软件学报ISSN 1000-9825, CODEN RUXUEW E-mail: jos@iscas.ac.cn

*Journal of Software*, [doi: 10.13328/j.cnki.jos.000000] http://www.jos.org.cn

©中国科学院软件研究所版权所有. Tel: +86-10-62562563

基于UML的实时性建模介绍及分析[[1]](#footnote-1)\*

摘 要: 随着对复杂的系统任务需求的增加，系统的及时性成为了软件开发者们比较关注的一个问题，系统的及时性不仅会影响用户使用，甚至可能会影响系统的安全，大多数现有的建模方法都是面向功能的模型，没有考虑系统中的及时性问题，本文总结了一种考虑复杂系统时效性的建模方法，该方法主要是把实时嵌入式系统添加到UML中，用其来描述时间信息。称为UML/MARTE模型，本文介绍了MARTE的相关知识，以及该模型的形式定义，最后以飞机着陆为例，验证了该模型的有效性。

关键词: 及时性，MATRE,UML模型扩展

State-of-the-Art Survey of Static Software Defect Prediction

**Abstract**: With the increasing demand for complex system tasks, the timeliness of the system has become a concern of software developers. The timeliness of the system will not only affect the user's use, but may even affect the security of the system. Most existing ones The modeling methods are all functional-oriented models. Without considering the timeliness in the system, this paper summarizes a modeling method that considers the timeliness of complex systems. This method mainly adds real-time embedded systems to UML and uses them. To describe the time information. Called the UML/MARTE model, this paper introduces the relevant knowledge of MARTE and the formal definition of the model. Finally, the aircraft landing is taken as an example to verify the validity of the model.

**Key words**: Timeliness, MATRE, UML model extension

# 绪论

像轨道交通系统，航空运输系统等复杂系统不仅关注性能，同时还对一些非功能的属性要求较高，比如本文即将分析的及时性问题。系统及时性是指系统在有效时间内的响应能力。随着任务要求的不断增加，系统的及时性日益突出，目前，针对系统的及时性研究主要集中在医疗，战斗指挥，金融系统等对及时性要求较高的领域。在这些研究中，人们尝试引入权重和动态模拟的方法来分析系统的及时性，但在一般的复杂系统中，并没有系统及时性的讨论，而且复杂系统的高度集成给构建模型带来了很大的困难，所以大多数现有的建模方法都是面向功能的建模，没有考虑系统中的及时性问题。

UML是一种通用的统一建模语言，它能通过多个角度描述系统的结构和行为特征，作为UML的扩展，实时嵌入式系统（MARTE）具有时间建模能力，弥补了UML在时间属性建模中的不足，针对复杂系统的时效性问题，本文对影响系统时效性的因素进行了总结，并介绍了一种通过UML扩展机制将MARTE整合起来的模型， 最后通过飞机着陆实例对该模型进行了分析。

# UML扩展及实时嵌入式系统介绍

## UML介绍

统一建模语言（Unified Modeling Language,UML）是一种可视化的建模语言，它能让系统构造者用标准的、易于理解的方式建立起能够表达他们想象力的系统蓝图，并且提供了便于不同人之间有效地共享和交流设计结果的机制。

UML作为通用的可视化建模语言，用于对软件进行描述、可视化处理、构造和建立软件系统制品的文档。它记录了对必须构造的系统的决定和理解，可用于对系统的理解、设计、浏览、配置、维护和信息控制。UML 适用于各种软件开发方法、软件生命周期的各个阶段、各种应用领域以及各种开发工具，UML 是一种总结了以往建模技术的经验并吸收当今优秀成果的标准建模方法。UML包括概念的语义，表示法和说明，提供了静态、动态、系统环境及组织结构的模型。它可被交互的可视化建模工具所支持，这些工具提供了代码生成器和报表生成器。UML标准并没有定义一种标准的开发过程，但它适用于迭代式的开发过程。它是为支持大部分现存的面向对象开发过程而设计的。

UML描述了一个系统的静态结构和动态行为。UML将系统描述为一些离散的相互作用的对象并最终为外部用户提供一定的功能的模型结构。静态结构定义了系统中的重要对象的属性和操作以及这些对象之间的相互关系。动态行为定义了对象的时间特性和对象为完成目标而相互进行通信的机制。从不同但相互联系的角度对系统建立的模型可用于不同的目的。

UML还包括可将模型分解成包的结构组件，以便于软件小组将大的系统分解成易于处理的块结构，并理解和控制各个包之间的依赖关系，在复杂的开发环境中管理模型单元。它还包括用于显示系统实现和组织运行的组件。

面向对象的分析与设计(OOAD)方法的发展在80年代末至90年代中出现了一个高潮,UML是这个高潮的产物。它不仅统一了Booch、Rumbaugh和Jacobson的表示方法,而且对其作了进一步的发展,并最终统一为大众所接受的标准建模语言。

面向对象技术和UML的发展过程可用下图来表示,标准建模语言的出现是其重要成果。在美国,截止1996年10月,UML获得了工业界、科技界和应用界的广泛支持,已有700多个公司表示支持采用UML作为建模语言。1996年底,UML已稳占面向对象技术市场的85%,成为可视化建模语言事实上的工业标准。1997年11月17日,OMG采纳UML 1.1作为基于面向对象技术的标准建模语言。UML代表了面向对象方法的软件开发技术的发展方向,具有巨大的市场前景,也具有重大的经济价值和国防价值。

## UML扩展

为了避免UML语言整体的复杂性，UML没有吸收所以的面向对象的建模机制和技术而是设计了扩展机制，通过扩展机制用户可以定义使用自己的元素。在前边介绍UML构成的时候，提到了UML的扩展机制：版型、标记值，约束。在很多情况下我们利用UML的版型这种机制对UML进行扩展，使其能够应用到更广泛的领域。

### 2.2.1版型

版型是建模元素的一种类型，扩展UML的语义。版型必须以UML中已经定义的元素为基础，可以扩展语义但不能扩展已存在的元素结构。版型不是给元素增加新的属性或约束，而是直接在已有元素中增加新的语义，这种机制可以看作是已有元素进行专有化。版型的表示方法是在模型元素的旁边添加一个版型的名称，版型名称使用双括号括起来，《版型名》。版型是一种非常好的扩展机制他避免了UML语义过于复杂化，同时也使得UML能够适应各种需求。

通常人们在特定方法或特定的应用领域中使用UML时，会使用版型。有些概念、方法或特定领域特有标注UML不支持，用户就可以自定义。自定义版型时需要作以下工作：描述自定义版型的基础是哪个元素；对该元素语义的影响；给出使用该版型的例子。

利用UML的扩展机制对UML进行扩展是已经非常有意义的工作，有时我们需要使用UML来建模，但是UML本身提供的元素满足不了我们的需要，此时并不意味者UML没有用了，而是需要我们应用UML扩展机制来实现自定义元素，从而实现建模。

### 2.2.2标记值

标记值是一对字符串包括标记字符串和值字符串也就是一个键值对，它存储着有关元素的一些信息。标记值可以与任何元素相关联，包括模型元素和表达元素。标记表示建模者想要记录的名字，值是给定元素的值。通常使用类似下面的方式表示：

｛Name=“Tom”｝

如果标记是个布尔类型，可以省略其值，默认为真。除了布尔类型以外其它的类型都必须明确写出值。

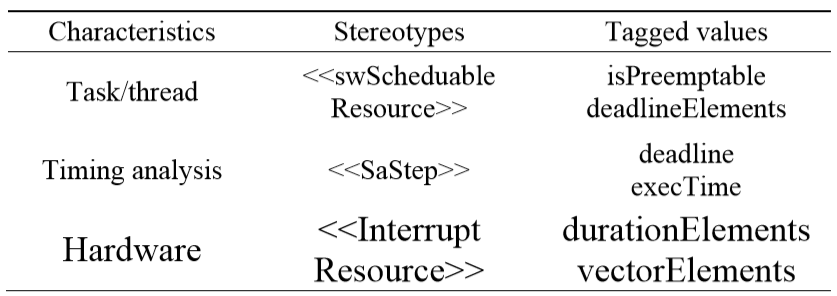
### 2.2.3约束

约束是用文字表达式表示的语义限制，它应用于元素。约束显示在｛｝内，可以直接放在图中或者独立出来。约束可以是UML预定义的，也可以之自定义的。描述约束的语言可以是正式的数学符号、OCL语言、伪代码或自然语言。如果这种语言是非正式的，那么它的解释也是非正式的，并且要由人来解释。即使约束由一种正式语言来表示，也不意味着它自动为有效约束。

## 实时嵌入式系统介绍

虽然UML具有强大的描述功能，但任然无法满足各个领域的建模需求，因此，UML提供了一种灵活的扩展机制，允许用户根据需要扩展UML。

MARTE规范是OMG发布的UML的扩展，弥补了UML在非功能属性建模中的缺点，MARTE采用部分排序的时间模型。MARTE有三种主要的时间抽象类用于表示行为流程。第一个是因果/时间：在这个模型里，我们只关心指令优先/依赖的问题。第二种是时钟/同步：这种时间模型增加了同时的概念，并将时间尺度话分为一系列不连续的瞬间。第三种是物理/实时：这类时间抽象要求对实时持续的时间进行精确建模，以便解决在关键系统中的调度问题。



MARTE为肺功能属性提供了预定义的构造型和标记值。上图是MARTE中定义的版型和标记值。本文介绍的方法采用了SaStep和标记值deadline execTime来描述系统的时间和约束信息。

# UML/MARTE 时间模型

## 系统的延迟种类

由于技术限制和人为因素，系统在运行过程中具有各种延迟，这些延迟反应了系统中的时间特征。这对系统能否成功完成任务具有重要影响。根据复杂系统的功能结构和任务过程的特点，系统延迟可以分为以下五类：

信息获取延迟：指信息获取设备获取原始信息所用的时间。

信息处理延迟：指处理情报信息，控制信息，协作信息等所用的时间。

信息传递延迟：指原市信息和处理过的各类信息在传送过程中花费的时间。

系统相应延迟：指接收到执行单元的指令后的相应时间。

系统操作延迟：指执行单元执行操作的时间。

系统中的各种延迟将不可避免地影响任务完成时间。只有在指定时间内完成指定的操作或者功能时，操作才有效。系统中的延迟不仅会影响系统任务，甚至可能导致安全问题，根据系统的流程，及时性定义为：复杂系统的及时性是系统在随机系统延迟的情况下，在系统组件未发生故障且人员决策和操作正确的情况下，在指定时间内完成任务的概率。

## UML/MARTE时间模型

对于复杂系统，这里采用类图，序列图和活动图来构件模型。用于构建静态结构模型的类图描述了系统中使用的类以及它们之间的关系; 序列图描述了系统中对象之间按时间顺序交换的消息，重点是消息传递过程; 活动图描述了系统功能的动态行为和活动之间的顺序关系，表示了系统的整个任务流程。

在本文中，基于复杂系统的及时性，我们利用UML的扩展机制添加MARTE规范的模型来表示系统延迟。

UML / MARTE时间模型由UML和MARTE注释组成。 本文使用带有MARTE注释的类图（CD / MARTE），带有MARTE注释的序列图（SD / MARTE）和带有MARTE注释的活动图（AD / MARTE）来描述系统的时间属性和约束。

1. CD/MARTE时间模型

定义：带有MARTE标注的类图。

类图通常被用来搭建一个静态的结构模型，这个模型可以描述每个对象的属性和对象之间的关系。时间限制通常用添加到对象以及对象关系上的MARTE标注来描述。

下图是一个带有MARTE标注的类图，有一个自定义类型《SaStep》和类图中的标记值：执行时间和结束时间，分别代表对象执行事件的时间和执行事件的最大事件。

CD/MARTE的通用定义是一个带有MARTE标注的类图是一个结构体(,)

是类图的名称

是复杂系统中新的类的集合

是新类的关系的集合，包含了关联，泛化，依赖和实现等。

是一个类的属性

是一个类的操作，代表一个类的行为。

是MARTE标注的集合。一个MARTE标注含有一组自定义类型和标记值。

1. SD/MARTE时间模型

序列图是按时间顺序在系统中的对象之间传递的消息的直观表示。它用于描述消息交互过程，我们可以通过在对象图中添加MARTE标注到它们之间的消息来描述时间约束。

下图是一个带有MARTE标注的序列图，有一个自定义类型《SaStep》和一个标记值：执行时间和结束时间，分别代表对象对象传输消息的最大时间以及传输消息所花费的时间。

CD/MARTE的通用定义是一个带有MARTE标注的类图是一个结构体(Nsd, Lsd, Esd, Msd, Tsd, MAsd, <sd⊆Osd×Osd, LOCsd:Osd→Lsd)

是序列图的名称。

是序列图生命线的集合。

是执行说明，表示一个处理从开始到结束的区间，在UML2.0中被称为激活。

是一组时间观测，一个时间观测由一个是时间值和事件发生说明组成。

<sd⊆Osd×Osd是一个总的排序函数，描述了两个相邻发生事件之间的顺序关系。

是MARTE标注的集合。一个MARTE标注含有一组自定义类型和标记值。

LOCsd:Osd→Lsd是一个定义发生事件说明位置的函数。

1. AD/MARTE time model

定义：带有MARTE标注的活动图。

活动图描述了动态功能，活动顺序以及系统的并发行为。 将MARTE注释添加到活动节点和活动图中

的转换箭头，可以代表每个节点活动的执行时间和节点之间的转换时间

下图是一个带有MARTE标注的活动图。有一个自定义类型《SaStep》和一个标记值：执行时间和结束时间，分别代表每个活动节点的执行时间，以及活动必须在结束时间之前执行完毕。

AD/MARTE的通用定义是一个带有MARTE标注的结构（(Nad, Aad, Tad, F, MAad, aI, aF）

Nad是活动图的名称

Aad={a1,a2,…,am}是一组活动状态。

Tad={t1,t2,…tn}是一组转换集合

F⊆(Aad×Tad)⋃(Tad×Aad)是流关系。

aI∈Aad 是初始状态, aF∈Aad is 最终状态

References:

1. Wang Q, Wu SJ, Li MS. Software defect prediction. Ruan Jian Xue Bao/Journal of Software, 2008,19(7):15651580 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1565.htm
2. Hall T, Beecham S, Bowes D, Gray D, Counsell S. A systematic literature review on fault prediction performance in software engineering. IEEE Trans. on Software Engineering, 2012,38(6):12761304.
3. Yu SS, Zhou SG, Guan JH. Software engineering data mining: A survey. Journal of Frontiers of Computer Science and Technology, 2012,6(1):131 (in Chinese with English abstract).
4. Radjenovic D, Hericko M, Torkar R, Zivkovic A. Software fault prediction metrics: A systematic literature review. Information and Software Technology, 2013,55(8):13971418.
5. Akiyama F. An example of software system debugging. In: Proc. of the IFIP Congress. 1971. 353359.
6. Halstead MH. Elements of Software Science (Operating and Programming Systems Series). New York: Elsevier Science Inc., 1977.
7. McCabe TJ. A complexity measure. IEEE Trans. on Software Engineering, 1976,2(4):308320.
8. Chidamber SR, Kemerer CF. A metrics suite for object oriented design. IEEE Trans. on Software Engineering, 1994,20(6): 476493.
9. Basili VR, Briand LC, Melo WL. A validation of object-oriented design metrics as quality indicators. IEEE Trans. on Software Engineering, 1996,22(10):751761.
10. Subramanyam R, Krishnan MS. Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects. IEEE Trans. on Software Engineering, 2003,29(4):297310.
11. Zhou YM, Xu BW, Leung H. On the ability of complexity metrics to predict fault-prone classes in object-oriented systems. Journal of Systems and Software, 2010,83(4):660674.
12. Zhou YM, Leung H, Xu BW. Examining the potentially confounding effect of class size on the associations between object- oriented metrics and change-proneness. IEEE Trans. on Software Engineering, 2009,35(5):607623.
13. Zhou YM, Xu BW, Leung H, Chen L. An in-depth study of the potentially confounding effect of class size in fault prediction. ACM Trans. on Software Engineering and Methodology, 2014,23(1):10:110:51.
14. Zhao YY, Yang YB, Lu HM, Zhou YM, Song QB, Xu BW. An empirical analysis of package-modularization metrics: Implications for software fault-proneness. Information and Software Technology, 2015,57:186203.
15. Yang YB, Zhou YM, Lu HM, Chen L, Chen ZY, Xu BW, Leung H, Zhang ZY. Are slice-based cohesion metrics actually useful in effort-aware post-release fault-proneness prediction? an empirical study. IEEE Trans. on Software Engineering, 2015,41(4): 331357.
16. Sarkar S, Kak AC, Rama GM. Metrics for measuring the quality of modularization of large-scale object-oriented software. IEEE Trans. on Software Engineering, 2008,34(5):700720.
17. Meyers TM, Binkley D. An empirical study of slice-based cohesion and coupling metrics. ACM Trans. on Software Engineering and Methodology, 2007,17(1):2:127.

附中文参考文献:

[1] 王青,伍书剑,李明树.软件缺陷预测技术.软件学报,2008,19(7):15651580. http://www.jos.org.cn/1000-9825/19/1565.htm

[3] 郁抒思,周水庚,关佶红.软件工程数据挖掘研究进展.计算机科学与探索,2012,6(1):131.

1. [↑](#footnote-ref-1)