文章编号:1001 - 9081(2006)01 - 0240 - 03

基于实时语言和面向方面的形式化开发方法

陈生庆[1],张立臣[2],陈广明[1]
(1. 嘉应学院 计算机科学与技术系,广东 梅州 514015;
2. 广东工业大学 计算机学院,广东 广州 510090)
(cgm@jyu.edu.cn)

摘 要:面向方面方法和实时语言特性应用于实时软件开发工程,将降低实时软件开发的复杂性,而形式化方法将提升系统的可信度。该文提出的一种面向方面的实时软件开发方法 AOSDBRTL,它基于经面向方面扩展的形式化方法 AO-RT-Z,在编码阶段应用实时语言 PEARL,实现了软件开发各个阶段对面向方面的无缝支持。

关键词:面向方面;实时系统;形式化方法;RT-Z;AO-RT-Z;PEARL 语言
中图分类号: TP311 **文献标识码**:A

Formal development method based on real-time language method and aspect-oriented

CHEN Sheng-qing[1], ZHANG Li-chen[2], CHEN Guang-ming[1]
(1. *Department of Computer Science & Technology, Jiaying University, Meizhou Guangdong* 514015, *China;*
2. *Computer College, Guangdong University of Technology, Guangzhou Guangdong* 510090, *China*)

Abstract: Aspect-oriented Software Development and real-time Language Programming can reduce the complexity of real-time software. The formal methods can increase dependability of Software Development. The aspect-oriented formal development method AOSDBRTL was established. The real-time language PEARL was used in implementing stage and the AO-RT-Z - the Aspect-Oriented formalism based on RT-Z was used in designing stage. AOSDBRTL was a seamless process on the phases of the software development.

Key words: aspect-oriented; real-time system; formal method; RT-Z; AO-RT-Z; PEARL language

0 引言

面向方面的编程(AOP)吸取了面向对象开发的优点,它克服了面向对象编程(OOP)对于实时约束、安全策略、异常扑获、日志处理这样一些跨越多个对象和模块,横切(crosscut)整个系统的需求的分散处理所带来的问题,同时继承了 OOP 方法的绝大部分优点。成为一种具有生命力的新型程序设计方法。目前支持 AOP 的语言有 AspectC、Aspect C++、AspectJ 等[1]。随着其应用领域的扩展,从软件生命周期的宏观视角构建对 AOP 支持的 AOSD (Aspect-Oriented Software Development)研究成为热点领域。

通常,一个典型的 AOSD 方法应在软件生命周期的各阶段均提供对 AOP 的支持,以此来适应迭代式的开发过程,有效处理各个开发阶段的增量问题。目前这方面的研究主要集中在对传统建模语言和形式化规格说明语言的面向方面扩展[14]。

模型的复杂性和平台相关性使实时软件具有较强的耦合性和较弱的内聚性,不利于通用开发方法的使用和推广。而专门的实时语言能够支持跨平台的编程,如 PEARL 语言[15]编写的实时程序本身能够部分接管通用操作系统的进程管理,以黑盒形式实现对实时特性的支持,从而使开发者可以集中于基本功能的实现。

RT-Z 是 Z 同 Timed CSP 相结合的支持实时并发系统开发的规格说明语言,结合 Object-Z 在语言结构上的优点对其进行了面向方面的扩展,形成了支持组件开发过程的新的规格说明语言 AO-RT-Z,本文将 AO-RT-Z 同 PEARL 相结合,提出面向方面的实时软件开发方法 AOSDBRTL(Aspect Oriented Software Development Based Real Time Languange)。

1 AO-RT-Z

1.1 AO-RT-Z 概述

RT-Z 结合了 Z 的状态描述能力和 Timed-CSP 的进程描述能力,其基本原理是以 Timed-CSP 进程项表达式表示系统的实时、并发关系,用 Z 的操作模式表示进程所形成的状态转化,而 Z 部分的状态及其约束条件又成为系统进程转换的依据[3~6]。RT-Z 以此为基础来定义的语法、结构和语义模型,实际上用 Z 定义 Timed-CSP 的语义模型,即可在保持 Z 的几乎所有语法要素的情况下同 Timed CSP 在语义解释上相一致。由于 Timed CSP 有不同的语义模型,因此这种定义是有差别的,这也使最终形成的规格说明语言规范有所不同。

CBSD(Component Based Software Development)采用黑盒方式,强调的组件的封装和隐藏性,而 AOSD 更强调通过联结过程来改变组件内部的代码,优化或定制系统的横切关注点,采用的是白盒方式,AO-RT-Z 基于“灰盒组件”的概念,它能够在

收稿日期:2005 - 07 - 16;**修订日期**:2005 - 11 - 10 **基金项目**:国家自然科学基金资助项目(60474072;60174050);广东省自然科学基金资助项目(04009465;010059);广东省高校自然科学研究项目(Z03024)

作者简介:陈生庆(1971 -),女,青海湟中人,讲师,硕士,主要研究方向:软件工程技术、实时系统; 张立臣(1962 -),男,吉林长春人,教授,主要研究方向:分布式、实时系统; 陈广明(1971 -),男,广东梅州人,讲师,硕士,主要研究方向:软件工程技术、形式化方法.

尽量保持黑盒组件主要特征的情况下通过定义的方面接口通过联结过程改变其内部代码。整个组件的原理如图 1 所示。

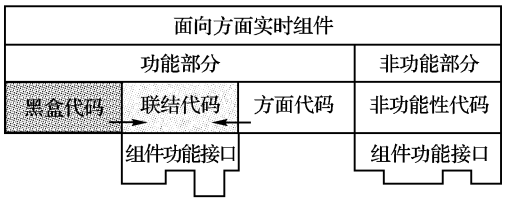


图 1　面向方面的实时组件基本原理

1.2　AO-RT-Z 的框架结构

AO-RT-Z 组件框架改变了 RT-Z 将 CSP 部分和 Z 部分分别定义的结构特点,更直观的体现了组件的封装性,同时按照组件的需要增加了接口和面向方面扩展的语法规范。

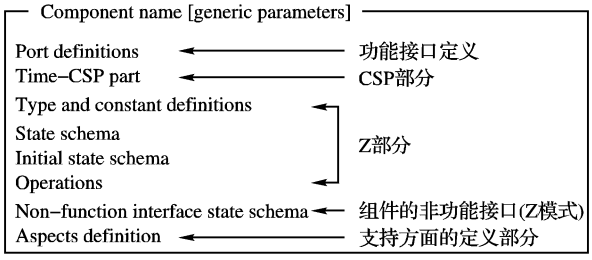


图 2　AO-RT-Z 的实时组件框架

① Port definitions 定义了组件间的功能接口,接口的表示为 Port p:[$p1$:$ty1$; $p2$:$ty2$;…pn:tyn],它代表了组件通信的事件集合,P 是通道名,与其同名的 Z 操作模式定义了对该具体的操作,[$p1$, $p2$, …,pn] 是参数名,可以为空,$ty1$,$ty2$,…,tyn 是参数的类型,不同组件接口通过这些事件进行同步,从而实现了组件的连接。

② Time-CSP 部分是一组 CSP-Name = CSP-Process 结构的表达式集合,此处的 CSP-Process 使用经过 Z 类型扩展的 CSP 模型检查器 FDR(Failure divergence relation) 的语法结构[7],为统一起见,使用关键字 main 作为主进程的标识。

③ Type and constant definitions 定义了组件中的类型和常量;State schema 定义了组件中的状态模式;Initial state schema 初始化时的状态模式;Operations 定义了操作模式,给出了对同名事件的响应操作。

④ Non-function interface state schema 用 Z 的状态模式定义了组件之间的非功能接口部分,它包括 provides 和 allows 两个部分,allows 定义了装配该组件所需要的环境状态及对其他组件的状态要求,provides 表明该组件的非功能状态。

⑤ Aspects definition 部分定义了组件对方面的支持,它是三元组 $< pointcut, advice, mode >$ 的集合,其中 $pointcut$ 是 Time-CSP 部分所含有的进程项,而 $advice$ 同名于某个操作模式的 Time-CSP 进程项,mode := $< posimode, timedmode >$, $posimode \in \{before, after, repeat\}$, $timedmode \in \{< time, event > | time: R^+ \wedge event: \sum^+\}$, $time$ 表示时间项, $\sum^+$ 表示事件集合,其中 NULL 表示空事件,因此二元组 $< time, event >$ 表示事件及其上的时间约束。

1.3　AO-RT-Z 组件的接口

非功能接口采用标准 Z 模式,经转换可用 CSP 的 Failure-Devegence 语义模型表示为二元组 $\langle \{F(C.provide), F(C.allow)\}, \{D(C.provide), D(C.allow)\} \rangle$

定义 1　组件 C_1 与 C_2 是非功能匹配的当且仅当 $F(C.provide) \subseteq F(C.allow) \wedge F(D.provide) \subseteq F(D.allow)$

从定义 1 可知所有能够连接的组件必是非功能匹配的。

(4) AO-RT-Z 组件的方面联结

Time-CSP 项的 BNF 定义为[11]:

$$P ::= STOP \mid SKIP \mid WAIT\ t \mid a \rightarrow P \mid P;P \mid P \xrightarrow{t} P \mid P \square P \mid P \prod P \mid \prod_{i \subseteq I} P_i \mid a:A \rightarrow Pa \mid P_A \mid\mid_A P \mid P \mid\mid\mid P \mid P \backslash A \mid f(P) \mid f^{-1}(P) \mid X \mid \mu X \cdot P$$

定义 2　对于三元组 $< X, Y, < p, < t, u > > >$, X 和 Y 为 Time-CSP 的两个进程项符号, $< t, u >$ 为具有时间约束性的事件,两个进程项符号的胶合表示为 $Z = glue(X, Y)$,如果 u = NULL, $glue(X, Y) = (X \xrightarrow{t} Y)$;如果 u = NULL, $glue(X, Y) = (X \xrightarrow{(t,u)} Y)$;方面的联结(weaving aspects) 过程为:(1) 当 p = $after$ X = $glue(X, Y)$;(2) 当 p = $before$ X = $glue(Y, X)$;(3) 当 p = $repeat$ X = Y。

联结过程将方面定义嵌入了规格说明,从而实现了对 AOP 的支持。

2　PEARL 的实时支持机制

PEARL 出现于德国,上世纪 80 年代 DIN 对其进行了标准化工作,而 90 年代推出的 HI-PERAL 推出极大地提高了语言的功能,使专用实时语言进入了新的阶段[15]。HI-PEARL 对并发实时的支持可以分为代码级支持和运行时支持两种。

代码级支持包括:(a)独立于机器的程序结构,PEARL 的多任务语句独立于特定的计算机,语句可以通过 PEARL 编译器直接转换为操作系统的多任务机制;(b)对共享的基本对象、数组、结构提供保护方式;(c)灵活有效的资源访问方式;(d)时限循环,当循环次数过了规定值,系统将终止循环。

运行时支持是指运行中无须代码支持的策略实现机制,主要包括:(a)同步和资源分配策略[5],系统根据任务的优先级别处理冲突。(b)定时约束, HI-PEARL 可以为每一个任务指定预定的、现有的和剩余的运行时间或时间上限。在紧急情况或过载状态下可以终止任务或用运行时间短的任务替换。(c)过载探测和处理,在任务执行之前,系统通过调度算法检测任务集在最后期限,预测可能出现的过载,如有过载则屏蔽所有的中断、终止所有的任务、删除任务的激活调度;(d)瞬时过载的处理, 对某一任务提供不同的执行体。在任务的定义中包含了运行时间属性,编译器按其降序存储这些选项,可选的任务体被激活时调度程序按照顺序表执行第一个任务选项,如有瞬时过载发生,则选择执行较少运行时间的任务体。(e)调度分析,调度分析器的前端被合并到编译器中,精确遵守定时信息;后端独立于语言和硬件,从前端获得任务争用和延迟等中间量以及最坏情况反应时等信息。

3　AOSDBRTL 的基本框架和应用

3.1　AOSDBRTL 的基本框架

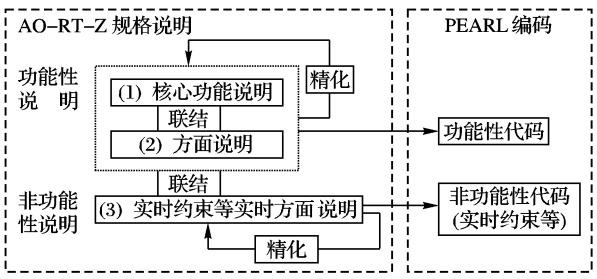


图 3　AOSDBRTL 的基本框架

AO-RT-Z 不仅仅是规格说明语言,同时形式化方法的推理证明机制在 CASE 工具的帮助下可以完成系统的精化,因此也能作为一种建模语言完成系统的设计,在实际设计过程

中将系统的方面定义分为一般方面说明和专门说明实时约束的实时方面说明,由于联结机制,可以建立整个规格说明统一语义,实现形式化方法的推理证明等机制。同时,在编码阶段,仍然能分而治之,充分利用 PEARL 语言的高层实时支持机制同规格说明在组织结构和实现逻辑的内在一致性,灵活方便地实现系统功能。

3.2 开发方法

AO-RT-Z 规格说明包括两个基本模块,功能性说明主要是描述系统的主要功能,它包括核心功能和通过方面联结机制加入的方面规格说明,根据形式化方法的推理证明机制可以推进和检验精化过程的正确性,同时提供的结构化语法支持也能有效地帮助系统的分解和细化,最终达到设计目标。而非功能说明主要包括的是与实时系统相关的实时约束条件,它主要是由 RT-Z 的模式表示,同样也可以完成精化并联结到主功能模块中,这样就形成了统一的语义说明,可对整个说明进行统一处理。

设计阶段的形式化描述在代码实现时是分离的,这样可以非常方便地利用 PEARL 语言的实时支持机制,其中功能部分由 PEARL 的普通语句实现,而实时约束部分则可以通过 PEARL 的实时支持语句所完成。这样在简化编码过程的同时程序的可读性和可维护性也得到了很大的提高。

3.3 应用实例

中断系统是读者所熟悉的,以此为例来说明在 AOSDBRTL 的应用。篇幅所限仅给出设计的部分片段,有关 RT-Z 规格说明和 PEARL 语言代码的细节可参阅文献[15]。

(a) 实时约束说明的部分 RT-Z 描述:.

EA2 ≙ ∀ t: T; int: IntID; pri: IntPri · SysInt_Req. (int, pri) at t→SysInt_Resp. (int, pri) Open from $t + t_{sysreq} + t_{sysresp}$

/ * 系统对反应时间的要求 * /

(b) (a)所对应的 PEARL 代码为:

```
supervise: TASK GLOBAL;                    / * 被主任务激活 * /
SysSchedule : =0B;  Int-signal : =0B
EXPECT
AWAIT WHEN SysSchedule DO
AWAIT WHEN Int-signal : =1B DO
IF Int-signal AND Time_of_sysresp <0.01 SEC
  THEN
     ALL 0.03 SEC EXACTLY ACTIVATE IntSchedule;
     ALL 0.03 SEC EXACTLY ACTIVATE IntResp;
     ALL 0.05 SEC EXACTLY ACTIVATE IntEnters;
     OUIT;
  ELSE IntSchedule : =0B;
     FIN;
  FIN;
END;
```

(c) 中断系统功能性说明部分 RT-Z 描述:

系统方面说明同系统功能规格说明经联结而成为具有统一语义的 RT-Z 规格说明。其中黑体部分为联结后增加的方面代码,它表示了横切系统的错误处理机制,方面联结定义为集合:{ < *InterReqSuccx*, *Int_Req*, < *after*, < 0, ∅ > >, < *Dispacth*, *Int_Req*, < *after*, < 0, ∅ > > }, 联结后的规格说明为:

```
BehaviourI ≙ S
IS ≙ IntReqSetEmptyx → Join; IS □IntReqSetNonEmpty → ( Join
     □Check); IS
Join ≙ IntReq_Entry?req? = req: IntIDIntPri→
  ( InterReqSuccx? < | req? = req | > →Int_Req! < | req? = req | >
     →skip□InterReqFailx? < | req? = req | > Jionx < | req? = req |
     > skip)
Check ≙ Int_Check?reqempty: IntIDIntPri→
  ( CheckReqSuccx? Par[ req! : req IntIDIntPri] →
  Dispacth! < | req? = req | > Int_Req! < | req? = req | > →skip)
```

以上黑体部分表示了联结过程在原有功能说明的基础上所增加的进程。

(d) (c)对应的 PEARL 代码为:

```
Behaviour: TASK DUE AFTER 3 SEC TIME SYSTEM KEEP;
                                          / * 循环处理 * /
IF IntReqSetEmpty AND TIME <0.01
  THEN
     ALL 0.03 EXACTLY Join;
  ELSE
     ALL 0.03 EXACTLY Join;
     ALL 0.03 EXACTLY Check;
  FIN;
  END;
TASK-BODY Join WITH-RUNTIME 0.01 SEC;
TASK-BODY Check WITH-RUNTIME 0.01 SEC
```

4 结语

本文建立了一个简单的实时系统开发框架,它以形式化方法为基础,提供了对实时和嵌入式系统的重要组件规格和设计的无二义性,同时利用了实时语言的特点,保证了代码质量。而 AOSD 的引入则在很大程度上降低了开发的难度,AOSD 的理念正逐步成为软件开发的重要方法,随着 CASE 工具的开发和应用,相信一定会成为 OOSD 的重要补充。

同形式化方法相比较,UML 具有更为简单的语言特点,可视化的编程方式以及比较完善的 CASE 工具支持,同 AOP 技术相结合的研究也取得了一定的进展,充分利用 AOP 的思想和 UML 的强大的建模能力和形式化方法的严谨性及语义,是今后研究的一个方向。

把时间约束分离出来构造为一个时间方面,建立一个比较完整的时间模型来建模系统时间,同时促使时间方面与系统其他方面分离,进行单独的时间约束设计,根据需要把设计好的时间方面织入到系统中,以简化实时系统建模复杂性,是今后的另一个研究方向。

同时,对现有的实时语言进行 AOP 扩展,使之同 AOSD 实现无缝联结具有重要的实用价值,也是值得关注的问题。

参考文献:

[1] The AspectJ Programming Guide[EB/OL]. http://aspectj.org/doc/dist/progguide/index.html, 2002.

[2] 任洪敏, 钱乐秋. 构件组装及其形式化推导研究, 软件学报, 2003, 14(6): 1669 - 1677.

[3] SUHL C. RT-Z: An integration of Z and timed CSP[A]. AGT99[C], 1999. 51 - 65.

[4] SUHL C. Applying RT-Z to develop safety-critical systems[A]. Proceedings of the Third International Conference on Fundamental Approaches to Software Engineering (FASE 2000)[C]. number 1783 in Lecture Notes in Computer Science, Springer-Verlag, 2000. 51 - 65.

[5] FISCHER C. How to combine Z with a process algebra[A]. BOWEN J, FETT A, HINCHEY M, eds. ZUM'98 The Z Formal Specification Notation, volume 1493 of LNCS[C], 1998. 5 - 23.

[6] ANDREWS JH. Process - algebraic foundations of aspect - oriented programming[A]. Processdings of the Third International Conference on Metalevel Architectures and Separtion of Crosscutting Concerns, volume 2192 of Lecture Notes in Computer Science[C]. Benlin, 2001. 187 - 209.

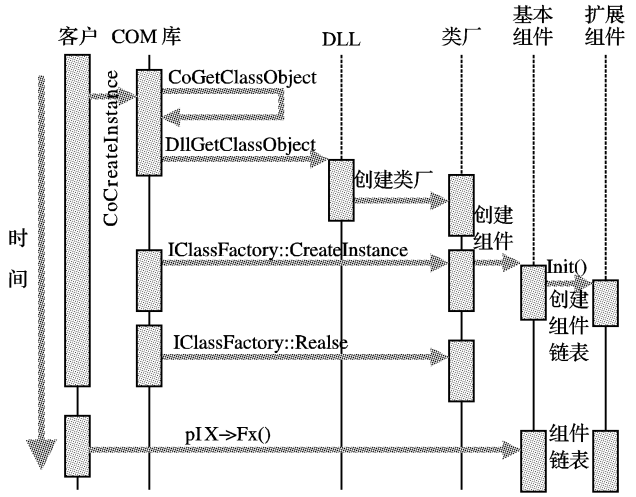
[7] RAISE Language Group. The RAISE Specification Language. BCS Practitioner Series[M]. Prentice-Hall, 1992.

(下转第 248 页)

类厂,并查询类厂接口。

然后,CoCreateInstance 使用返回的类厂接口,来调用类厂的 CreateInstance 函数。该函数先调用组件的 CreateInstance 函数(在具体组件实现的文件中定义)创建基本组件,接着调用组件的 Init 函数(在组件的基类中实现)创建扩展组件并建立组件链表。最后查询组件的 QueryInterface(在组件的基类中实现,并在具体组件实现的文件中重载),返回接口指针。

第二步,客户调用返回的接口指针,执行接口定义的功能。



5 个结构化元素:客户、COM 库、DLL、类厂、组件
实线——元素已经建好并活动的
虚线——元素不再活动
灰框——每个操作生命期

图 2 CoCreateInstance 创建组件的过程

从上面过程可以看出,这与原来 COM 调用组件的过程几乎一样,只不过增加了基本组件初始化创建扩展组件的一步,而且这一工作主要由基类提供,对具体组件实现的工作量增加很少。

3.3 在 CAD 软件中如何应用扩充技术

当不同的软件功能由不同的团队开发,并且独立发布时,采用这种技术可以方便的添加组件功能,而无需更改被扩展组件的代码,甚至可以实现扩展组件的动态发布。例如,一个机翼可以用一个组件表示。一个开发团队可以开发机翼的空气动力学基本性能(如基本流场计算),其他团队开发独立出售的高级功能(如振动性能)。这些独立开发的组件只要遵循上述机制,发布后就可以无缝的连接起来,从外部视点来看,就像一个组件。

当产品采用打包配置时,不同的配置组件的数量不同,同一组件实现的功能可能也有不同。因此,对有些组件采用这种扩展机制,可以根据企业的需要,方便、低成本的升级,而不影响以前的投资。

软件给用户提供二次开发的平台,如果支持这种技术,可以使用户的扩展与原来的系统无缝连接,并且不会影响到原有的应用。

3.4 扩充技术的不足和需要进一步研究的问题

尽管这种新技术有上述的优点,但也存在着一些不足。例如:

当分别在基本组件和扩展组件中实现的接口,被频繁的交替查询时,这种机制的查询效率会降低。幸运的是,这种情况出现的几率比较低。

由于要维护一个组件字典,给使用和维护带来了一些不便。不过由于使用和维护都很简单,而且只对组件“服务器端”有影响,对使用这些组件的“客户”而言,几乎没什么影响。

对于已有的组件和购买的组件,由于基类没添加相应的数据和方法,所以无法在不改变源代码的情况下使用该方法。

组件字典的定义、内容和如何维护是需要仔细研究的一个重点,这关系到性能、功能和易用性。另外,通过组件字典是否也可以实现其他技术(例如,条件接口)也是一个需要研究的问题。

4 结语

组件模型对软件开发来说是一个关键因素,本文针对目前 CAD 开发的趋势,提出采用在 COM 组件模型的基础上增加适当的特征的组件模型选择方案,可以最大限度地利用当前市场上已有组件,便于与其他专业公司合作开发,而且比自行开发组件模型的风险要小的多。在 COM 组件模型的基础上,我们提出一个组件复用和扩展的新技术,并在VC++6.0 环境中,实现了这种技术。结果表明,采用它可以有效的支持组件的组合、复用和动态扩展。

参考文献:

[1] IVICA C, MAGNUS L. Building reliable component-based software systems[M]. London: Artech House publishers, 2002. 375 - 386.

[2] 叶修梓, 彭维, 唐荣锡. 国际 CAD 产业的发展历史回顾与几点经验教训[J]. 计算机辅助设计与图形学学报, 2003, 15(10): 1185 - 1193.

[3] 白跃伟, 陈明, 陈卓宁, 等. 基于 Windows 组件可复用的 CAD 系统开发[J]. 计算机工程与应用, 2004, (13): 99 - 101.

[4] ROGERSON D. Inside COM[M]. Washington: Microsoft Press, 1997.

[5] WHITEHEAD K. Component - based Development [M]. London: Addison Wesley, 2003. 17 - 113.

(上接第 242 页)

[8] 陈广明, 陈生庆, 张立臣. Z 实时扩展及基于多视点的应用模式[J]. 计算机应用, 2005, 25(2), 362 - 365.

[9] Ftp://ftp.irt.uni-hannover.de/pub/pearl/report.pdf[EB/OL].

[10] FISCHER C. How to combine Z with a process algebra[A]. BOWEN JP, FETT A, HINCHEY MG, eds. ZUM'98 The Z Formal Specification Notation, volume 1493 of LNCS[C], 2000.

[11] STANKAVIC J, ZHU R, POORNALINGAM R, *et al.* VEST.: an aspect-based composition tool for real-time systems[A]. the 9th Real-Time Applications Symposium 2003[C]. Toronto, Canada: IEEE Computer Society Press, 2003. 110 - 123.

[12] STOYENKO AD, MARLOWE TJ. polynomial-Time Transformations and Schedulablity Analysis of Parallel Real-Time Programs with restricted Resource Contention[A]. Real-Time Systems[C], 1992. 307 - 329.

[13] SCHNEIDER S. An Operational Semantics for Timed CSP[A]. number 1453 in Lecture Notes in Computer Science[C]. Springer-Verlag, 1997. 45 - 61.

[14] 陈广明, 张立臣, 陈生庆. 面向方面的实时软件开发方法[J]. 计算机科学, 2005, 32(7).

[15] STOYENKO A, HALANG W. High - Integrity PEARL and its Schedulability Analyzer: Transferring State-of-the Art Real-Time Software Technology from University Laboratories to industry[R] Tech. report CIS-91-16, CIS Dept, NEW Jersey Institute of Technology, Newark, 1991.