基于UML的计算机联锁软件的分析与建模

武晓春 高雪娟
(兰州交通大学自动化与电气工程学院 兰州 730070)

摘 要 有效地测试、分析和验证计算机联锁软件是保证列车运行安全和旅客生命财产安全的重要手段,而形式化模型是系统测试、分析和验证的基础。以联锁软件的UML非形式化模型为基础,以有限状态机模型为系统形式化模型描述的数学工具,研究UML顺序图(场景)自动转化为有限状态机模型的方法。首先将场景的UML顺序图转化为FSP进程代数模型,然后通过合并不同对象的进程代数模型,得到系统的有限状态机模型。最后以接车进路用例为例生成系统的有限状态机模型,以验证该方法的可行性和有效性。

关键词 联锁软件,UML,顺序图,FSP,有限状态机
中图法分类号 TP311.5 **文献标识码** A

Analysis and Modeling of Computer Interlocking Software Based on UML

WU Xiao-chun GAO Xue-juan
(School of Automation and Electrical Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract It is an important way to make sure the safety of train running and passengers' life and property by effectively testing, analyzing and validating computer interlocking software. Formal model is the foundation of system testing, analyzing and validating. Based on interlocking software's UML informal model, using finite state machine model as the mathematical tools to describe system formal model, this paper studied the method to traverse the UML sequence diagram or scenarios to finite state machine model. Firstly the UML sequence diagram was traversed to FSP process mathematical model, and then systematic finite state machine model was obtained by merging all objects' process mathematic models in the UML sequence diagram. Finally, the case of controlling of entry routes was used to generate systematic finite state machine model to invalidate the feasibility and effectiveness of this method.

Keywords Interlocking software, UML, Sequence diagram, FSP, Finite process machine

铁路计算机联锁软件是一种安全苛求软件,它的错误输出尤其是危险侧的安全防护失效将可能造成重大的生命财产损失。但在联锁软件的开发过程中引入软件缺陷往往是不可避免的,一方面是软件需求说明不够详尽或者说明有误造成的,另一方面是人类大脑对复杂联锁逻辑系统理解与控制的局限性造成开发人员考虑不周而引起的[1]。国内不同厂家生产的计算机联锁系统的软件测试报告也显示,联锁软件均存在一些错误[2]。因此,如何保证计算机联锁软件的高安全和高可靠性,确保联锁软件执行的正确性,满足联锁的"故障—安全"是急需解决的问题。

目前,对于车站联锁系统功能的安全性保障主要通过模拟验证[3,4]和仿真测试[5,6]的手段来验证和确认。文献[3]采用Rhapsody工具对联锁软件的UML模型进行功能模拟来判断功能的安全性。文献[4]采用EVALPSN程序模拟器对联锁系统进行模拟验证。文献[5]研究了基于测试的计算机联锁软件的安全性评价基准。文献[6]采用Dijkstra分布式终止探测算法对铁路控制系统软件进行了测试。文献[7]采用符号模型检测技术和CTL时序逻辑,应用Versus工具验证了联锁系统的安全逻辑。文献[8]采用故障树分析结合LTS模型检测的方法对安全苛求系统的安全性进行分析与验证,提出基于模型检测的软件安全性验证方法。以上的形式化方法需要较深的数学知识,难以得到推广和应用。

本文采用UML为计算机联锁软件建模,不仅可以使联锁逻辑关系更加清晰易懂,还可以改进领域专家和开发人员的交流,有益于完善铁路联锁软件的需求规格说明。以进路建立用例为例,用顺序图描述该用例的场景,并对各场景进行一致性分析,引入有限状态自动机FSM实现对联锁软件的UML模型形式化,得到该用例更精确和细致的模型。该模型可作为联锁软件系统分析、验证和测试的基础。

1 计算机联锁软件建模

1.1 计算机联锁软件

联锁机软件是整个联锁系统中最为关键的部分,其软件的功能安全直接决定了整个系统的功能安全。联锁机软件应当具有如下功能[9]:

(1)信息交互功能:指一方面能够发送信息到上位机,一

到稿日期:2013-04-27 返修日期:2013-10-08 本文受基于受控拉格朗日函数的多欠驱动度力学系统控制器设计(61164010)资助。
武晓春(1973—),女,副教授,主要研究方向为交通信息工程及控制;高雪娟(1990—),女,硕士生,主要研究方向为交通信息工程及控制、软件测试,E-mail:gaoxuejuan126@163.com。

方面能够接收上位机的按钮操作信息。

(2)联锁运算功能：主要指进路控制功能，包括进路的建立、进路锁闭、信号开放、进路正常解锁、进路非正常解锁、道岔单操等。

(3)驱采功能：指能够根据联锁运算的结果发送驱动命令到驱动板，同时能够不断接收采集板得到的现场设备的实时信息。

1.2 UML 的建模方法

标准建模语言 UML[10] 定义良好、易于表达、功能强大，提供了一整套描述软件模型的概念和图形表示法，可以对联锁软件的静态结构和动态行为进行建模，适用于系统开发过程中从需求规格分析到最后测试交付的各个不同阶段。如通过用例图从用户的角度来捕获系统、子系统或类的行为，通过类图和对象图来描述系统对象及对象间的关系，通过顺序图、协作图和状态图描述对象间的交互关系，通过组件图和配置图来描述软硬件体系结构及通信机制。

1.3 联锁软件需求分析

需求分析就是确定系统的目的、范围、定义和功能。通过系统需求规范、系统功能需求规范以及系统与外围接口的关系，明确系统需要完成哪些功能[11]。系统的需求分析通过 UML 用例模型实现。

根据《计算机联锁技术条件》，可构建整个联锁机系统的用例图，如图 1 所示。

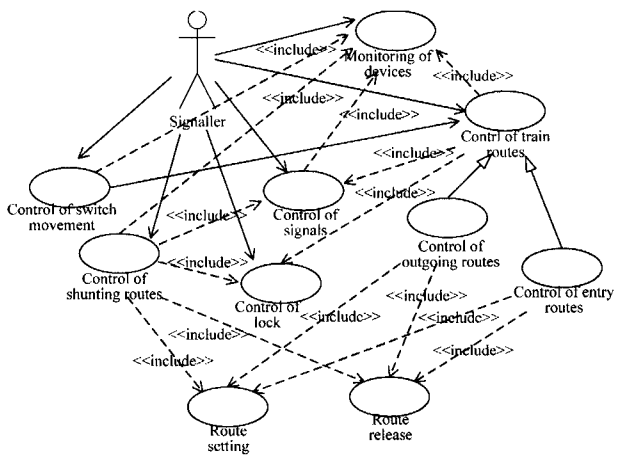

图 1 联锁机软件用例图

车站信号员(Signaller)实现对道岔的控制(Control of switch movement)、进路的建立及解锁，包括列车进路(Control of train routes)和调车进路(Control of shunting routes)，其中列车进路又包含接车进路(Control of entry routes)和发车进路(Control of outgoing routes)、信号设备(包括信号机、道岔和轨道区段)的监控。进路处理是联锁软件的核心功能。

2 联锁软件系统分析

通过分析需求分析的每一个用例模型，得到实现这一用例的基本类及其之间的关系。以办理接车进路用例为例进行分析。该用例包含 5 个类，即信号员、联锁系统、道岔、区段和信号机。车站信号员选出进路后，对应进路有其相应的道岔、防护区段以及防护该进路的信号机及敌对信号。车站调度员从开始办理进路到防护该进路的信号机开放，可分为以下 4 个阶段：(1)选路阶段：根据调度员的操作，确定对应的进路；(2)选排一致检查及道岔控制命令生成阶段：检查进路中的道岔位置是否符合进路要求的位置，若不符合，检查道岔所在区段是否空闲，道岔是否被单锁，如果空闲且为单锁，则转换道岔至进路规定的位置；(3)进路锁闭阶段：在道岔位置正确、进路空闲、敌对进路未建立的情况下，锁闭道岔和进路；(4)开放信号阶段：开放信号，指示列车可进站。

采用 UML 顺序图描述接车进路用例，如图 2 所示，该用例对应 9 个场景。

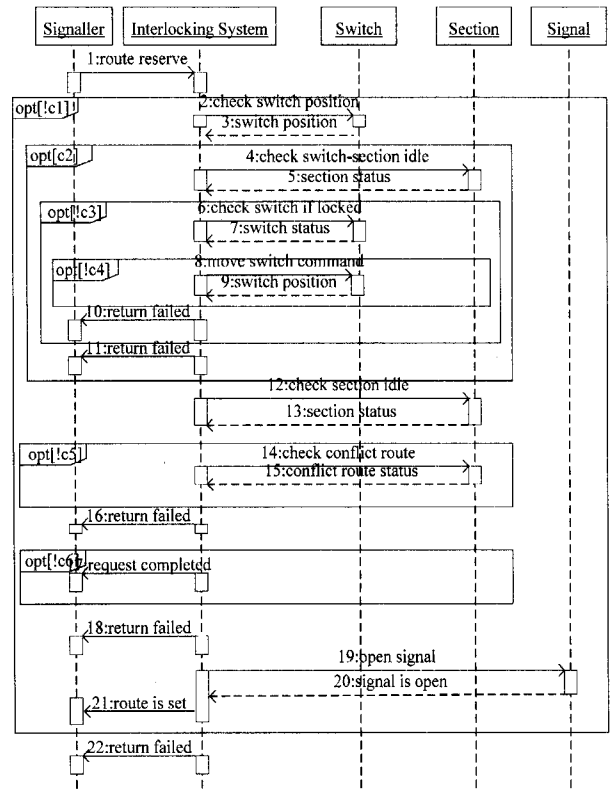

图 2 接车进路用例的顺序图

各监护条件 $ci(i=1,\cdots,6)$ 表示含义见表 1。

表 1 各监护条件表示含义

监护条件	表示意义
c1	操作不规范无对应进路
c2	进路所包含的道岔位置不正确
c3	道岔所在区段有车占用
c4	道岔锁闭
c5	进路包含区段不空闲
c6	敌对进路已建立

2.1 系统需求场景的一致性分析方法

对用顺序图描述的不同系统需求场景，根据消息的前/后置条件对顺序图进行矛盾检测及矛盾消除，识别出 UML 顺序图中对象包含的顺序行为序列、可选择性行为及并发行为，并对多个 UML 顺序图中包含的相同或相似行为进行合并。

其具体过程如下：

(1)根据需求场景，把系统设计过程中需要定义的全局变量组合成一个状态向量 $S=\langle s_1, s_2, \cdots, s_n\rangle$，其中 $s_i(1\leqslant i\leqslant n)$ 表示状态变量，若 s_i 未知，则 s_i 为 null。

(2)根据需求场景，结合领域知识，分析 UML 顺序图中各个消息的前/后置条件是否一致。分析依据为：

①对于单个 UML 顺序图，若 m_{i-1} 与 m_i 为两个顺序发生的消息，且 $post\_v_{i-1}\neq pre\_v_i$，则 m_{i-1} 与 m_i 之间存在矛盾。

②对于两个 UML 顺序图 UML_i 和 UML_j，假设 $UML_{i,mi}$ 为 UML_i 中的消息，$UML_{j,mj}$ 为 UML_j 中的消息，且 $UML_{i,mi}$ 与 $UML_{j,mj}$ 为同一消息，若 $UML_{i,mi}$ 与 $UML_{j,mj}$ 的状态向量值不相等，则 UML_i 与 UML_j 之间存在矛盾。

(3)根据领域知识，对检测出来的矛盾进行分析，若需求场景不正确，则修改需求场景，得到一致的 UML 顺序图。

2.2 接车进路用例一致性分析

下面以进路中包含的道岔位置是否正确、其他联锁条件满足的条件的两个用例进行场景一致性分析。

根据需求场景，选取布尔型的车站联锁系统的状态向量：SwitchPosCorrect(道岔位置是否在规定位置)，SwitchSecIdle(道岔所在区段是否空闲)，SwitchLock(道岔是否被单独锁闭)，SectIdle(进路中的区段是否空闲)，ConflictRouteBuilt(敌对进路是否建立)，SignalOpen(信号机是否开放)，RouteBuilt(进路是否建立)。

当进路中道岔位置正确，其他联锁条件满足时，即满足 $!c1\&!c2\&!c5\&!c6$ 条件时，其对应的消息的前/后置状态向量值见表 2。

当进路中道岔位置不在进路规定位置，其他联锁条件满足时，即满足 $!c1\&c2\&!c3\&!c4\&!c5\&!c6$ 条件时，其对应的消息的前/后置状态向量值见表 3。

根据消息之间的矛盾检测情形，表 2 和表 3 中无消息矛盾，满足场景一致性。分析其他 7 个场景消息的前/后置条件，无消息矛盾，接车进路用例的对应顺序图满足一致性要求。

表 2 UML 顺序图道岔位置正确场景中消息的前/后置状态向量值

M	Pre-state vector value	Post-state vector value	M	Pre-state vector value	Post-state vector value
m_1	⟨null,null,null,null,null,f,f⟩	⟨null,null,null,null,null,f,f⟩	m_{15}	⟨t,null,null,t,null,f,f⟩	⟨t,null,null,t,t,f,f⟩
m_2	⟨null,null,null,null,null,f,f⟩	⟨null,null,null,null,null,f,f⟩	m_{17}	⟨t,null,null,t,t,f,f⟩	⟨t,null,null,t,t,f,f⟩
m_3	⟨null,null,null,null,null,f,f⟩	⟨t,null,null,null,null,f,f⟩	m_{19}	⟨t,null,null,t,t,f,f⟩	⟨t,null,null,t,t,f,f⟩
m_{12}	⟨t,null,null,null,null,f,f⟩	⟨t,null,null,null,null,f,f⟩	m_{20}	⟨t,null,null,t,t,f,f⟩	⟨t,null,null,t,t,t,f⟩
m_{13}	⟨t,null,null,null,null,f,f⟩	⟨t,null,null,t,null,f,f⟩	m_{21}	⟨t,null,null,t,t,t,f⟩	⟨t,null,null,t,t,t,t⟩
m_{14}	⟨t,null,null,t,null,f,f⟩	⟨t,null,null,t,null,f,f⟩			

表 3 UML 顺序图道岔位置不正确场景中消息的前/后置状态向量值

M	Pre-state vector value	Post-state vector value	M	Pre-state vector value	Post-state vector value
m_1	⟨null,null,null,null,null,f,f⟩	⟨null,null,null,null,null,f,f⟩	m_{12}	⟨t,t,t,null,null,f,f⟩	⟨t,t,t,null,null,f,f⟩
m_2	⟨null,null,null,null,null,f,f⟩	⟨null,null,null,null,null,f,f⟩	m_{13}	⟨t,t,t,null,null,f,f⟩	⟨t,t,t,t,null,f,f⟩
m_3	⟨null,null,null,null,null,f,f⟩	⟨f,null,null,null,null,f,f⟩	m_{14}	⟨t,t,t,t,null,f,f⟩	⟨t,t,t,t,null,f,f⟩
m_4	⟨f,null,null,null,null,f,f⟩	⟨f,null,null,null,null,f,f⟩	m_{15}	⟨t,t,t,t,null,f,f⟩	⟨t,t,t,t,t,f,f⟩
m_5	⟨f,null,null,null,null,f,f⟩	⟨f,t,null,null,null,f,f⟩	m_{17}	⟨t,t,t,t,t,f,f⟩	⟨t,t,t,t,t,f,f⟩
m_6	⟨f,t,null,null,null,f,f⟩	⟨f,t,null,null,null,f,f⟩	m_{19}	⟨t,t,t,t,t,f,f⟩	⟨t,t,t,t,t,f,f⟩
m_7	⟨f,t,null,null,null,f,f⟩	⟨f,t,t,null,null,f,f⟩	m_{20}	⟨t,t,t,t,t,f,f⟩	⟨t,t,t,t,t,t,f⟩
m_8	⟨f,t,t,null,null,f,f⟩	⟨f,t,t,null,null,f,f⟩	m_{21}	⟨t,t,t,t,t,t,f⟩	⟨t,t,t,t,t,t,t⟩
m_9	⟨f,t,t,null,null,f,f⟩	⟨t,t,t,null,null,f,f⟩			

3 联锁软件形式化模型生成方法

本节提出基于场景分析的系统形式化模型生成方法。该方法从基于 UML 顺序图的需求描述出发，根据消息的前/后置条件进行一致性分析。通过结合领域知识，对多个 UML 顺序图中包含的相同或相似行为进行合并，在得到一致的需求场景的基础上，识别 UML 顺序图中对象交互的行为序列将其转换成相应的 FSP 进程代数模型，并将其做组合运算后，得到系统的有穷状态机 FSM(Finite States Model)。

3.1 系统对象的 FSP 模型生成方法

FSP 能够有效地描述并发系统和反应式系统的动态行为，是一种描述进程模型的代数标记，它通过层次、并发和广播通信机制来描述和解释系统的复杂行为，刻画了进程在其生命周期内接受激励后的状态变化过程[12]。

在 FSP 中主要有以下一些操作符：

(1)顺序：->，表示动作间的顺序。如 $x->P$ 表示先执行动作 x，再执行进程 P 描述的动作序列。

(2)选择：|，表示不同执行动作的选择。如 $x->P|y->Q$ 表示或者先执行动作 x，再按照 P 的描述执行；或者先执行动作 y，再按照 Q 的描述执行。

(3)条件控制：when，表示在给定条件下执行某一动作。如 when B $x->P|y->Q$ 表示：若条件 B 成立，则除了 $y->Q$ 以外可以选择执行动作 x 再执行进程 P 的动作序列；若 B 不成立，则只能执行动作 y，然后再执行进程 Q 的动作序列。B 是一个状态谓词。

(4)并发组合：||，表示多个简单进程间的组合。如 $P||Q$ 表示在条件允许的情况下并发执行进程 P 和 Q 中的动作序列。

从 UML 顺序图得到 FSP 进程代数模型的流程图如图 3 所示。

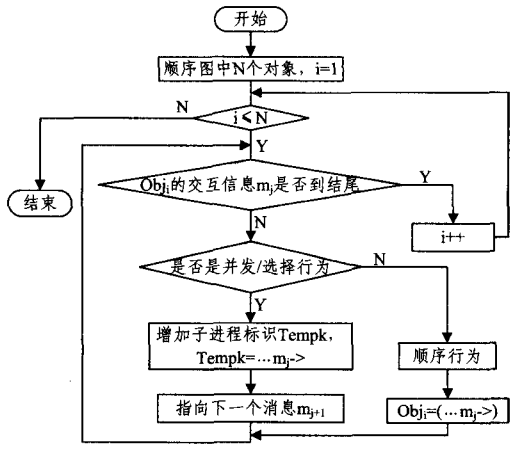

图 3 从顺序图生成 FSP 模型的算法流程

输入：顺序图

输出：各对象的 FSP 模型

步骤：

(1)判断当前对象是否是顺序图中的对象；

(2)判断该对象 Obj_i 当前交互信息 m_j 是否是结束信息；

(3)判断该信息是否是并发/选择行为；

(4)如果该信息是并发/选择行为，则相应地增加子进程标识 $Temp_k$；

(5)否则就是顺序行为，则将该信息顺序添加到 FSP 模型；

(6)递增到下一个信息。

3.2 接车进路用例的形式化模型生成

接车进路用例对应的各对象的 FSP 进程代数模型如下：

Signaller=(m_1 −> when (c1) m_{22} | when (!c1&c2&!c3&c4)m_{10} | when(!c1&c2&c3)m_{11} | when(!c1&(c2&!c3&!c4|!c2)&c5)m_{16} | when(!c1&(c2&!c3&!c4|!c2)&!c5&c6)m_{18} | when(!c1&(c2&!c3&!c4|!c2)&!c5&!c6)m_{17} −>m_{21})

Interlocking System=(m_1 −>when(c1)m_{22} | $Temp_1$)

$Temp_1$=(m_2 −>m_3 −>when(c2)$Temp_{11}$ | $Temp_{14}$)

$Temp_{11}$=(m_4 −>m_5 −>when(!c3)$Temp_{12}$ | m_{11})

$Temp_{12}$=(m_6 −>m_7 −>when(!c4)$Temp_{13}$ | m_{10})

$Temp_{13}$=(m_8 −>m_9 −>$Temp_{14}$)

$Temp_{14}$=(m_{12} −>m_{13} −>when(!c5)$Temp_{15}$ | m_{16})

$Temp_{15}$=(m_{14} −>m_{15} −>when(!c6)$Temp_{16}$ | m_{18})

$Temp_{16}$=(m_{17} −>m_{19} −>m_{20} −>m_{21})

Switch=(when (!c1) m_2 −> m_3 −> when (c2&!c3) $Temp_2$)

$Temp_2$=(m_6 −>m_7 −>when(!c4)$Temp_{21}$)

$Temp_{21}$=(m_8 −>m_9)

Section=(when(!c1&!c2)$Temp_3$ | when(!c1&c2)m_4 −>m_5 −>$Temp_3$)

$Temp_3$=(m_{12} −>m_{13} −>when(!c5)$Temp_{31}$)

$Temp_{31}$=(m_{14} −>m_{15})

Signal=(m_{19} −>m_{20})

对已经生成的 FSP 进程代数模型进行组合后，得到系统的有穷状态机模型，如图 4 所示。

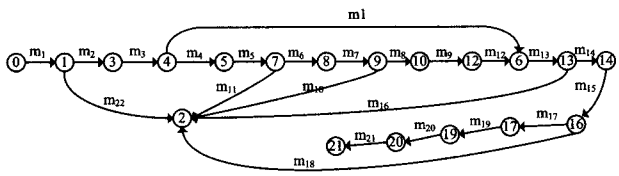

图 4 系统的有穷状态机模型

结束语 本文采用 UML 时序图描述系统需求场景，通过对 UML 顺序图中的消息前/后置条件进行分析，得到一致的需求场景；在识别 UML 顺序图中对象交互的行为序列及其类别的基础上，通过系统形式化模型生成算法将 UML 顺序图转换成 FSP(Finite State Process)，并得到系统的有穷状态机模型。该方法为安全苛求领域面临的形式化建模难这一问题的解决提供了有效途径，也为形式化验证与分析过程中系统的形式化建模提供了新思路，从安全质量方面改善了安全苛求软件的设计与开发，丰厚了基于模型的软件形式化开发方法。

参 考 文 献

[1] 王铁江，郦萌. 计算机联锁软件的 Z 规格说明[J]. 铁道学报，2003，25(4)：62-66

[2] 吴芳美. 计算机联锁软件测试评估[J]. 铁路计算机应用，1999，8(1)：7-10

[3] 李颖. 基于 UML 的车站信号软件建模[D]. 北京：北京交通大学，2008

[4] Nakarnatsu K，Kiuchi Y，et al. Intelligent Railway Interlocking Safety on Annotated Logic Program and Verification Based its Simulation[C]// Proceedings of the 2004 IEEE International Conference on Networkinp，Sensing & Control. Taipei，Taiwan，2004

[5] 吴芳美. 计算机联锁软件基于测试的安全性评价基准研究[J]. 铁道学报，2005，27(3)：97-101

[6] Blom S，Ioustinova N，Pol J，et al. Simulated Time for Testing Railway Interlockings with TTCN-3[C]//Proceedings of the 5th International Workshop on Formal Approaches to. Testing of Software. LNCS 3997，2006：1-15

[7] Garmhausen V H，Campos S，Cimatti A. Verification of a safety-critical railway interlocking system with real-time constraints [J]. Elsevier Science of Computer Programming，2000(36)：1546-1563

[8] 王曦，徐中伟，梅萌. 基于模型检测的软件安全性验证方法[J]. 武汉大学学报，2010，56(2)：156-160

[9] 赵志熙. 计算机联锁系统技术[M]. 北京：中国铁道出版社

[10] Arlow J，Neustadt J. UML2 and the Unified Process[M]. China Machine Process

[11] 王帅，吉吟东，杨士元. 一种基于场景的 CTCS-3 列车控制系统建模方法研究[J]. 铁道学报，2011，23(9)：55-61

[12] Magge J，Krammer J. Associated Concurrency：State Models and Java Programs[M]. Wiley，1999

(上接第 218 页)

调度策略可降低实时系统的任务截止期错失率。接下来的工作是研究适合 TSCTTL 调度策略的优先级分派策略，及采用 TSCTTL 策略对 CPU 利用率的影响情况。

参 考 文 献

[1] Nagy S，Bestavros A. Admission control for soft-deadline transactions in ACCORD[C]// Proceedings of the 3rd IEEE Real-Time Technology and Applications Symposium. Montreal，Canada，1997：160-165

[2] 夏家莉. 支持替代/补偿的实时调度策略[J]. 小型微型计算机系统，2005，26(2)：248-251

[3] 金宏，王强，王宏安，等. 基于动态抢占阈值的实时调度[J]. 计算机研究与发展，2004，41(3)：393-398

[4] 李琦，巴巍. 两种改进的 EDF 软实时动态调度算法[J]. 计算机学报，2011，34(5)：943-950

[5] Liu C L，Lavland J W. Scheduling algorithm for multipro gramming in a hard real-time environment[J]. Journal of ACM，1973，20(1)：40-61

[6] Semghouni S，Amanton L，Sadeg B，et al. On new scheduling policy for the improvement of firm RTDBSs performances [J]. Data & Knowledge Engineering，2007，63(2)：414-432