRK和720°CRK重叠.这些时间点也可以由一些特殊的交互发生的时刻表示,这两种表示之间的对应关系如表6所示.

表6 时钟时间点之间的对应关系

0	180	360	540	720
$@_{int_8}$	$@_{int_9}$	$@_{int_{10}}$	$@_{int_{11}}$	$@_{int_7}$

时钟 crkClk 的时间点图如图9所示.

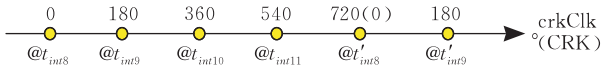


图9 时钟 crkClk 的时间点图

(3)气缸(Cylinder)和爆震控制系统(KS)

这两者的时间都可以用曲轴的角度来刻画,所以还是采用 crkClk 来描述它们.它们涉及的交互之间存在着一定的先后顺序关系.首先要读取当前曲轴角度(int_4),读取当前空气/汽油比率和当前发动机的速度(int_1),根据爆震强度(int_3)进行提前角修正,得到点火时刻(int_2)和爆震窗(int_6).上述交互之间存在如下偏序关系:

$$@t_{int_4}<@t_{int_1}<@t_{int_3}<@t_{int_2}<@t_{int_6} \tag{1}$$

这些交互都是发生在压缩阶段的,存在如下3条关系:

①读取当前空气/汽油比率和当前发动机的速度(int_1)肯定要发生在进气关闭 Intake closes(int_{19})之后,所以有

$$@t_{int_{19}}<@t_{int_1} \tag{2}$$

②得到点火时刻int_2必然发生在压缩关闭 CC(int_{21})之前,因此有

$$@t_{int_2}<@t_{int_{21}} \tag{3}$$

③从理论上来讲,由于气缸的4个冲程状态,在气缸的各个状态交互之间应该存在着这样的偏序关系:

$$@t_{int_{18}}<@t_{int_{19}}<@t_{int_{20}}<@t_{int_{21}}<@t_{int_{22}}<@t_{int_{23}}<@t_{int_{24}}<@t_{int_{25}} \tag{4}$$

即气缸的进气开始 IO(int_{18})之后是进气结束 IC(int_{19}),然后压缩开始 CO(int_{20}),CC 结束(int_{21}),然后燃烧开始 CP(int_{22}),结束 CL(int_{23}),最后排气开始 EO(int_{24})和结束 EC(int_{25}).

但是实际上,气缸的 EC(int_{25})在 IO(int_{18})之后发生,CO(int_{20})在 IC(int_{21})之前发生,CC(int_{21})跟 CP(int_{22})同时发生且发生在 ITDC(int_{10}),EO (int_{24})在 CL(int_{23})之前发生,IO(int_{18})在 EC(int_{25})之前发生.因此,实际上,关系(4)应该是

$$@t_{int_{18}}<@t_{int_{25}}<@t_{int_{20}}<@t_{int_{19}}<@t_{int_{21}}=@t_{int_{19}}<@t_{int_{24}}<@t_{int_{23}} \tag{5}$$

由此得到其时间点图,如图10所示.

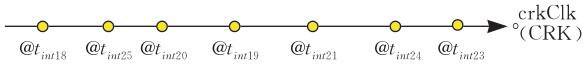


图 10　气缸和爆震控制系统的时间点图

(4)爆震控制系统的问题领域时间建模

爆震控制系统的问题领域包括 CR、KE、CRS 和 CN,这些问题领域的时间都可以通过曲轴的转角来衡量,所以,全部采用时钟 crkClk 来衡量.它们涉及的交互之间存在着一定的先后顺序关系.首先,由 ITDP(int_{10})触发爆震过程,然后根据上次爆震强度 KI(int_{31})修正之后得到本次 KAW(int_{30}),接下来执行爆震信号获取(KSA)活动(int_{26}),读取 Filtering(int_{27}),最后得到 KI(int_{28}).上述交互发生的时间点之间存在如下关系:

$$@t_{int_{10}} < @t_{int_{31}} < @t_{int_{30}} < @t_{int_{26}} < @t_{int_{29}} < @t_{int_{27}} < @t_{int_{28}}.$$

根据上面建模结果,最终可以得到带有时钟的点火系统和爆震系统问题图,如图 11 和图 12 所示.

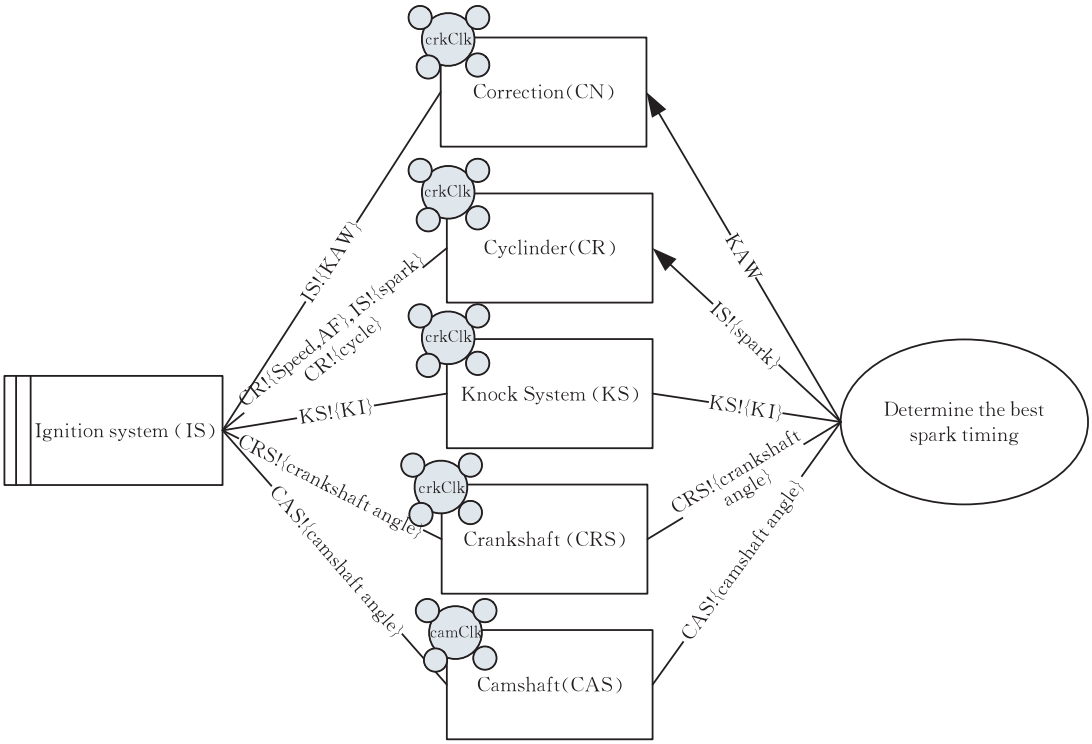


图 11　带有时钟的点火系统问题图

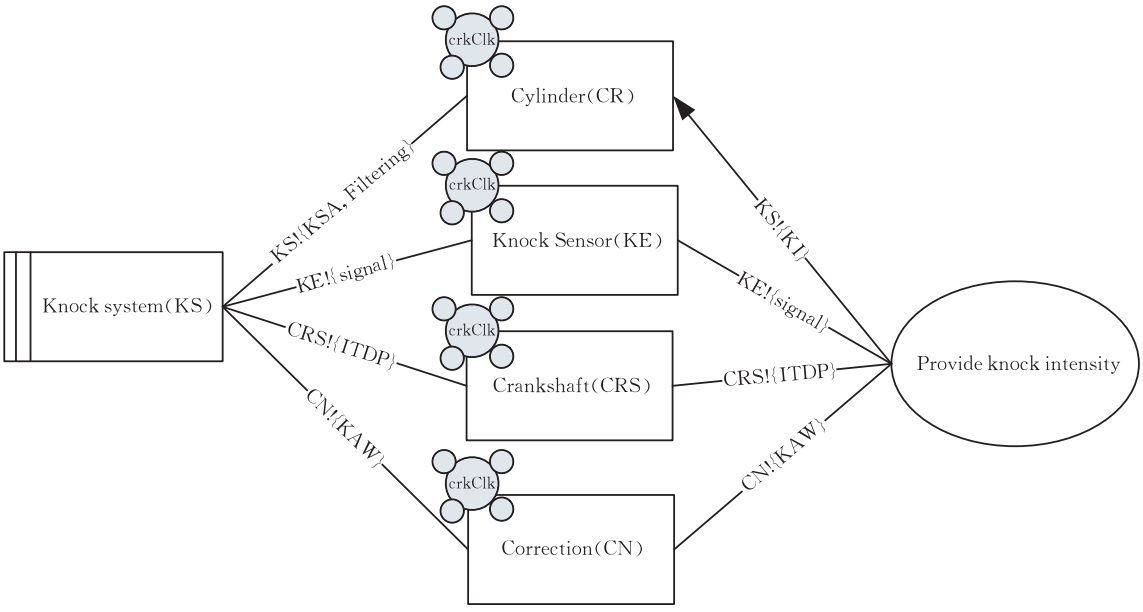


图 12　带有时钟的爆震系统问题图

4.2.3 构建时钟约束关系

(1) camClk 和 crkClk

由于凸轮轴和曲轴是机械上联系在一起的,后者比前者快一倍.因此,时钟 camClk 和 crkClk 紧密地相互依赖,这种依赖关系可以由时钟约束 filteredBy 来表示.图 13 表示了这两个时钟之间时间点之间的关系.

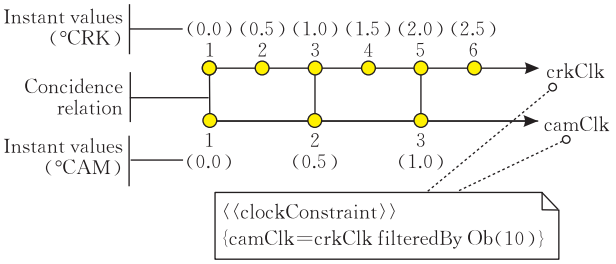


图 13 时钟 camClk 和 crkClk 之间的时钟约束

时钟约束显示 camClk 是 crkClk 的一个子时钟,其中 camClk 的时间点和 crkClk 的每两个时间点之间存在同时发生关系.其具体的时钟约束为 Clock.

Constraint 1:

Clock Constraint 1:

对 camClk 和 crkClk 的约束

camClk=crkClk filteredBy ob(10)

其中,ob(10)是一个周期性的二进制子代表 1010…(其实也就是 10 的无限重复).实际上,时间点之间的关系为

时间点 $0.5\times(k-1) \bmod 720$°CRK 关联在第 $k(k=1,2,\cdots)$个 crkClk 的时间点.

(2) crkClk 和 idealClk

一个发动机的最大速度为 4500 转/分,而一圈所花时间为 360° CRK,则最大的发动机速度为 $(4500/60)\times 360=27000$°CRK/s,因此有

Clock Constraint 2:

对 crkClk 和 idealClk 的约束

1°CRK$\geq 37\,\mu$s

4.2.4 构建时间点之间的定量约束

(1) 点火系统的约束

首先,在点火系统中,气缸的状态和曲轴的位置之间存在着一定的关系,因此在气缸的交互和曲轴的交互发生的时刻点存在着定量的关系.比如,$@t_{int_{25}}$(exhaust closes)在 OTDC 位置出现之后 5~20°CRK,因此存在如下约束:

$$@t_{int_{25}}=@t_{int_8}(\text{OTDC})+[5..20]$$

即为

$$@t_{int_8}+5\text{°CRK}\leq @t_{int_{25}}\leq @t_{int_8}+20\text{°CRK} \quad (6)$$

类似地,可以得到

Constraint 1:

$@t_{int_{18}}=@t_{int_8}-[10..16]$ //IO 在 OTDC 前 10°~16°CRK 发生

$@t_{int_{19}}=@t_{int_9}+[40..60]$ //IC 在 FBDC 后 40°~60°CRK 发生

$@t_{int_{24}}=@t_{int_{11}}-[45..60]$ //EO 在 SBDC 前 45°~60°CRK 发生

$@t_{int_{25}}=@t_{int_8}+[5..20]$ //EC 在 OTDC 后 5°~20°CRK 发生

$@t_{int_{23}}=@t_{int_{11}}$//CL 和 SBDC 同时发生

$@t_{int_{21}}=@t_{int_{22}}=@t_{int_{10}}$ //CC 和 CP 和 ITDC 同时发生

根据上述关系和式(5),可以推导出如下交互发生的时间顺序关系:

$$@t_{int_{18}}<@t_{int_8}<@t_{int_{25}}<@t_{int_{20}}<@t_{int_{19}}<@t_{int_{21}}=@t_{int_{22}}=t_{int_{10}}<@t_{int_{24}}<@t_{int_{23}}=@t_{int_{11}} \quad (7)$$

其次,点火系统的一系列动作由 IDP 触发,而 IDP 出现在一个固定的曲轴角度,它的每次出现即为时钟 crkClk 的一个时间点.而最佳的点火提前角(AIAA)的位置肯定会出现在最大点火角(MIAA,对应着最坏的情况)之前.因此,点火控制行为的间隔(从 reading speed and AF(int_1)到 igniting spark(int_2))必须比图 5 中的 δm(从 IDP 到 MIAA 的时间差)要小.对这个间隔约束可以描述为

Constraint 2:

$$@t_{int_2}-@t_{int_1}<\delta m$$

(2) 爆震控制系统的约束

首先考虑爆震窗.每个气缸都有对 KI 的存储,因此,没有额外的约束.但是,Buffer 是每个气缸共享的,这就意味着每个气缸的 Filtering 操作必须在下一个气缸的 KSA 操作开始之前完成.爆震窗(KAW)在 55°CRK(KAWS)之前开始,最多将会持续 55°CRK(KAWD).因此,在最坏的情况下,整个操作将在 110°CRK 下完成.由此得到如下约束:

Constraint 3:对 KAW 的约束

KAW.start $\leq$ 55°CRK

KAW.duration $\leq$ 55°CRK

即$@t_{int_{32}}\leq$ 55°CRK

$@t_{int_{33}}-@t_{int_{32}}\leq$110°CRK

接下来考虑 Filtering 操作.先考虑单缸发动机的情况.在爆震控制活动中,Buffer 是一个关键的资源,它由 KSA 写入,由 Filtering 读取.在 k 次循环的 Filtering 中,必须在下一次循环 $k+1$ 的 IDP 点出现之前给出爆震强度 KI. IDP 点通常是一个固定的角度(比如,在 ITDC 之后 665°,或者正好在 ITDC 之前 55°).假设给定爆震窗开始的最大值为 KAWSmax 和爆震窗的持续间隔为 KAWDmax,可以推导出

Filtering.duration≤665−KAWSmax−KAWDmax.

若 KAWSmax=KAWDmax=55°CRK

由此可以得到如下约束：

Constraint 4：单缸发动机的 Filtering 约束

Filtering.duration ≤ 555°CRK

即$@t_{int_{35}}-@t_{int_{34}}\leq 555°CRK$

在多气缸的情况下，对 Filtering 的 duration 的约束也就更严格. 交互(ITDC,KWB…)可以由汽缸号来标识：ITDCm、KWB^m、KWE^m 和 KID^m 都指汽缸 m 的交互，而 ITDCn 指得是汽缸 n，它是下一个点火的汽缸. 注意，KID 的最后期限已经不是 IDP，而是 ITDC$_n$. 时间间隔不再是 ITDC$_k$ 和 IDP$_{k+1}$ 之间的 665°CRK 间隔，而是在 ITDCm 和 ITDCn 之间的 720/4=180°CRK，这就给 4 缸发动机留下了 180−110=70 °CRK 的过滤间隔. 因此，得到如下约束 5：

Constraint 5：4 缸发动机的 Filtering 约束

Filtering.duration ≤ 70°CRK

即$@t_{int_{35}}-@t_{int_{34}}\leq 70°CRK$

因此，在最大的发动机速度下，Filtering.duration 不应该超过 70×37=2590 μs，比单汽缸的情况要少很多.

4.3　软件需求规约的抽取

4.3.1　点火系统(IS)

根据第 3 章中软件需求规约的获取步骤，我们首先构建其整体规约. 根据问题图 1 和表 2 知，上层的交互有 int_1，int_2，int_3，int_4，int_5，int_6，int_7. 由表 3 知，int_4 的开始瞬时交互为 int_8，int_5 的开始瞬时交互为 int_{13}，int_7 的开始瞬时交互为 int_{18}，结束瞬时交互为 int_{25}. 这些交互中已经明确关系的有 $@t_{int_4}<@t_{int_1}<@t_{int_3}<@t_{int_2}<@t_{int_6}$(由式(1)知).

其它的推理过程如下：

由式(2)知，$@t_{int_{19}}<@t_{int_1}$

由式(5)知，$@t_{int_{18}}<@t_{int_{19}}$

由传递性可得$@t_{int_{18}}<@t_{int_1}$

由 int_7 和 int_{18} 关系可得

$$@t_{int_7}<@t_{int_1} \tag{8}$$

由 Constraint 1 知：$@t_{int_{18}}=@t_{int_8}-[10..16]$

则$@t_{int_{18}}<@t_{int_8}$

根据 int_7 和 int_{18} 以及 int_4 和 int_8 的关系，可得

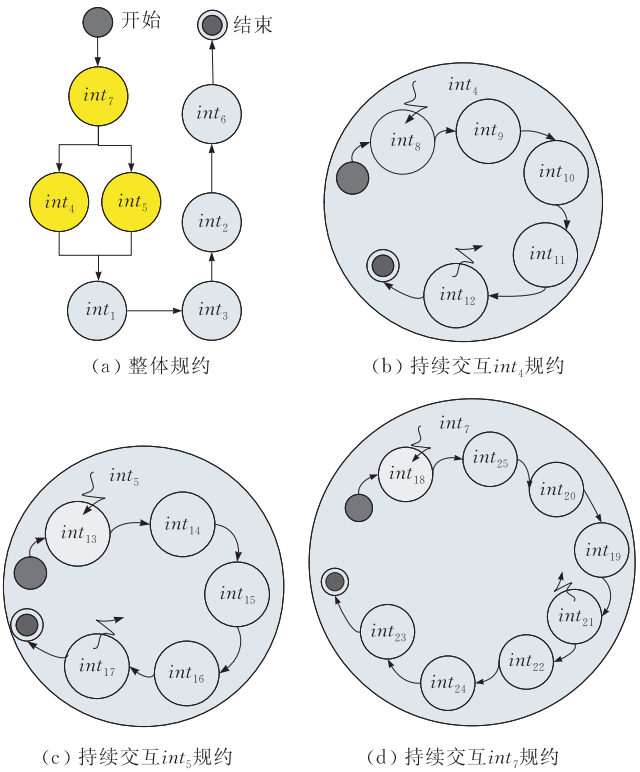
$$@t_{int_7}<@t_{int_4} \tag{9}$$

根据时钟约束关系 Clock Constraint 1 知，int_8 应该和 int_{13} 同时发生，所以

$$@t_{int_4}\equiv @t_{int_5} \tag{10}$$

根据式(1)、(8)、(9)和式(10)，得到点火系统的整体需求规约如图 14(a)所示，其含义为点火系统 IS 查询气缸状态(int_7)、查询曲轴的旋转角度(int_4)和凸轮轴角度(int_5)，当到达 IDP 点时，查询当前空气/汽油比率和当前发动机的速度(int_1)、爆震强度 KI(int_3)，根据点火时机点火(int_2)，并设置爆震窗(int_6).

上述需求整体规约中的 int_4，int_5，int_7 都属于持续交互，分别用图 14(b)、(c)、(d)中的持续交互规约表示. 以 int_4 为例，根据图 14(b)，可以看出其经历了从 OTDC(int_8)到 FBDC(int_9)到 ITDC(int_{10})到 SBDC(int_{11})到 IDP(int_{12})的全过程，当观察到它到达 IDP 时，触发了其它的交互. 关于 int_5，int_7 的描述详见图 14(c)和(d). 这些交互之间要满足 Constraint 1 和 Constraint 2 的约束关系.



(a) 整体规约　(b) 持续交互int_4规约

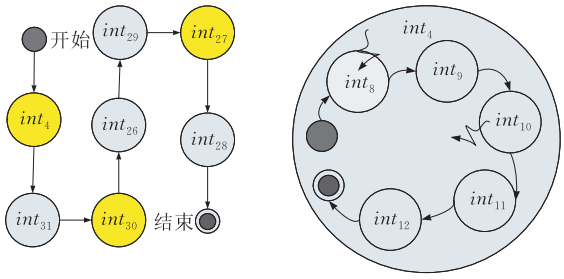
(c) 持续交互int_5规约　(d) 持续交互int_7规约

图 14　点火系统的需求规约

4.3.2　爆震控制系统(KS)

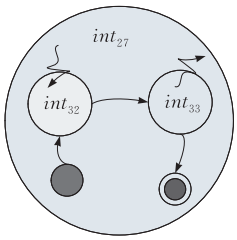
与点火系统类似，可以得到爆震控制系统的需求整体规约，如图 15(a)所示，其含义为：爆震控制系统 KS 查询曲轴的旋转角度(int_4)，当到达 ITDP 点(int_{10})时，查询上次爆震强度 KI(int_{31})，等待延迟的 KAW(int_{30})，执行信号获取(KSA)活动(int_{26})，获取爆震信号(int_{29})，然后执行 Filtering 过滤(int_{27})，得到 KI(int_{28}). 对于其中涉及的持续交互 int_4，int_{27}，int_{30}，我们分别采用持续交互规约来表示，结果如图 15(b)、(c)、(d)所示. 需要指出的

是,图 15(b)中的 int_4 与图 14(b)的 int_4 经历相同,不同之处在于观察结束点不同,图 15(b)中当观察到 ITDP 点(int_{10})时,软件就触发了爆震的其它活动.图 15(c)、(d)则分别表示了 Filtering 和 KAW 的开始和结束.这些交互要满足 Constraint 3 与 Constraint 4(单缸发动机)或者 Constraint 3 与 Constraint 5(4 缸发动机)的约束关系.

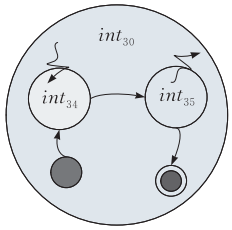


(a) 整体规约

(b) 持续交互int_4规约



(c) 持续交互int_{27}规约



(d) 持续交互int_{30}规约

图 15 爆震控制系统的需求规约

5 相关工作比较

跟本文相关的工作有多形态时间、时间需求建模、问题框架方法和需求规约的抽取.

多形态时间(Multiform Time)最早在反应式同步语言[9]中定义和使用.它认为事件流可以用不同的时间单位来度量.MARTE(Modeling and Analysis of Real-Time Embedded systems)是 OMG 组织公布的实时嵌入系统的建模规范[10].它将多形态时间引入其时间模型.它将时间定义为一组有不同时间单位的时钟.这些时钟可以离散,也可以连续.另外,它还定义了相应的时钟约束语言 CCSL(Clock Constraint Specification Language).该工作是本文工作的研究基础,本文的时间模型也是多形态时间模型,不同之处在于它专门为需求阶段量身打造,其时钟是连续模型,既能描述现实世界,也能描述基于现实世界的变化.

我们将时间需求建模分为 3 种,基于时序逻辑的时间需求建模、基于 MARTE 的时间需求建模和基于时间自动机等形式化方法的时间需求建模.基于时序逻辑的时间需求建模在现代需求工程方法中出现的比较多.例如,面向目标的方法[5]与面向主体和意图的方法[6].面向目标的方法将目标看作是需求的来源,KAOS(Knowledge Acquisition in Automated Specification)[16]是其代表性工作.KAOS 中包含很多传统的时序算子和混合附加的实时算子,由此来确定涉及实时期限的性质.它使用带类型的一阶时序逻辑建模实时特性.

面向主体和意图使用参与者作为线索来识别需求,其代表性工作是 i* 框架[17]和 Formal Tropos[18].Formal Tropos 语言用由 KAOS 激发的时序规约语言补充 i*.Formal Tropos 也使用带类型的一阶时序逻辑建模实时特性.

上述基于时序逻辑的工作,都采用时序逻辑或其扩展对时间需求进行建模,针对时间点和时间段进行了定性与定量的描述.它假设时间领域离散,其时间模型为基于物理时间的多时钟模型,很难精确地捕捉和推理涉及时间的连续变量.多时钟模型涉及多个时钟,这些时钟可以为物理时钟,也可以为逻辑时钟,但这些时钟的时间单位相同,分辨率也相同.这些模型没有解决环境时间的多度量描述的问题.跟它们相比,除了物理时间之外,由于时间的多形态表示,本文方法还可从其它角度来定义逻辑时间,其时间单位可以不同,分辨率也可以不同,更加适用于描述多样的环境.另外,本文方法从环境出发建模时间,也更加适用于描述环境时间,跟上述其它需求工程方法相比,更适合解决多样的环境时间描述与单一软件时间描述的融合问题.

基于 MARTE 的需求工程方法不多.2011 年我们将问题框架和 MARTE/CCSL 结合起来,为嵌入式系统提供了一个时间描述机制[7].该工作将每个问题领域跟一个时钟关联在一起,并将该问题领域涉及的交互对应到该时钟的时间点上,由此构造了嵌入式系统的时间模型.该工作主要针对时间点和时间段进行了描述.实际上,该时间模型是离散的多时钟模型.

形式化方法在时间模型上有很多工作.比如由 Alur 和Dill[23]提出的时间自动机(Timed Automata),它在有限状态自动机的基础上增加了取实数值的时钟变量用以刻画连续变化的时间,可以准确地表示实时系统的各种带时间约束的行为.由周巢尘院士和 Hoare 等人[24]联合提出的时段演算(Duration Calculus),能够应用于对混合系统的实时需求进行刻画和精化以及用来计算关于系统需求的满足概

率. 由牛津大学的 Reed 和 Roscoe 等人[25]提出的时间通信顺序进程(Timed Communicating Sequential Process，TCSP)，引入了时间因子的概念，并定义了一些带有时间因子的算子，实现了实时性的相关描述和分析.

这些形式化的时间模型不大适合在需求的早期使用，且它们大部分是多时钟模型，也存在不能描述多度量方式的环境时间问题.

基于环境的问题框架方法[13]是本文工作的基础. 问题框架方法中涉及时间需求的工作比较少. Lavazza 等人[19]使用时间事件来触发在特定时间点要做的特定系统行为. Nelson 等人[20]为地理领域提出了 12 个地理问题框架，其中一个是跟时间相关的，称为趋势/时序框架，这个框架捕获了这样的问题：当比较不同时间的同一区域的两个地图时，什么发生了变化？ Barroca 等人[21]使用 Timer 作为问题领域的一部分，机器可以观察和重置 Timer，Timer 也可以被客户观察到. Timer 提供了事件可以报告流逝的时间. Bianco 等人[22]使用情景研究问题框架，情景可以用事件现象来表示，其时间约束是用来推理系统的性能. 这些工作都基于时序逻辑及其扩展.

跟上述基于问题框架的时间建模方法相比，我们的时间需求模型是一个多形态时间模型，它建模连续时间，从现实世界出发，从不同的角度看待时间和描述时间，其时间单位多样，分辨率也不同，适合建模多样的环境时间. 由于我们的时间约束基于问题领域和交互，而这二者是功能性需求的主要成分，因此我们的方法植根于功能性需求之上，这使得时间需求有牢固的实现基础.

需求规约的抽取工作跟本文关系最密切的是 Jackson 等人[14]的工作. 他们是从需求中抽取需求规约的专门工作，跟本文工作相比，他们具有更全面的抽取过程，包括如何从需求中通过环境推理出软件应该具备的行为. 它也涉及到一些实时需求，与他们不同的是，我们的工作里面涉及了一些持续交互，并以特定的方式表达了出来.

6　结束语

时间需求是嵌入式系统的核心，其成功描述是实现该系统的关键，因此时间需求建模的重要性毋庸置疑.

本文在我们前期的功能性需求建模的基础上，提出构建基于环境的时间需求模型，主要贡献有：

(1) 以基于环境的功能需求建模为基础，在交互和问题领域两大环境因素的基础上构建多形态时间需求模型，为基于环境的非功能需求建模开辟了道路.

(2) 支持多形态时间建模，从多种角度看待和描述环境时间，适合描述多样的环境时间和单一软件时间.

本文的主要工作包括：

(1) 构建了一个多形态时间需求模型，提供了描述时间需求的基本概念和术语，以帮助进行时间需求的描述.

(2) 在多形态时间需求模型的制导下，提供了一个时间需求的建模过程，支持从功能性需求到时间需求的描述.

(3) 提供了从时间需求模型到需求规约的转换.

通过上述工作，可以得到时间软件需求规约，这样的规约为进一步设计和开发具有强时间约束且与环境多实时交互的软件提供了依据. 目前，本方法提出的时间需求规约还没有进行验证，下一步的工作就是对它们进行验证，并建立工具支持建模和自动化验证.

参　考　文　献

[1] Sarma S, Brock D, Ashton K. The networked physical world. Massachusetts Institute of Technology, Massachusetts: Technical Report MIT-AUTOID-WH-001, 2000

[2] Domestic policy council office of science and technology policy, USA. American Competitiveness Initiative. http://www.casted.org.cn/upload/news/Attach-20080805111202.pdf

[3] Alur R, Courcoubetis C, Dill D L. Model-checking in dense real-time. Information and Computation, 1993, 104(1): 2-34

[4] Alur R, Henzinger T A. A really temporal logic. Journal of the ACM, 1994, 41(1): 181-204

[5] Lamsweerde V A. Goal-oriented requirements engineering: A guided tour//Proceedings of the 5th IEEE International Symposium on Requirements Engineering(RE'01). Toronto, 2001: 249-263

[6] Yu E. Agent orientation as a modeling paradigm. Wirtschaftsinformatik, 2001, 43(2): 123-132

[7] Chen Xiaohong, Liu Jing, Mallet Frederic, Jin Zhi. Modeling timing requirements in problem frames using CCSL// Proceedings of the 18th Asian Pacific Software Engineering Conference(APSEC 2011). Vienam, 2011: 381-388

[8] 何积丰. Cyber physical systems. 中国计算机学会通讯, 2001, 6(1): 25-29(in Chinese)

(He Ji-Feng. Cyber physical systems. Communications of the China Computer Federation, 2011, 6(1): 25-29)

[9] Benveniste A, Caspi P, Edwards S A, Halbwachs N, Le Guernic P, de Simone R. The synchronous languages 12 years later. Proceedings of the IEEE, 2003, 91(1): 64-83

[10] OMG. UML profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE). OMG document number: realtime/2005-02-06. Framing-Ham: Object Management Group, 2005

[11] Mallet F, Peraldi-Frati M, André C. Marte CCSL to execute east-ADL timing requirements//Proceedings of the International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC'09). Tokyo, Japan, 2009: 249-253

[12] Chen Xiao-Hong, Yin Bin, Jin Zhi. Ontology guided requirements modeling based on problem frames approach. Journal of Software, 2011, 22(2): 177-194(in Chinese)
(陈小红, 尹斌, 金芝. 基于问题框架的需求建模:一种本体制导的方法. 软件学报, 2011, 22(2): 177-194)

[13] Jackson M. Problem Frames: Analyzing and Structuring Software Development Problems. Reading, Massachusetts: Addison-Wesley, 2001

[14] Jackson Michael, Zave Pamela. Deriving specifications from requirements: An example//Proceedings of the 17th International Conference on Software Engineering (ICSE'95). Seattle, Washington, USA, 1995: 15-25

[15] Robert Bosch GmbH. Bosch Automotive Handbook. 6th Edition. Manhattan, New York: John Wiley & Sons, 2004

[16] van Lamsweerde A. Formal refinement patterns for goal driven requirements elaboration//Proceedings of the 4th ACM Symposium on the Foundations of Software Engineering(FSE4). San Francisco, USA, 1996: 179-190

[17] Yu E. Modelling organizations for information systems requirements engineering//Proceedings of the 1st IEEE Symposium on Requirements Engineering. San Diego, USA, 1993: 34-41

[18] Yu E. Towards modeling and reasoning support for early-phase requirements engineering//Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97). Annapolis, USA, 1997: 226-235

[19] Lavazza L, Bianco V D. A uml-based approach for representing problem frames//Cox K, Hall J, Rapanotti L eds. Proceedings of the 1st International Workshop on Applications and Advances of Problem Frames (IWAAPF 2004). 2004: 27-34

[20] Nelson M, Cowan D, Alencar P. Geographic problem frames//Proceedings of the 5th International Symposium on Requirements Engineering (RE'01). Toronto, Canada, 2011: 103-104

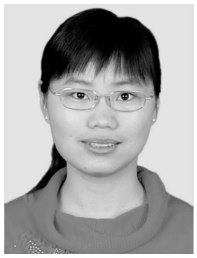
[21] Barroca L, Fiadeiro J, Jackson M, Laney R, Nuseibeh B. Problem frames: A case for coordination//Proceedings of the 6th International Conference on Coordination Models and Languages (COORDINATION2004). Pisa, Italy, 2004: 5-19

[22] Bianco V D, Lavazza L. Enhancing problem frames with scenarios and histories: A preliminary study//Proceedings of the 2nd International Workshop on Advances and Applications of Problem Frames (IWAAPF 2006). Shanghai, China, 2006: 25-31

[23] Alur R, Dill D. A theory of timed automata. Theoretical Computer Science, 1994, 126(2): 183-235

[24] Zhou C C, Hoare C A, Ravn A P. A calculus of duration. Information Processing Letters, 1991, 40(5): 269-276

[25] Reed G M, Roscoe A W. A timed model for communicating sequential processes. Theoretical Computer Science, 1988, 58(1-3): 249-261



CHEN Xiao-Hong, born in 1982, Ph. D., lecturer. Her research interests include requirements engineering and software engineering.

LIU Jing, born in 1965, Ph. D., professor, Ph. D. supervisor. Her main research interests include model driven software development and software engineering.

Background

This work is supported financially by the National Natural Science Fund for the National Basic Research and Development 973 Program (Grant No. 2009CB320702), the National Natural Science Foundation of China for young (Grant No. 61202104, 60903021), the National Natural Science Foundation of China (Grant No. 61170084), Creative Team of NSFC (Grant No. 61021004), the Opening Fund of Top Key Discipline of Computer Software and Theory in Zhejiang Provincial Colleges at Zhejiang Normal University (Grant No. ZSDZZZZXK30), and the Doctoral Fund of Ministry of Education of China (Grant No. 20120076120016), as well as the National High Technology Research and Development Program (863 Program) of China (Grant No. 2011AA010101).

Timing requirements modeling is very important for the embedded systems development. Much work has been done for timing requirements modeling in requirements engineering. For instance, the goal oriented approaches such as KAOS (Knowledge Acquisition in Automated Specification) and the agent oriented approaches such as i* framework and Formal Tropos. KOAS consists of traditional temporal operators, together with additional real-time operators for specifying properties involving real-time deadlines. It models real-time properties concisely by using the typed first-order real-time logic. The Formal Tropos language supplements i* with a rich temporal specification language inspired by KAOS. Formal Tropos also uses a linear-time typed first-order temporal logic.

All these approaches of requirements engineering are based on logic. A limitation of the logic currently used is that the time domain is assumed to be discrete. This makes it difficult to accurately capture and reason about properties involving time derivatives and integrals of time-continuous variables. It is easy for them to describe the time instants and time durations. However, they do not solve the integration problem between multiple environment time description and single software time description in timing requirements.

On the basis of the environment based functional requirements modeling, this paper proposes a multiform time requirement modeling approach. It tries to solve the integration problem of multiple environment time description and single software time description at the requirement level. Our timing requirements model is a multiform time model. It is continuous because it is rooted in the real world other than logic world. It can describe time from different ways using different time units and resolutions, thus making multiple environment timing description possible. Besides, our timing constraints are based on the interactions and the problem domains which are essential parts of functional requirements. In the future, the authors will verify the timing requirements specification, and develop supporting tool for modeling and automated verification.