**文档编号： TMD\_IMP**

**版本号：V0.1**

系统与软件工程 基于模型的系统工程

统一架构建模语言

System and software engineering—Model-based systems engineering—Unified architecture modeling language

**北京中科蜂巢科技有限公司**

|  |  |  |  |
| --- | --- | --- | --- |
| **编制** | 鲁金直 | **生效日期** | 2018-03-13 |
| **审核** |  | **批准** |  |

目次

[前言 II](#_Toc172743692)

[引言 III](#_Toc172743693)

[1 范围 5](#_Toc172743694)

[2 规范性引用文件 5](#_Toc172743695)

[3 术语和定义 5](#_Toc172743696)

[4 统一架构建模语言总体框架 10](#_Toc172743697)

[5 关键字词汇结构 11](#_Toc172743698)

[6 类语法结构 17](#_Toc172743699)

[附录A（资料性） 架构语言建模示例 36](#_Toc172743700)

[附录B（资料性） 飞机机载维护与健康管理系统示例 43](#_Toc172743701)

[附录C（资料性） 单位 1](#_Toc172743702)

1. 前言

本文件按照GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

本文件由全国信息技术标准化技术委员会（SAC/TC 28）提出并归口。

本文件起草单位：

本文件主要起草人：

1. 引言

随着装备研制过程的复杂性逐渐增加，基于模型的系统工程（MBSE）广泛应用于各装备研发过程中的项目管理、总体设计、需求分析、架构设计等活动，提高研制的效率及效能。在面向复杂装备的正向设计过程中，不同层级架构设计需要不同的建模语言，如系统之系统层级使用UPDM建模语言，系统层级使用SysML建模语言，子系统层级使用EAST-ADL建模语言，不同建模语言的语义及存储结构不同，导致语言的集成和交互困难。

因此本文件定义统一架构建模语言将复杂装备的正向设计过程中的多种建模语言进行统一的架构视图描述与表达，实现多种建模语言的集成、数据的互通和一致性传递。本文件通过统一架构建模语言的元元模型类、元模型类及模型类等语法定义，明确语言的语法结构，支持基于统一建模语言的工具开发及模型集成。

本文件是在元对象机制（MOF）框架的基础上形成，图 1 展示统一架构建模语言的构建的四个层次的：M3层（元元模型层）、M2层（元模型层）、M1层（模型层）、M0层（现实世界）。M3至M0是逐层实例化的过程，M0至M3是逐层抽象的过程。



1. 基于MOF的建模框架

图1中各层级的主要内容如下。

1. M3为元元模型层，代表最高程度的元模型抽象，即元模型的元语言。本质上元模型和元元模型没有区别，只是在元对象机制中的抽象水平不同。通过这种方式，设计人员按照需求解除元模型的限制——利用元元模型的组合自行构建多领域元模型库，实现不同模型的抽象与交互。在元对象机制M3层含义的基础上，统一架构建模语言提供了六种元元模型以及其之间的规则和约束，支持描述更多领域元模型之间的关联和约束关系。这六种元元模型分为属性元素和非属性元素两类：前者指属性元元模型，用来描述非属性元素的性质；而后者包括图元元模型、对象元元模型、点元元模型、角色元元模型、关系元元模型等。扩展概念用于指定六种元元模型之间的关联关系，并说明其规则和约束。这些扩展概念包括分解、剖视、类引用等。
2. M2为元模型层，是对元元模型的实例化，基于元对象机制的M2层框架，本文件的M2层包括图元模型、对象元模型、点元模型、角色元模型、关系元模型、属性元模型以及元模型b扩展，通过元模型的组合形成建模语言。这些元模型通过图形化的方式为系统设计提供建模手段。
3. M1为模型层，模型是对元模型的实例化，是通常意义上的建模过程，用于描述系统生存周期的活动。
4. M0是对模型层在现实世界中的真实表征，即从某个系统视角对关注的问题进行的描述。

本文件基于元对象机制的M3-M1框架中指定六种元元模型和扩展，通过定义元元模型、元模型和模型的类，实现统一架构建模语言的规范定义。

系统与软件工程 基于模型的系统工程

统一架构建模语言

* 1. 范围

本文件确立了统一架构建模语言的总体框架，规定了关键字词汇结构和类语法结构。

本文件适用于在系统之系统、系统、子系统等层级，需求、功能、逻辑等架构设计阶段开展的架构视图构建过程中的统一描述，支持多种建模工具间的数据交互及语义集成。

* 1. 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB 3100 国标单位制及其应用

GB/T 3101 有关量、单位和符号的一般原则

GB/T 3102.1 空间和时间的量和单位

GB/T 3102.2 周期及其有关现象的量和单位

GB/T 3102.3 力学的量和单位

GB/T 3102.4 热学的量和单位

GB/T 3102.5 电学和磁学的量和单位

CB/T 3102.6 光及有关电磁辐射的量和单位

GB/T 3102.7 声学的量和单位

GB/T 3102.8 物理化学和分子物理学的量和单位

GB/T 3102.9 原子物理学和核物理学的量和单位

GB/T 3102.10 核反应和电离辐射的量和单位

GB/T 3102.11 物理科学和技术中使用的数学符号

GB/T 3102.12 特征数

GB/T 3102.13 固体物理学的量和单位

GB/T 11457 信息技术 软件工程术语

GB/T 13000 信息技术 通用多八位编码字符集（UCS）

* 1. 术语和定义
     1. 术语和定义

GB/T 11457界定的以及下列术语和定义适用于本文件。



架构 architecture

实体在其环境中的基本概念或属性，以及该实体及其相关生存周期过程的实现和演化的管控原则。

[来源：ISO/IEC/IEEE 42010:2022, 3.1]

类 class

用于将数据和操作的方法封装在一起，表示问题域的抽象概念。

属性类引用 class reference

用于在模型中通过属性引用对象、关系、角色、点、属性等模型，类引用通常在属性元模型中定义。

元元模型 meta-meta model

定义元模型的元数据结构和语义，是元模型进一步抽象。

1. 元元模型与元模型之间的关系，类同于元模型与模型间的关系。
2. 本文件包含图元元模型、对象元元模型、关系元元模型、点元元模型、角色元元模型、属性元元模型，以及扩展。

[来源: GB/T 28174.1:2011,3.1.73]

元模型 meta-model

一种定义用于表达模型的语言的模型。

1. 元模型的典型作用，是定义模型元素在模型中如何得到实例化的语义。模型通常包含模型元素。这些模型元素是通过将元模型的模型元素（即元模型元素）实例化而创建的。

[来源: GB/T 28174.1:2011,3.1.73]

图元模型 meta-model of graph

模型的基础视图要素。

注：图元模型定义了在模型中使用的对象、关系元模型等语法和语义规则。

对象元模型 meta-model of object

模型中需要使用的主要建模对象的元模型。

关系元模型 meta-model of relationship

模型中需要使用的将建模对象建立关联关系的连接实体的元模型。

点元模型 meta-model of point

模型中对建模对象附加的端口实体的元模型。

属性元模型 meta-model of property

模型中视图、要素、关系、端口等要素基础参数的元模型，提供在模型中各要素的基础参数。

角色元模型 meta-model of role

模型中定义关系实体两端输入输出性质的元模型。

元模型映射 meta-model mapping

两类元模型要素之间的映射，支撑采用元模型定义的模型之间的转换。

模型 model

实体或实体集合的抽象表征，提供了在所关注的条件或情况下，描绘、理解或预测实体或实体集合属性或特征的能力。

1. 模型准许使用形式理论，而形式理论基于数学或者科学的原理与概念。模型准许用已确立的元模型生成。元模型通常用于促进开发准确、完整、一致且可理解的模型。
2. 参考模型用于刻画一般实例，该一般实例用作具体条件或情况创建特殊实例模型的基础。参考模型能用于促进和加强架构与架构元素的一致性。
3. 根据具体情况，模型准许是架构模型、架构实体模型、概念模型或者参考模型。
4. 物理模型是一种具体表示，它和数学与逻辑模型相区别，后两种模型都是系统的更抽象表示。抽象模型还能进一步分为描述模型（类似于逻辑模型）和分析模型（类似于数学模型）。
5. （建模和仿真）模型是对现实世界过程、概念或系统的选定方面的近似、表示或理想化，这些选定方面包括结构、行为、运行或其他特征。模型准许以其他模型作为组件。

[来源: ISO/IEC/IEEE 42020:2019, 3.13, 有修改]

模型实例 model instance

基于图元模型指定的图形化视图要素的集合。

对象实例 object instance

模型中用来建模的要素，是基于对象元模型的实例化。



关系实例 relationship instance

模型中用来建立连接关系的要素，是基于关系元模型的实例化。



点实例 point instance

模型中用来指定对象端口的要素，是基于点元模型的实例化。



属性实例 property instance

模型中用来指定各要素参数信息的要素，是基于属性元模型的实例化出。

角色实例 role instance

模型中用来规定关系流方向的要素，是基于角色元模型的实例化。

建模语言 modeling language

按照一套系统的规则和框架来设计和构建结构和模型的图形或文本计算机语言。

1. 建模语言与编程语言不同，一般来说，编程语言抽象层次较低，采用面向计算机实现的概念作为建模元素，语言的可执行性强；建模语言抽象层次高，采用面向问题领域的概念作为建模元素，语言的可执行性弱。编程语言更侧重于技术域的计算机实现，其语法、语义的研究对象实际上是计算机程序；而建模语言的语法和语义则更针对问题域的研究对象描述。

基于模型的系统工程 model-based systems engineering

建模方法的形式化应用，以支持系统从概念设计阶段一直持续到开发阶段和后续生存周期阶段的需求、设计、分析、验证和确认活动。

模型元素 model element

由单个组件、动作、状态、消息、属性、关系构成的原子（基本）项，或其他描述系统的组成、特性或行为的其他项。

阶段 stage

生存周期中的时间段，在这段时间内为实现本段时间的目标而执行的活动。

[来源: ISO/IEC/IEEE 42020:2019, 3.15]

系统 system

部件或元素的组合体，组合后表现出单个部件不具备的既定的行为或意义。

1. 一个系统被认为是一种产品或者是一种它所提供的服务。
2. 实际中，对系统含义的解释通常通过使用一个联合名词来阐述，例如飞行器系统。有时候 “系统”也准许简单地由依赖语境的同义词来替代，例如飞行器，虽然会使系统的角度不太明显。
3. 完整的系统包括相关装置、设备、原料、计算机程序、固件、技术文档、服务和运行、支持所必需的人力，在预期的环境中实现某种程度的自给自足。

[来源: ISO/IEC/IEEE 15288:2023, 3.46]

系统元素 system element

满足特定需求的一个系统的离散部分。

1. 硬件、软件、数据、人、过程（例如为用户提供服务的过程）、规程（例如操作指南）、设备、材料、自然存在的实体或其任意组合。

[来源: ISO/IEC/IEEE 15288:2023, 3.47]

统一架构建模语言 unified architecture modeling language

用于支持基于模型的系统工程架构建模的可读文本式及语义式语言。

1. 采用一种统一的语义式建模语言及技术实现复杂系统的全周期、全要素、全领域架构描述及表达的方法。
2. 基于“文本可读的语义式形式化语言”的核心理念对复杂系统的需求、功能、逻辑、物理结构等系统工程架构视点进行建模。借助统一架构建模语言支持复杂系统系统之系统、系统、子系统等不同层级的架构视点统一语义表达，如任务定义、系统之系统结构定义、系统需求、系统功能、系统逻辑、系统物理结构等。统一架构建模语言实现对元模型及模型的形式化描述，是一种面向对象的可读的文本语言，有关模型的所有信息及数据都被形式化在统一架构建模语言中，包括元元模型、元模型以及模型。统一架构建模语言形式化模型支持面向对象的语法，例如“继承”、“封装”等。
3. 具有下列主要能力：

a）支持不同图形化架构建模语言及框架的元模型开发及建模，如UML、SysML、BPMN、UPDM等；

b）支持基于特定域建模语言的元模型开发及建模，如EAST-ADL等。

1. 具有以下主要特点：

a）以可读文本的形式化语言表达MBSE模型，使语义描述更加清晰；

b）支持元模型设计，准许基于统一架构建模语言进行图元模型构建并编译；

c）支持基于元模型的架构建模。



视图 view

从单个视点的角度对系统的表示。

视点 viewpoint

从不同建模角度和利益相关方关注角度出发，为创建、解释和使用模型视图而建立的一组约定，以框定一个或多个关注点。

1. 一个视点包含多个视图模型类型。

[来源: ISO/IEC/IEEE 42010:2022, 3.8,有修改]

* + 1. 缩略语

下列缩略语适用于本文件。

BPMN：业务过程模型和符号（Business Process Model and Notation）

EAST-ADL：电子体系架构与软件技术-架构描述语言（Electronics Architecture and Software Technology- Architecture Description Language）

ID：唯一编码(Identity document)

MBSE：基于模型的系统工程（Model-Based Systems Engineering）

MOF：元对象机制（Meta Object Facility）

SE：系统工程（Systems Engineering）

SysML：系统建模语言（Systems Modeling Language）

UML：统一建模语言（Unified Modeling Language）

UPDM：DoDAF/MODAF的统一配置文件（The Unified Profile for DoDAF/MODAF）

* 1. 统一架构建模语言总体框架

统一架构建模语言应包含元元模型类集合、元模型类集合、模型类集合、组织类集合，可包含插件类集合，见图2。各类集合的定义应使用第5章中的关键字。其中：

1. 元元模型类集合应包含图元元模型类、对象元元模型类、关系元元模型类、点元元模型类、属性元元模型类、角色元元模型类和扩展；
2. 元模型类集合根据所需构建建模语言的视图内容，应包含图元模型类，可包含对象元模型类、关系元模型类、点元模型类、属性元模型类、角色元模型类，以及元模型扩展子集合；当定义关系元模型类时，应同时定义角色元模型类；当定义对象元模型类时，可定义点元模型类；当未定义对象元模型类时，不应定义点元模型类；不应在建模语言中只定义角色元模型类、点元模型类和属性元模型类；
3. 模型类集合应包含图模型实例类，根据元元模型的组合及建模情况，应包含图模型实例类，可包含对象实例类、关系实例类、点实例类、属性实例类、点实例类，以及模型扩展子集合；
4. 组织类集合是分不同层次，对模型进行组织的一类集合，应包含项目类、语言类和包类；
5. 插件类集合是除了元模型之外能够单独构建的一类元素，用于扩展构建模型的能力，可包含甘特图类、矩阵类、布局类。



1. 统一架构建模语言总体框架

各类的语法要求见第6章。通用的架构建模语言示例见附录A。针对飞机机载维护与健康管理系统，通过构建机载维护系统特定域元模型库，即机载维护系统特定域建模语言，并通过使命分析、运行分析、功能分析、逻辑架构分析以及物理架构分析进行流程建模，形成的飞机机载维护与健康管理系统示例见附录B。

属性中涉及到单位应依据GB 3100-1993、GB/T 3101-1993、GB/T 3102.1-1993、GB/T 3102.2-1993、GB/T 3102.3-1993、GB/T 3102.4-1993、GB/T 3102.5-1993、GB/T 3102.6-1993、GB/T 3102.7-1993、GB/T 3102.8-1993、GB/T 3102.9-1993、GB/T 3102.10-1993、GB/T 3102.11-1993、GB/T 3102.12-1993、GB/T 3102.13-1993的要求。常用的单位见附录C。

* 1. 关键字词汇结构
     1. 概述

本章规定了统一架构建模语言语法的词汇结构，包含字符流关键字、字符常量等内容。

1. 统一架构建模语言的最小组成单元是单个字符，字符被组合成词汇单位，称为字符流。字符流包含文字常量和标识符。
2. 这些字符流会被统一架构建模语言翻译器检测。
3. 字符流被组合为统一架构建模语言的语法。
   * 1. 字符编码

统一架构建模语言的字符编码应按GB/T 13000规定选用。

* + 1. 注释

注释并不是真正的词汇单位，是统一架构建模语言的解释和说明。注释在被词法分析器检测到之后会被忽略，不会被编译。统一架构建模语言中包含两种注释方法，见表1。

1. 注释方法

| 注释方式 | 解释 |
| --- | --- |
| // 注释 | 双斜杠//后到这一行结尾之间的字符均被忽略 |
| /\* 注释 \*/ | /\* 和 \*/之间的字符均被忽略 |

统一架构建模语言的注释不准许嵌套，也就是说，/\* 和 \*/之间不准许再嵌套一个/\* 注释 \*/。示例是不准许的。

1. /\* 注释不准许被嵌套/\* 该模块用于模型解释\*/
   * 1. 标识符

标识符是字母、数字和下划线组成的字符序列，用于命名统一架构建模语言中的类名称、变量名称、常数名称等其他项目的名称。标识符应以字母或者下划线开头，后续字符准许为字母、下划线或数字。标识符定义规则见表2。

1. 标识符

| 类别 | 组成 |
| --- | --- |
| IDENT | NONDIGITAL(DIGITAL|NONDIGITAL)\* |
| DIGITAL | 0~9 |
| NONDIGITAL | a~z|A~Z| ”\_” |

* + 1. 字符的关键字

统一架构建模语言字符流中的关键字不应作为标识符命名统一架构建模语言的任何要素，统一架构建模语言字符流中包括的关键字见表3。

1. 关键字及定义

|  |  |  |  |
| --- | --- | --- | --- |
| 关键字 | | 含义 | |
| annotation | | 属性注解 | |
| Array | | 数组 | |
| avoidObstructionsRouting | | 避开障碍 | |
| bind | | 表示绑定约束关系 | |
| blankPropertyMode | | 对象的显示模式——不显示属性 | |
| Boolean | | 布尔型 | |
| Cell | | 元胞数组 | |
| circle | | 固定语法（形状）——圆 | |
| class<Graph> | | 图类型 | |
| class<Object> | | 对象类型 | |
| class<Relationship> | | 关系类型 | |
| class<Role> | | 角色类型 | |
| class<Point> | | 点类型 | |
| class<Property> | | 属性类型 | |
| closestDistanceRouting | | 线的最短距离 | |
| Collection | | 集合 | |
| color | | 定义颜色 | |
| Constraint | | 约束 | |
| Connect | | 定义连接器 | |
| connection | | 链接 | |
| connector | | 标识连接器 | |
| containerMode | | 对象的显示模式——容器模式 | |
| content | | 标识文本内容 | |
| customImage | | 固定语法（形状）——自定义图片 | |
| decompose | | 对象/点/关系/角色的分解 | |
| description | | 描述 | |
| diagramKind | | 声明图的类型 | |
| Distribution | | 分布 | |
| Double | | 双精度浮点数 | |
| end | | 标识结束 | |
| endLocation | | 关系的端点坐标 | |
| Enumeration | | 枚举 | |
| explode | | 对象/点/关系/角色的剖视 | |
| Export | | 角色方向——终端 | |
| extends | | 继承 | |
| false | | 布尔型——假 | |
| frame | | 对象边框的设置 | |
| Float | | 单精度浮点数 | |
| GanttChart | | 标识甘特图插件 | |
| Graph | | 标识图元元模型 | |
| icon | | 图标 | |
| id | | 唯一编码 | |
| imageAddress | | 对象的图片地址 | |
| import | | 角色方向-始端 | |
| initial | | 对元模型属性赋初始值的标志 | |
| initialLocation | | 定义水平/垂直位置和大小 | |
| Input | | 点方向——输入 | |
| instanceOf | | 图元模型到模型的实例化 | |
| Int | | 整型 | |
| jumpLinkStatus | | 跳转链接状态 | |
| jumpLinkType | | 跳转链接类型 | |
| labelDisplay | | 标签展示 | |
| Language | | 标识语言 | |
| Layout | | 标识布局插件 | |
| List | | 有序列表 | |
| localLabel | | 本地名称 | |
| Map | | 键值对 | |
| Matrix | | 矩阵 | |
| metaGraph | | 组织图元模型文件夹名 | |
| metaObject | | 组织对象元模型文件夹名 | |
| metaPoint | | 组织点元模型文件夹名 | |
| metaProperty | | 组织属性元模型文件夹名 | |
| metaRelationship | | 组织关系元模型文件夹名 | |
| metaRole | | 组织角色元模型文件夹名 | |
| Model | | 标识模型 | |
| modelElementName | | 声明模型元素名称 | |
| modelElementType | | 声明模型元素类型 | |
| Node | | 标识笔记 | |
| nodeAttachment | | 标识笔记附件 | |
| Object | | 标识对象元元模型 | |
| OrderSet | | 有序不重复列表 | |
| Output | | 点方向——输出 | |
| packageDiagram | | 固定语法（形状）——包型 | |
| Point | | 标识点元元模型 | |
| polylineLocation | | 关系上各折点坐标 | |
| Project | | 标识项目 | |
| Property | | 标识属性元元模型 | |
| propertyMode | | 对象的显示模式——显示属性 | |
| Real | | 实数 | |
| rectangle | | 固定语法（形状）——矩形 | |
| refer | | 标识元模型实例化 | |
| Relationship | | 标识关系元模型 | |
| reverseJumpLink | | 反向跳转链接 | |
| Role | | 标识角色元元模型 | |
| roundedRectangle | | 固定语法（形状）——圆角矩形 | |
| routingType | | 关系的线型 | |
| Set | | 无序不重复列表 | |
| SDIdentification | | 时序图的唯一编码 | |
| shape | | 固定语法（形状） | |
| smoothness | | 线的平滑度 | |
| strikeThrough | | 删除线 | |
| String | | 字符串 | |
| Table | | 标识表单插件 | |
| tableType | | 表单类型 | |
| text | | 标识对象文本字体设置 | |
| Text | | 标识文本 | |
| time | | 时间，包含时间单位、时间间隔、起始时间、终止时间 | |
| this | | 指代 | |
| triangle | | 固定语法（形状）——三角箭头 | |
| type | | 图的类型 | |
| true | | 布尔型——真 | |
| Tuple | | 元组 | |
| unique | | 唯一性 | |
| unit | | 单位 | |
| underline | | 下划线 | |
| Undirected | | 无向的点 | |
| UnfoldDirection | | 方向 | |
| value | | 表单的值（以矩阵形式表示） | |
| visualizationList | | 分解、剖视图位置 | |
| visualizedMode | | 显示模式 | |
| width | | 宽度 | |
| X | | 方向——横向 | |
| Y | | 方向——纵向 | |
| 1. 关键词对大小写敏感，区分大小写。 2. 括号内为定义内容的辅助说明。 | | | |

* + 1. 字符常量
       1. 实数型字符

实数型字符应在字符中带有小数点的十进制的数字序列。实数的小数点之后至少带有一个数字。实数准许表示15位小数。

1. 3.0、322.56。
   * + 1. 整数型字符

整数型字符应在字符中十进制的数字序列。整型的最小取值为-231，最大取值为231-1。

1. 33、0、30044。
   * + 1. 布尔型字符

布尔型字符应取值范围为真或假的枚举类型。

* + - 1. 字符串

字符串一般应出现在双引号之间，如“字符串”。除了双引号和反斜杠以外，都应准许直接包括在字符串中，无需使用转义码。表4中字符不应作为字符串。

1. 非字符串字符

|  |  |
| --- | --- |
| 字符 | 含义 |
| \’ | 单引号 |
| \” | 双引号 |
| \\ | 反斜杠 |
| \n | 换行 |
| \r | 回车 |
| \t | 水平标签 |

* + - 1. 单精度浮点数型字符

单精度浮点数型字符应由符号位、指数位和尾数位共32位组成，其中指数位占 8 位，尾数位占 23 位，提供约7位十进制数的精度。

1. Float myFloat = 3.14f。
   * + 1. 双精度浮点数型字符

双精度浮点数型字符应由符号位、指数位和尾数位共 64 位组成，其中指数位占 11 位，尾数位占 52 位，提供约15位十进制数的高精度。

1. Double myDouble = 3.14。
   * + 1. 元胞数组

元胞数组应由元胞组成的数组，准许是一维、二维或多维。不同于普通的数组，其中的元素准许是不同类型的数据。

1. language1 = {SysML,OMG}; language2 = {UML,OMG}; languageArray = {language1,language2};则languageArray为元胞数组，元胞存储了语言名称和提出组织的信息。
   * + 1. 枚举

枚举应是一种数据类型，其实例是命名值的列表。

1. enum Day { SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY}。
   * + 1. 数组

数组应是一种用于存储相同类型元素的数据结构，这些元素通过索引进行访问。提供了一种有效的方式来组织和管理大量相似类型的数据。

1. String[] fruits = new String[]{"Apple", "Banana", "Orange", "Grapes"}。
   * + 1. 矩阵

矩阵应是一个由数值按照规定的规则排列成的矩形数组。

1. [1,2,3]。
   * + 1. 元组

元组应是一个有序的、不可变的数据序列。包含不同数据类型的元素，且每个元素在元组中有固定的位置。元组的长度是固定的，一旦创建，就不准许修改。

1. my\_tuple = (1, "hello", 3.14, True)。
   * + 1. 集合

集合应是一个无序的、不包含重复元素的数据结构。通常用于表示一组独特的元素，而不关心元素的顺序。

* + - 1. 分布

分布应是一组数据或事件在不同取值或位置上的分散程度、频率或概率分布。

* + - 1. 有序列表

有序列表应是按照一定的顺序（通常是数字或字母）对列表项进行编号。

* + - 1. 无序不重复列表

无序不重复列表应是一个无序的、不包含重复元素的数据结构。通常用于表示一组独特的元素，而不关心元素的顺序。

* + - 1. 有序不重复列表

有序不重复列表应在有序性中的元素按照一定的顺序排列，列表中的元素不应重复。

1. unique\_set = {1, 2, 3, 4, 5}。
   * + 1. 键值对

键值对应是一种数据存储和表示方式，其中数据被组织成一对（或多对）键和相应的值。每个键与其关联的值之间存在映射关系，准许通过键来访问对应的值。

* + - 1. 图类型

图类型应是限定模型中属性引用的数据类型为图模型实例。

* + - 1. 对象类型

对象类型应是限定模型中属性引用的数据类型为对象实例。

* + - 1. 关系类型

关系类型应是限定模型中属性引用的数据类型为关系实例。

* + - 1. 角色类型

角色类型应是限定模型中属性引用的数据类型为角色实例。

* + - 1. 点类型

点类型应是限定模型中属性引用的数据类型为点实例。

* + - 1. 属性类型

属性类型应是限定模型中属性引用的数据类型为属性实例。

* 1. 类语法结构
     1. 元元模型类集合

元元模型类集合包含图、对象、关系、点、属性、角色和扩展，应按照表5的要求描述统一架构建模语言中的元元模型。

1. 元元模型类集合

|  |  |  |
| --- | --- | --- |
| 元元模型类型 | 语法 | 含义 |
| 图 | Graph | 用于对系统的某个视点进行分析，在一个框图内对系统进行描述。 |
| 对象 | Object | 模型中的主要元素，用于表示一种实体，可以单独存在，也可以与其他对象进行关系链接。 |
| 关系 | Relationship | 通过角色与对象连接，表示对象之间以何种方式进行连接。 |
| 点 | Point | 应附属在对象上，用于表示对象和角色连接的端口。 |
| 属性 | Property | 用于定义及描述一些元类型（对象、角色及关系）的特性。属性不能单独存在，应附属到其他元元模型上用于表示其特性。 |
| 角色 | Role | 应在关系的两端，表示对象以何种方式进行连接。 |
| 扩展 | Extension | 表示分解、剖视、属性类引用等约束的集合。 |

* + 1. 元模型类集合
       1. 图元模型类语法

图元模型类可包含图的属性、对象元模型的声明、关系元模型的声明、约束，以及图元模型的注释共5个部分。其中，图的属性是对属性元模型的继承，约束可包含对象元模型和关系元模型的绑定关系、对象的分解和剖视，注释中应包含名称、描述和图的类型。

图元模型类语法实例如下。

partial Graph diagram1 //diagram1是图元模型的唯一编码。

initial Property unique graphProperty1 = "value" extends height1 annotation (localLabel="重量"); //图的属性中extends是图的属性继承属性元模型使用的关键字；"value"是字符串形式的属性值；"重量"是属性的名称。

Object requirement1; //requirement1是声明的对象元模型的唯一编码。

Object requirement2;

Relationship relat1; //relat1是声明的关系元模型的唯一编码。

Relationship relat1;

Constraint

(

bind connector(requirement1.inPoint1, relat2.satisfyTo), //对象requirement1配置的点inPoint1**绑定**关系relat2的角色satisfyTo。

bind connector(requirement1.outPoint1, relat2.satisfyFrom),

bind connector(requirement2, relat1.deriveFrom),

bind connector(requirement2, relat1.deriveTo),

requirement1.decompose(Diagram1,Diagram2), //对象requirement1可分解为Diagram1和Diagram2。通过“，”+唯一编码的方式增加可分解的图元模型。

requirement2.decompose(Diagram2,Diagram3),

requirement1.explode(Diagram2,Diagram3), //对象requirement1可剖视为Diagram1和Diagram2。通过“，”+唯一编码的方式增加可剖视的图元模型。

relat1.deriveTo.explode(Diagram3),

);

annotation

(

localLabel="图",

description="描述",

type = diagram,

);

end diagram1;

其中，图元模型类关键字的含义见表6。

1. 图元模型类含义

|  |  |
| --- | --- |
| 关键字 | 含义 |
| partial | 图元模型的标识。 |
| Graph | 定义图元模型的关键字。 |
| Object | 在图下的声明对象元模型的关键字。 |
| Relationship | 在图下声明关系元模型的关键字。 |
| Constraint: | 对图的约束定义，包含绑定bind语句以及分解decompose()语句和剖视explode()语句。 |
| bind | 定义图元模型中绑定的关键字。 |
| connector | 确定实现绑定的连接器，通过声明应进行绑定的对象元模型、点元模型、关系元模型、角色元模型，将对象和关系进行绑定。 |
| decompose | 定义分解模式的关键字，分解是指对一个对象（点、关系）模型的展开，将一个对象（点、关系）模型分解为一个图模型进行展示和分析。图元模型定义分解是从元模型层确定对象（点、关系）展开为哪些类的图。 |
| explode | 定义剖视模式的关键字，剖视是指从不同角度对对象（点、关系、角色）模型的内部结构用一个图模型进行展示。图元模型定义剖视是从元模型层确定对象（点、关系、角色）展开为那些类的图。 |
| localLabel | 名称，用来命名图元模型的通用名称，可重复，类比人名，形式为带双引号的字符串。 |
| description | 描述，描述系统工程师定义该属性的意图，形式为带双引号的字符串。 |
| type | 对图元模型类型的定义，不同的图元模型所包含的对象在画布的展示不同。图类型包含：图（diagram），时序图（sequenceDiagram），版本表格（editionTable），交叉表（crossTable），树形（tree）。 |
| 1. 标点符号说明：图元模型属性使用半角分号“;”，Constraint()内部内容以半角逗号“,”隔开，外部使用半角分号“;”，annotation()内部内容以半角逗号“,”隔开，外部使用半角分号“;”，图元模型定义结尾end <对象元模型ID>后用半角分号“;”结束。 | |

* + - 1. 对象元模型类语法

对象元模型类定义一组或一类具有相同特征的对象，是图元模型中的主要元素。对象元模型类可包含对象的属性、对象配置的点，应包含对象元模型的注释。其中，对象的属性是对属性元模型的继承，在注释中应包含名称、描述和对象角标、对象模式和对象形状。

元模型类定义的语法实例如下。

Object requirement1 //requirement1 是对象元模型的唯一编码。

initial Property unique celltestPro = "value" extends celltest annotation (localLabel="身高");

Output Point outPoint1 extends outPoint annotation (localLabel="输出点"); //对象点通过extends继承点元模型。

annotation

(

localLabel="对象requirement1", //对象元模型名称

description="描述", //对象元模型描述

icon="Entity.png", //对象的角标

visualizedMode= property mode, //对象模式

shape="circle", //对象形状

);

end requirement1;

对象元模型类关键字含义见表7。

1. 对象元模型类含义

|  |  |
| --- | --- |
| 关键字 | 含义 |
| Object | 是定义对象元模型的关键字。 |
| Output | 表示输出方向，与始端角色绑定，该位置可替换为Input，表示输入方向，与终端角色绑定，可替换为Undirected，表示没有方向性，准许与始端、终端、无向角色绑定。 |
| Point | 是在对象元模型下声明对象点元模型的关键字。 |
| extends | 元模型部分涉及继承的通用关键字。 |
| annotation | 对象元模型包含五个固定语法：localLabel、description、icon、visualizedMode、shape。 |
| icon | 对象元模型的图标，图标名是png、jpg、gif、svg等格式，图标名显示为带双引号的图标名全称，如“object.gif”。 |
| visualizedMode | 对象元模型的可视化模式定义，可视化模式主要以在对象模型中是否可视化显示对象属性作为区分，可显示属性的定义为property mode，不显示属性的定义为blank property mode |
| shape | 是对象元模型的形状定义，形状的分类受可视化模式约束。在“property mode”模式下，形状包含：圆角矩形（rounded rectangle）,菱形（rhombus）；在“blank property mode”模式下，形状包含：矩形（rectangle）,圆（circle）,菱形（rhombus）,椭圆（oval） |
| 1. 标点符号说明：annotation()内部内容以半角逗号“,”隔开，外部使用半角分号“;”，对象元模型属性使用半角分号“;”，对象点使用半角分号“;”，对象元模型定义结尾 end+对象元模型ID 后用半角分号“;”结束。 | |

* + - 1. 关系元模型类语法

关系元模型类应定义对象模块和对象模块之间的连接关系。关系元模型类可包含关系的属性，应包含一个始端角色和一个终端角色的声明，应包含关系元模型的注释。

关系元模型类定义的语法实例如下。

Relationship relat1 //relat1是关系元模型的唯一编码。

initial Property unique relationshipColor extends colors annotation (localLabel="颜色"); //relationshipColor 是关系的属性的唯一编码，colors 是关系的属性引用的属性元模型的唯一编码。

Role deriveTo; //输出角色声明。

Role deriveFrom; //输入角色声明。

annotation

(

localLabel="关系 relat1", //关系名称。

description="描述", //关系元模型描述。

icon="Exchange.png", //关系元模型的角标。

shape="solid", //关系元模型的形状。

);

end relat1;

关系元模型类关键字的含义见表8。

1. 关系元模型类含义

|  |  |
| --- | --- |
| 关键字 | 含义 |
| Relationship | 定义关系元模型的关键字。 |
| Role | 是在关系元模型下声明角色元模型的关键字。 |
| annotation | 关系元模型包含四个固定语法：localLabel、description、icon、shape。 |
| icon | 关系元模型的图标，图标名是png、jpg、gif等格式，图标名显示为带双引号的图标名全称。 |
| shape | 是关系元模型的形状定义，目前shape已有三种，包含：实线 （solid）, 虚线（dash）, 点线（dot）, 点划线（dash dot）。 |
| 1. 标点符号说明：Relationship中的annotation()内部内容以半角逗号“,”隔开，外部使用半角分号“;”，关系属性元模型使用半角分号“;”，声明的角色使用半角分号“;”，关系元模型定义结尾 end+关系元模型ID 后用半角分号“;”结束。 | |

* + - 1. 点元模型类语法

点元模型定义一组或一类端口，表示对象如何通过关系连接提供额外的语义或约束。点元模型应用于限定对象应被连接的关系元模型，其特性包括：

1. 依赖关系：点元模型应准许独自定义，但点模型应依赖对象模型存在，当删除对象模型时，该对象模型上的点模型也一并删除；
2. 多样性：对象模型上应准许添加多个点，且点的方向不受限制；
3. 多重性：对象模型上的每一个点都应准许与多个角色绑定，或者说每一个对象都通过点与多个关系连接。

点元模型类可包含点的属性，应包含点元模型的注释。

点元模型类定义的语法实例如下。

Point outPoint1 //outPoint1是点元模型的唯一编码。

initial Property unique address1 = "value " extends address00 annotation (localLabel="地址"); //点元模型属性

annotation

(

localLabel="点", //点元模型的名称

description="描述", //点元模型的描述

shape="circle"， //点元模型的形状

);

end outPoint1;

点元模型类关键字的含义见表9。

1. 点元模型类含义

|  |  |
| --- | --- |
| 关键字及固有参数 | 含义 |
| Point | 定义点元模型的关键字。 |
| initial | 表示初始化点元模型属性。 |
| Property | 是在点元模型下声明属性元模型的关键字。 |
| unique | 表示唯一性约束，指该属性在模型中只准许被实例化一次，不准许有同源属性。 |
| annotation | 点元模型包含三个固定语法：localLabel、description、shape。 |
| shape | 是点的形状定义，为建模提供多种形状展示，shape可包含：矩形（rectangle），圆（circle），三角形 （triangle）。 |
| 1. 标点符号说明：annotation()内部内容以半角逗号“,”隔开，外部使用半角分号“;”，点元模型属性使用以半角分号“;”,点元模型定义结尾end+点ID后用半角分号“;”结束。 | |

* + - 1. 属性元模型类语法

属性元模型类是其他元模型元素的属性。独立的属性元模型无实际含义，应配置在其他五种元模型使用。属性元模型的核心是定义数据类型，见章节5.6，当属性元模型的数据类型为“Int”、“Real”时，应准许匹配响应的单位，属性中的单位见附录C。

属性元模型定义的语法实例如下。

Int Property address1 annotation (localLabel="属性1",unit = “kg”, description="属性元模型"); //address1 是属性元模型的唯一编码。

属性元模型类关键字的含义见表10。

1. 属性元模型类含义

|  |  |
| --- | --- |
| 关键字及固有参数 | 含义 |
| Property | 是定义属性类的关键字。 |
| annotation | 注释元素的固定语法；不同的元素包含的固定语法不同，属性元模型包含两个固定语法：localLabel、description。 |
| unit | 单位，只有是在Int或者Real类型的时候选择单位，形式为带双引号的字符串。 |
| 1. 标点符号说明：annotation内部内容以半角逗号“,”隔开，属性定义结尾以半角分号“;”结束。 | |

* + - 1. 角色元模型类语法

角色元模型类定义对象元模型如何参与关系元模型。关系元模型应通过角色元模型间接关联对象元模型，关系元模型的一些特征被转移到角色元模型。角色元模型类可包含角色的属性，应包含角色元模型的注释。

角色元模型类定义的语法实例如下。

Export Role deriveTo // deriveTo 是角色元模型的唯一编码。

initial Property unique address111 extends address00 annotation (localLabel="地址"); //角色元模型属性

annotation

(

localLabel="输出角色", //角色元模型名称。

description="角色的描述"， //角色元模型描述。

shape="solid arrow", //角色元模型形状。

);

end deriveTo;

角色类元模型类语法中关键字的含义见表11。

1. 角色元模型类含义

|  |  |
| --- | --- |
| 关键字及固有参数 | 含义 |
| Export | 是定义角色的方向，端口方向有三种：始端（Import），终端（Export），无向端（Undirected）。 |
| Role | 是定义角色元模型的关键字。 |
| annotation | 角色元模型包含三个固定语法：localLabel、description、shape。 |
| Shape | 是角色的形状定义，为建模提供多种形状展示，shape已有三种，包含：实心箭头（solidArrow），虚心箭头（dottedArrow），箭形（arrowShape），实心菱形（solidDiamond）， 虚心菱形（dottedDiamond），无形状（none），圆环（ring），实心圆（solidCircle）。 |
| 1. 标点符号说明：annotation()内部内容以半角逗号“,”隔开，外部使用半角分号“;”，角色元模型属性使用以半角分号“;”,角色元模型定义结尾end+点ID后用半角分号“;”结束。 | |

* + - 1. 元模型扩展子集合
         1. 通则

元模型扩展集合表达模型要素的引用关系，可包含分解、剖视以及类引用，见图3，其中：

1. 分解：对象元模型类、点元模型类、关系元模型类或角色元模型类可分解为1个或者多个图元模型类；
2. 剖视：对象元模型类、点元模型类与关系元模型类可剖视为1个或者多个图元模型类；
3. 类引用：属性元模型类可与图元模型类、对象元模型类、点元模型类、关系元模型类、角色元模型类或者属性元模型类建立类引用关系。



1. 元模型类引用关系
   * + - 1. 分解

展开视图的一种形式，通过一个新的视图展示视图中一个对象(关系、角色、点)内部包含的构造。

元模型中，一个对象(关系、角色、点)可分解出多个新的视图。分解过程应在图元模型中配置，关键词是decompose，定义在Constraint()之中，分解在图元模型的语法见6.2.1节。

* + - * 1. 剖视

展开视图的一种形式，剖视从一个或多个视图的角度对原视图中一个对象(关系、点)的内部构造进行展示。剖视过程应在图元模型中配置，关键词是explode，定义在Constraint()之中，剖视在图元模型的语法见6.2.1节。

* + - * 1. 属性类引用

用于在模型中通过属性引用对象、关系、角色、点、属性等模型，类引用通常在属性元模型中定义。类引用通过关键词class<>实现，包含图类型（class<Graph, ID>）、对象类型（class<Object,ID>）、关系类型（class<Relationship,ID>）、点类型（class<Point,ID>）、角色类型（class<Role,ID>）、属性类型（class<Property,ID>），这里ID是该元素的唯一编码。同样，多个同样类型使用关键词classArray<>实现。在属性元模型中定义对象类型的语法实例如下。

class<Object, obj1> Property property1 annotation (localLabel="属性1",unit = “null”, description="属性元模型"); //obj1是引用的对象类的唯一编码，property1是属性元模型的唯一编码。

classArray<Object,obj1,obj2> Property property1 annotation (localLabel="属性2",unit = “null”, description="属性元模型"); //obj1,obj2为引用的两个对象类的唯一编码，property1是属性元模型的唯一编码。

* + 1. 模型类集合
       1. 图模型实例类语法

图模型实例类是对图元模型的实例化，图模型实例类可包含对象实例类和关系实例类，对象实例类和关系实例类通过连接器进行绑定，并在注释中，可包含甘特图、矩阵、SysML外框等插件类。

以某性能需求图为例，图模型实例类定义的语法如下。

partial Model RequirementDiagram instanceof requirement1 // RequirementDiagram 为模型的唯一编码，requirement1 为该模型实例化的元模型的唯一编码。

*ObjectInstances //需求图中的对象实例类，具体展开在6.3.2节。*

*RelationshipInstances* *//需求图中的关系实例类，具体展开在6.3.3节。*

Connect(

connector(object1, relationship1.role1), //object1是对象实例的唯一编码，relationship1是关系实例的唯一编码、role1是角色的唯一编码。该连接器的含义是对象object1与关系relationship1的角色role1连接在一起。

connector(object1, relationship1.role2)，

......

);//连接器实例

annotation(

localLabel="性能需求图", //图模型命名。

SDIdentification="null",

*plug-inClass //插件类，具体展开在6.5节。*

);

end RequirementDiagram;

图模型实例类关键字的含义见表12。

1. 图模型实例类含义

|  |  |
| --- | --- |
| 关键字 | 含义 |
| Model | 定义图模型的关键字。 |
| instanceof | 专指图模型通过继承图元模型实例化。 |
| connect | 表示对象和对象的连接关系，并通过两个connector()实现。 |
| connector | 表示连接器，可包含对象模型与角色模型绑定的连接器，或点模型与角色模型绑定的连接器。 |
| annotation | 图模型应包含固定语法localLabel，可包含插件类plug-inClass。 |
| plug-inClass | 图中包含的插件包含：甘特图、矩阵、SysML外框，在6.5节具体介绍。 |

* + - 1. 对象实例类语法

对象实例类语法定义一种实例化的对象，对象实例类可包含多个属性实例、点实例、分解、剖视，应包含注释。

对象实例类定义的语法实例如下所示。

Object RequirementBlock refer object\_Requirement //RequirementBlock 是对象实例的唯一编码，object\_Requirement是该对象的元模型的唯一编码。

Property objProperty = "透平入口温度需求" refer objProperty\_Requirement annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",11,Normal,Normal),underline=false,strikeThrough=false); //objProperty 是对象的属性实例的唯一编码，属性值为命名为"透平入口温度需求"的字符串，objProperty\_Requirement是该属性的元模型的唯一编码，关键字具体介绍在6.3.5节。

*PointInstances //对象中的点实例类，具体展开在6.3.4节。*

annotation(

initialLocation=(322,365,201,103),

localLabel="需求",

shape="rectangle",

color="0,0,0;0,0,0;0,0,0;255,245,181;255,245,181",

screenMode=propertyMode,

imageAddress="image.jpg",

text=("Microsoft YaHei UI",11,Normal,Normal),

iconDisplay=true,

frame=(1,"solid"),

visualizationList="",

);

end RequirementBlock;//完成对象实例的创建。

对象实例类关键字的含义见表13。

1. 对象实例类含义

|  |  |
| --- | --- |
| 关键字及固有参数 | 含义 |
| Object | 定义对象模型的关键字。 |
| refer | 模型中表示继承元模型的关键字。 |
| Property | 定义对象模型属性的关键字。 |
| PointInstance | 代表点模型，点模型的展开细节在下问有介绍，这里用于占位。 |
| this.decompose： | 定义该对象的分解图。 |
| this.explode： | 定义该对象的剖视图。 |
| annotation | 注释，对象模型包含的固定语法。 |
| initialLocation | 定义对象在画布的位置及尺寸大小，以对象最外框左上角为该对象的定位点，以画布左上角为坐标（0，0）点，定义对象定位点横坐标及对象定位点纵坐标，对象外框长和对象外框宽确定对象模型的尺寸大小。所有数值均为整数，单位为毫米。 |
| shape | 对象模型的形状与对象元模型形状种类相同，默认为对象元模型形状，根据具体建模要求进行变更。 |
| color | 定义对象的颜色，颜色为对象填充颜色、边框颜色、字体颜色三部分组成，每种颜色由三原色数字代码组成。 |
| screenMode | 显示模式，可以是分解模式、剖视模式、属性模式、容器模式。 |
| imageAddress | 定义对象的背景图，背景图名准许是png、jpg、gif、svg等格式，图标名显示为带双引号的图标名全称。 |
| text | 定义对象中文字的风格，如字体，字号，加粗体，斜体。字体显示为带双引号的字体名称，字号为整数，加粗显示为Bold,不加粗显示为Normal，是斜体显示为Italic，不是斜体显示为空Normal。 |
| iconDisplay | 布尔值，当显示图标时为true，不显示为false。 |
| frame | 对象边框，定义边框大小和线条样式。 |
| visualizationList | 定义分解剖视的视图内容。 |

* + - 1. 关系实例类语法

关系实例类定义一种实例化的关系，可包含关系的属性实例，分解、剖视，应包含两个角色以及注释。

关系实例类定义的语法实例如下所示。

Relationship relationship1 refer metaRelationship //relationship1是关系模型的唯一编码， metaRelationship 是继承的关系元模型的唯一编码。

Property relationship\_property\_1 = "null" refer Relationship\_Oper1 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

*RoleInstances*  //*角色中的点实例类，具体展开在6.3.6节。*

annotation(

localLabel="作战控制流",

shape="dashLine",

endLocation=(179,217;179,399),

polylineLocation="(179,217);(300,400);(179,399)",

color="136,136,136;0,0,0",

labelDisplay=true,

text=("Microsoft YaHei UI",8,Normal,Normal),

smoothness="None",

avoidObstructionsRouting=false,

closestDistanceRouting =false,

jumpLinkStatus="Below",

jumpLinkType="Semicircle",

reverseJumpLink=false,

routingType="Manhattan",

);

关系实例类关键字及固有参数的含义见表14。

1. 关系实例类含义

|  |  |
| --- | --- |
| 关键字 | 含义 |
| Relationship | 定义关系元模型的关键字。 |
| shape | 关系模型的形状与关系元模型形状种类相同，默认为关系元模型形状，准许根据具体建模要求进行变更。 |
| endLocation | 关系两端点位置坐标，分为始点横、纵坐标（179,217），终点横纵坐标（179,399），数据类型为整型，单位为mm。 |
| polylineLocation | 表示关系中的这点数量和坐标，用数字字符串表示，字符串中第一位数字3为关系中折点的数量（包含始端和终端），坐标分别为(179,217)，(300,400)，(179,399)。 |
| color | 关系线的颜色。 |
| labelDisplay | 关系的关系名称显示控制，当值为true时，显示关系名称；当值为false时，不显示关系名称 |
| smoothness | 关系线的平滑度，None为无平滑，Normal为正常的平滑，Less为较少平滑，More为更多平滑。 |
| avoidObstructionsRouting | 布尔值，true为避开障碍，false为不避障。 |
| closestDistanceRouting | 布尔值，true为最短距离，false为关闭最短距离。 |
| jumpLinkStatus | 跳转连接状态，None为无跳转，All为跳转所有连接，Above为向上跳转，Below为向下跳转。 |
| jumpLinkType | 跳转连接类型，Semicircle为半圆型，Square为直接型，Chamfered为倒棱型，Tunnel为隧道型 |
| reverseJumpLink | 布尔值，true为开启反向跳转连接，false为关闭反向跳转连接。 |
| routingType | 布线样式，Manhattan 为斜体，Straight为直线，Tree为树形。 |

* + - 1. 点实例类语法

点实例类定义一种实例化的端口，可包含点的属性实例，分解、剖视，应包含点的注释。

点实例类定义的语法实例如下所示。

Point point1 refer metaPoint1 //point1 是点模型的唯一编码，metaPoint1 是继承的点元模型的唯一编码。

Property point\_property\_1 = "null" refer point\_Oper1 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

annotation(

localLabel="接口点",

initialLocation=(59,-12,22,22),

shape="triangle",

color="0,0,0;0,0,0;0,0,0",

text=("Microsoft YaHei UI",7,Normal,Normal),

iconDisplay=false,

labelDisplay=false,

);

end metaObjPoint\_point\_Point1\_d70e\_57fd;

属性实例类关键字及固有参数的含义见表15。

1. 属性实例类含义

|  |  |
| --- | --- |
| 关键字及固有参数 | 含义 |
| Point | 定义点实例的关键字。 |
| Property | 定义属性实例类的关键字。 |
| annotation | 注释元素的固定语法。 |
| labelLocation | 端口名称在画布中的位置。 |
| initialLocation | 端口的初始位置。 |
| shape | 点模型的形状与点元模型形状种类相同，默认为点元模型形状，准许根据具体建模要求进行变更。 |
| color | 端口的颜色。 |
| iconDisplay | 布尔值，true为展示点的图标，false为不展示点的图标。 |
| labelDisplay | 布尔值，true为展示点的名称，false为不展示点的名称。 |
| 1. 标点符号说明：annotation内部内容以半角逗号“,”隔开，属性定义结尾以半角分号“;”结束。 | |

* + - 1. 角色实例类语法

角色实例类定义一种实例化的角色元，可包含角色的属性、分解，应包含角色的注释。

角色实例类定义的语法实例如下所示。

Role role1 refer role\_combeFrom //role1是角色实例的唯一编码，role\_combeFrom是继承的角色元模型的唯一编码。

annotation(

localLabel="组合始端",

shape="solidDiamond"

);

end role1;

角色实例类关键字的含义见表16。

1. 角色实例类含义

|  |  |
| --- | --- |
| 关键字及固有参数 | 含义 |
| Role | 定义角色元模型的关键字。 |
| refer | 含义与对象模型中相同。 |
| this.decompose | 含义与对象模型中相同。 |
| annotation | 角色模型包含两个固定语法：localLabel、shape。 |
| shape | 定义角色模型的形状，默认情况下与角色元模型定义的形状相同，在角色模型中准许手动进行变更。 |

* + - 1. 模型扩展子集合
         1. 通则

模型扩展集合表达模型要素引用关系，模型实例之间可包含分解、剖视、实例引用、克隆及拷贝，见图4，其中：

1. 分解：对象实例、点实例、关系实例以及角色实例可分解为图模型实例，每个实例应分解为一个图模型实例；
2. 剖视：对象实例、点实例以及关系实例可剖视为一个或多个图模型实例；
3. 类引用：属性实例可引用多个图模型实例、对象实例、点实例、关系实例、角色实例或属性实例。
4. 克隆：对象实例可通过克隆与另一个对象实例保持数据同步，且遵循组合关系。
5. 拷贝：对象实例可通过拷贝产生另一个对象实例，同步拷贝时的数据，且遵循聚合关系。



1. 模型实例类引用关系
   * + - 1. 分解

模型实例中，每一个对象（关系、角色、点）实例实体只应分解为一个模型实例，通过this.decompose()指定分解的目标模型，分解在对象模型应按照以下语法结构描述。

* + - * 1. 剖视

模型实例中，每一个对象（关系、点）实例实体只应剖视为不同元模型对应的多个模型实例，通过this.explode()指定剖视的目标模型，剖视在对象应按照以下语法结构描述。

* + - * 1. 属性类引用

模型中属性引用具体类在属性值中标识应按照以下语法结构描述。

Property pro11 = class<obj1> refer pro1 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property pro21 = "classArray<obj1，obj2，...,objn>" refer pro2 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

* + - * 1. 克隆

将一个模型中的对象克隆至同一个模型或另外一个模型中，克隆体对象保持与源对象属性的同步，当源对象的属性发生变化时，克隆体的属性同步发生变化，反之亦然。克隆应通过关键词**clone**实现引用,包含对象、点及属性的克隆。克隆至另外一个模型的实现应需满足该模型中存在此对象的元模型，即原对象模型所引用的对象元模型在该图元模型之中。克隆在对象模型应按照以下语法结构描述。

Object obj11 clone obj1 //对象obj11克隆自对象obj1。

Property pro11 = "XXX" clone pro1 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false); //属性pro11克隆自属性pro1。

Point point11 clone point1 //点point11克隆自点point1。

......

end point11;

annotation(

......

);

end obj11;

* + - * 1. 拷贝

将一个模型中的对象复制至同一个或另一个模型中，实现快速重构原有对象及属性信息，复制体对象修改后不同步至源对象。拷贝应通过关键字**copy**实现引用,包含对象、点及属性的拷贝。拷贝的实现应需满足该模型中存在此对象的元模型，即原对象模型ID所引用的对象元模型在该图元模型之中。克隆在对象模型应按照以下语法结构描述。

Object obj11 copy obj1 //对象obj11拷贝自对象obj1。

Property pro11 = "XXX" copy pro1 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false); //属性pro11拷贝自属性pro1。

Point point11 copy point1 //点point11拷贝自点point1。

......

end point11;

annotation(

......

);

end obj11;

* + 1. 组织类集合
       1. 项目类

项目类是开始统一架构建模语言建立的根节点，项目类应准许定义多个语言类。项目类定义的语法如下所示。

Project pro1 annotation (localLabel="XXX") //Project为项目类定义的关键字，pro1为定义项目类的唯一编码。

*language //项目可包含多个语言内容。*

......

end pro1

* + - 1. 语言类

语言类是建模语言元模型类及模型类的集合。语言类定义的语法如下所示。

Language lan1 annotation (localLabel="XXX") //Language为语言类定义的关键字，lan1为语言类定义的唯一编码。

*meta-model of graph //语言应包含图元模型的内容。*

*meta-model of object //语言应包含对象元模型的内容。*

*meta-model of relationship //语言应包含关系元模型的内容。*

*meta-model of point //语言应包含点元模型的内容。*

*meta-model of property //语言应包含属性元模型的内容。*

*meta-model of role //语言应包含角色元模型的内容。*

*model //语言应包含模型的内容。*

end lan1

* + - 1. 包类

包类是多个模型的集合，应用于表示由同一项目元模型实例化或相关性较大的模型的集合。语言类中应准许定义多个包类。

在包类中应准许定义多个模型类，应通过添加包类实现模型的需求、功能、逻辑、架构等设计阶段的归类或导航。

包类定义的语法如下所示。

Package pac1 annotation (localLabel="XXX") //Package为包类定义的关键字，pac1为定义包类的唯一编码。

*package //包内可包含多个包内容。*

*model //包内可包含多个模型内容。*

*......*

end pac1

* + 1. 插件类集合
       1. 甘特图类

甘特图类应通过条状图来显示项目、进度和其他时间相关的系统进展的内在关系随着时间进展的情况，甘特图类没有元模型定义，是在每个模型视图中都应准许添加的插件元素。

甘特图类定义的语法实例如下所示。

GanttChart gantt //gantt是定义甘特图的唯一编码。

(

localLabel="甘特图", //甘特图名称。

location=(600,400,300,400), //甘特图位置和大小 。

color="255,255,255;0,0,0;0,0,0", //甘特图背景颜色。

time=("年";2;22;34), //时间单位，时间间隔，起始时间，终止时间。

"task1" //甘特图任务名称。

(

gantt1("甘特图1";30;20;"255,0,255;0,0,0;0,0,0"), //子任务1。

gantt2("甘特图2";60;20;"255,255,0;0,0,0;0,0,0"), //子任务2。

)

"task2" //甘特图任务名称。

(

gantt3("甘特图3";30;20;"255,0,255;0,0,0;0,0,0"), //子任务。

)

),

甘特图类关键字的含义见表17。

1. 甘特图类含义

|  |  |
| --- | --- |
| 关键字及固有参数 | 含义 |
| GanttChart | 定义甘特图的关键字。 |
| localLabel | 同属性元模型中解释。 |
| location | 确定甘特图的位置和大小。 |
| color | 甘特图背景颜色的定义。 |
| time | 定义甘特图的时间单位，有“年”、“月”、“日”、“时”、“分”、“秒”，间隔表示在起始时间和结束时间之间每间隔几个单位进行显示。 |
| 1. 数量说明：GanttChart有0次或多次定义，GanttChart中的子任务0次或多次定义。 2. 标点符号说明：除time及子任务内部使用半角分号“;”，其他的结束位置都使用半角逗号“,”。 | |

* + - 1. 矩阵类

矩阵类是在系统设计中说明关系的一种传统媒介，是使用最少空间说明大量关系最有效的一种方式。

矩阵类定义的语法实例如下所示。

Matrix cellName //cellName是矩阵插件定义唯一编码

(

localLabel="矩阵", //矩阵插件名称

location=(10,0,0,0), //矩阵插件的位置

value = "[1,,3;4,,6;7,,9]", //矩阵内容，准许为数值型或者字符串

),

矩阵类关键字及固有参数的含义见表18。

1. 矩阵类含义

|  |  |
| --- | --- |
| 关键字及固有参数 | 含义 |
| Matrix | 定义矩阵的关键字。 |
| location | 确定矩阵的位置和大小。 |
| value | 定义矩阵的内容，形式为矩阵的字符串，或者是属性的ID。 |
| 1. 数量说明： Matrix有0次或多次定义。 2. 标点符号说明：结束位置都使用半角逗号“,”。 | |

* + - 1. 布局类

布局类是针对SysML语言定义的插件，主要应强调模型的头部信息，应包含四段信息：模型的名称、元模型的类型、模型元素类型、模型元素名称。

布局类定义的语法实例如下所示。

Layout drawing //drawing 为外框定义的唯一编码

(

localLabel="SysML外框", //模型的名称。

diagramKind=bdd, // 元模型的类型。

modelElementType=block, //模型元素的类型。

modelElementName="String", //模型元素的名称。

location=(10,0,0,0) //外框的位置和大小。

),

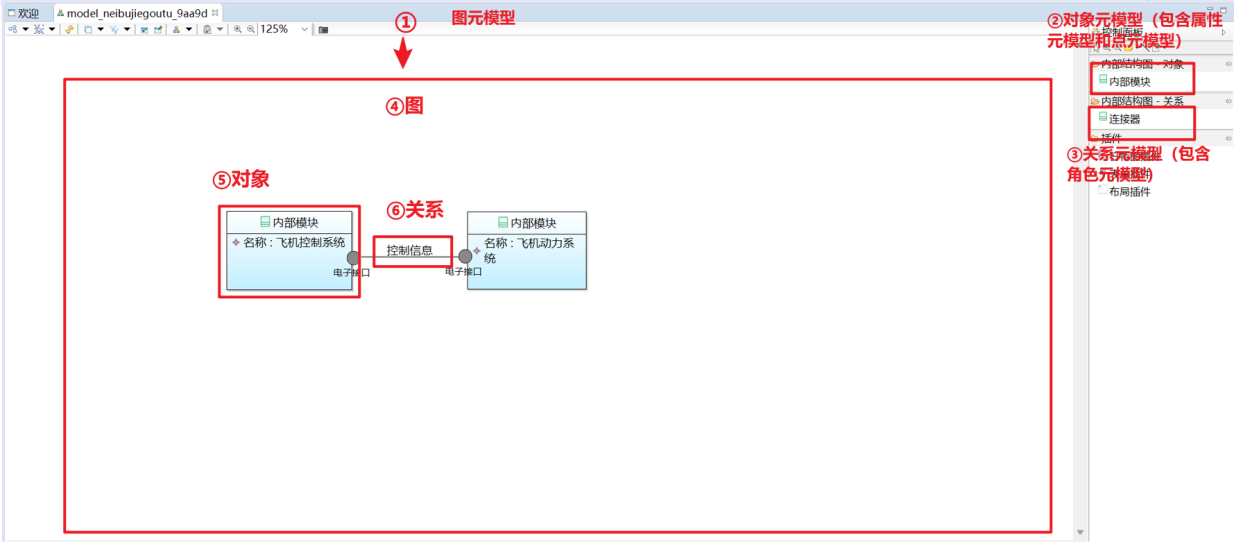
布局类关键字的含义见表19。

1. 布局类含义

|  |  |
| --- | --- |
| 关键字及固有参数 | 含义 |
| Layout | 定义SysML外框插件的关键字。 |
| bdd | 元模型类型包含模块定义图（bdd）、内部模块图（ibd）、用例图（uc）、活动图（act）、序列图（sd）、状态机图（stm）、参数图（par）、需求图（req）、包图（pkg）。 |
| modelElementType | 模型元素类型，每个图都代表已经在系统模型中某处定义的元素，这里声明该元素的类型 |
| modelElementName | 声明该模型元素的名称。 |
| location | 确定SysML外框的位置和大小。 |
| 1. 数量说明：Layout有0次或多次定义。 2. 标点符号说明：结束位置都使用半角逗号“,”。 | |

1. （资料性）  
   架构语言建模示例
   1. 可视化示例

该模型中包含两个名称为内部模块的对象实例，其中一个的属性实例的名称为飞机控制系统，另一个的属性实例的名称为飞机动力系统。对象实例之间的关系实例，名称为控制信息。每个对象实例各包含一个点实例，名称均为电子接口，见图A.1。



* 1. 案例模型图
  2. 元模型示例

元模型包括图元模型、对象元模型、关系元模型、点元模型、属性元模型和角色元模型。

* + 1. 图元模型

图A.1中①，模型对应的图元模型为neibujiegou. 该图元模型中描述了所配置的图元模型ID、对象元模型ID、关系元模型ID、以及关系元模型的绑定关系、图类型、图元模型本地名称、图元模型的描述内容。

partial Graph neibujiegoutu // 图元模型ID为 “neibujiegoutu”。

Object neibumokuai; // 配置的对象元模型ID为“neibumokuai”。

Relationship connectors; // 配置的关系元模型ID为“connectors”。

Constraint ( // 绑定关系。

bind connector(neibumokuai,connectors.Role\_Link\_Input), //绑定始端：对象元模型neibumokuai；绑定终端：与关系元模型connectors绑定的Role\_Link\_Input角色元模型。

bind connector(neibumokuai.neibumokuai\_jiekou\_452c14,connectors.Role\_Link\_Input), //绑定始端：依附于对象元模型neibumokuai的点元模型neibumokuai\_jiekou\_452c14；绑定终端：与关系元模型connectors绑定的Role\_Link\_Input角色元模型。

bind connector(neibumokuai,connectors.Role\_Link\_Output), //绑定始端：对象元模型neibumokuai；绑定终端：与关系元模型connectors绑定的Role\_Link\_Output角色元模型。

bind connector(neibumokuai.neibumokuai\_jiekou\_452c14,connectors.Role\_Link\_Output), //绑定始端：依附于对象元模型neibumokuai的点元模型neibumokuai\_jiekou\_452c14；绑定终端：与关系元模型connectors绑定的Role\_Link\_Output角色元模型。

);

annotation(

localLabel="内部结构图", //图元模型的本地名称为“内部结构图”。

type = diagram, // 图类型为diagram。

description="", //描述内容为空。

);

end neibujiegoutu; //图元模型ID为“neibujiegoutu”。

* + 1. 对象元模型

图A.1中②，模型中包含的对象元模型有neibumokuai,该对象元模型中描述了对象元模型ID等信息，以及配置的属性和点的信息。

Object neibumokuai //对象元模型ID为“neibumokuai”。

initial Property neibumokuai\_env\_Name\_e79f8f = "null" extends env\_Name annotation(localLabel ="名称"); //对象元模型配置的属性元模型ID为“env\_Name”，属性ID为“neibumokuai\_env\_Name\_e79f8f”，属性的本地名称为“名称”，属性的值为“null”。

Undirected Point neibumokuai\_jiekou\_452c14 extends jiekou annotation (localLabel ="接口");//对象元模型配置的点元模型ID为“jiekou”，点ID 为“neibumokuai\_jiekou\_452c14”，点的本地名称为“接口”。

annotation(

localLabel="内部模块", //对象元模型的本地名称为"内部模块"。

description="", //对象元模型的描述为空。

icon="", //对象元模型的图标为默认。

shape="rectangle", //对象元模型的形状为“rectangle”，表示矩形。

visualizedMode=propertyMode, //对象元模型的可视化模式为propertyMode。

imageAddress="", //对象元模型无背景图片。

UnfoldDirection=Y, //在容器模式下支持方向。

);

end neibumokuai; //对象元模型的ID为“neibumokuai”。

* + 1. 关系元模型

图A.1中③，模型中包含的关系元模型有connectors,该关系元模型中描述了关系元模型ID等信息，以及配置的属性和角色的信息。

Relationship connectors //关系元模型ID为“connectors”。

initial Property connectors\_env\_Name\_df6ecb = "null" extends env\_Name annotation(localLabel ="名称"); //关系元模型配置的属性元模型ID为“env\_Name”，属性的ID为“connectors\_env\_Name\_df6ecb”，属性的本地名称为“名称”，属性的值为“null”。

Role Role\_Link\_Input; //关系元模型绑定的角色元模型ID为“Role\_Link\_Input”。

Role Role\_Link\_Output; //关系元模型绑定的角色元模型ID为“Role\_Link\_Output”。

annotation(

localLabel="连接器", //关系元模型的本地名称为“连接器”。

description="", //关系元模型描述为空。

icon="", //关系元模型图标为默认。

shape="boldSolidLine", //关系元模型的形状为“boldSolidLine”。

);

end connectors; //关系元模型的ID为“connectors”。

* + 1. 属性元模型

图A.1中②对象元模型中包含的属性元模型有env\_Name,该属性元模型中描述了属性元模型ID、属性名、属性单位等信息。

String Property env\_Name annotation (localLabel ="名称",unit = "null", description=""); //属性类型为“String”,属性元模型ID为“env\_Name”，属性名为“名称”，属性单位为“null”,属性描述内容为空。

* + 1. 角色元模型

见图A.1中③关系元模型包含的角色元模型有Role\_Link\_Input和Role\_Link\_Output两种,角色元模型中描述了角色元模型ID、本地名称等信息。

Import Role Role\_Link\_Input //角色的方向端口为“始端”，角色元模型ID为“Role\_Link\_Input”。

annotation(

localLabel="链接输入", //角色的本地名称为“链接输入”。

description="", //角色的描述为空。

shape="no", //角色的形状为“no”，表示无形状。

);

end Role\_Link\_Input; //角色元模型ID为“Role\_Link\_Input”。

Export Role Role\_Link\_Output //角色的方向端口为“终端”，角色元模型ID为“Role\_Link\_Output”。

annotation(

localLabel="链接输出", //角色的本地名称为“链接输出”。

description="", //角色的描述为空。

shape="no", //角色的形状为“no”。

);

end Role\_Link\_Output; //角色的元模型ID为“Role\_Link\_Output”。

* + 1. 点元模型

图A.1中②对象元模型包含的点元模型有jiekou，点元模型中描述了点元模型ID、配置的属性、本地名称等信息。

Point jiekou //点元模型的ID为“jiekou”。

initial Property jiekou\_env\_Name\_c69e78 = "null" extends env\_Name annotation(localLabel ="名称"); //点元模型配置的属性元模型ID为“env\_Name”，属性名为“名称”，属性ID为“jiekou\_env\_Name\_c69e78”。

annotation(

localLabel="接口", //点的本地名称为“接口”。

description="", //点的描述为空。

shape="circle", //点的形状为圆形。

imageAddress="", //点无背景图片。

);

end jiekou; //点元模型ID为“jiekou”。

* 1. 模型示例
     1. 图

图A.1中的④表示模型，该模型为图元模型的实例。在下方代码中，“neibujiegoutu”代表图元模型的ID，“model\_neibujiegoutu\_9aa9d”代表模型ID，“内部结构图示例”代表模型的本地名称。

partial Model model\_neibujiegoutu\_9aa9d instanceof neibujiegoutu //模型ID为“model\_neibujiegoutu\_9aa9d”，图元模型ID为“neibujiegoutu” 。

annotation(

localLabel="内部结构图示例", //本地名称为“内部结构图示例”。

);

end model\_neibujiegoutu\_9aa9d; //模型ID为“model\_neibujiegoutu\_9aa9d”。

* + 1. 对象、点、属性

图A.1中的⑤表示对象，图中包含两个“内部模块”对象，每个对象包含一个“名称”属性以及一个“电子接口”点。以其中一个对象为例，下方代码定义了该对象、对象具有的属性、依附于对象的点的相关信息。

Object model\_neibujiegoutu\_9aa9d\_object\_d22d refer neibumokuai //对象元模型ID为“neibumokuai”，对象ID为“model\_neibujiegoutu\_9aa9d\_object\_d22d”。

Property model\_neibujiegoutu\_9aa9d\_object\_d22d\_property\_1fb2 = "飞机控制系统" refer neibumokuai\_env\_Name\_e79f8f annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",8,Normal,Normal),underline=false,strikeThrough=false);

//该对象具有的属性ID为“neibumokuai\_env\_Name\_e79f8f”，实例化后的属性ID为“model\_neibujiegoutu\_9aa9d\_object\_d22d\_property\_1fb2”，属性名为“名称”，属性值为“飞机控制系统”。

Point model\_neibujiegoutu\_9aa9d\_object\_d22d\_undirected\_aa23 refer neibumokuai\_jiekou\_452c14 //该对象配置的点ID为“neibumokuai\_jiekou\_452c14”，实例化后的点ID为“model\_neibujiegoutu\_9aa9d\_object\_d22d\_undirected\_aa23”

Property model\_neibujiegoutu\_9aa9d\_object\_d22d\_undirected\_aa23\_property\_f683 = "null" refer jiekou\_env\_Name\_c69e78 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false); //该点具有的属性ID为“jiekou\_env\_Name\_c69e78”，实例化后的属性ID为“model\_neibujiegoutu\_9aa9d\_object\_d22d\_undirected\_aa23\_property\_f683”，属性名为“名称”，属性值为“null”。

annotation(

localLabel="电子接口", //点的本地名称为“电子接口”。

initialLocation=(168,56,22,22), //点的位置为(168,56,22,22)。

shape="circle", //点的形状为圆形。

color="136,136,136;0,0,0;0,0,0",//点的颜色为“136,136,136;0,0,0;0,0,0”。

text=("Microsoft YaHei UI",7,Normal,Normal),

iconDisplay=false, //是否展示图标：否。

labelDisplay=false, //是否展示标签：否。

imageAddress="", //点无背景照片。

);

end model\_neibujiegoutu\_9aa9d\_object\_d22d\_undirected\_aa23;

annotation(

initialLocation=(308,245,178,113), //对象在模型中的位置为(308,245,178,113)。

localLabel="内部模块", //对象的本地名称为“内部模块”。

shape="rectangle", //对象的形状为矩形。

color="0,0,0;0,0,0;0,0,0;194,239,255;255,255,255", //对象的颜色为“0,0,0;0,0,0;0,0,0;194,239,255;255,255,255”。

screenMode=propertyMode, //对象的显示模式为“显示属性模型”。

imageAddress="", //对象无背景照片。

SDIdentification="",

text=("Microsoft YaHei UI",8,Normal,Normal),

iconDisplay=true, //图标是否展示：“是”。

frame=(1,"solid"), //对象边框设置。

visualizationList="", //无分解剖视。

);

end model\_neibujiegoutu\_9aa9d\_object\_d22d; //对象的ID为“model\_neibujiegoutu\_9aa9d\_object\_d22d”

* + 1. 关系、属性、角色

图A.1中的⑥表示关系。下方代码定义了该关系的ID、关系具有的属性、与关系绑定的角色以及关系两端绑定的模型元素等相关信息。

Relationship model\_neibujiegoutu\_9aa9d\_relationship\_13d9 refer connectors//关系元模型ID为“connectors”，关系ID为“model\_neibujiegoutu\_9aa9d\_relationship\_13d9”。

Property model\_neibujiegoutu\_9aa9d\_relationship\_13d9\_property\_2592 = "null" refer connectors\_env\_Name\_df6ecb annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false); //该关系具有的属性ID为“connectors\_env\_Name\_df6ecb”，实例化后的属性ID为“model\_neibujiegoutu\_9aa9d\_relationship\_13d9\_property\_2592”，属性名为“名称”，属性值为“null”。

Role model\_neibujiegoutu\_9aa9d\_fromrole\_cc74 refer Role\_Link\_Input //该关系绑定的角色元模型ID为“Role\_Link\_Input”，角色ID为“model\_neibujiegoutu\_9aa9d\_fromrole\_cc74”。

annotation(

localLabel="链接输入", //角色的本地名称为“链接输入”。

shape="no" //角色的形状为“no”，表示没有形状。

);

end model\_neibujiegoutu\_9aa9d\_fromrole\_cc74;

Role model\_neibujiegoutu\_9aa9d\_torole\_f876 refer Role\_Link\_Output//该关系绑定的角色元模型ID为“Role\_Link\_Output”，角色ID为“model\_neibujiegoutu\_9aa9d\_torole\_f876”。

annotation(

localLabel="链接输出",//角色的本地名称为“链接输出”。

shape="no" //角色的形状为“no” ，表示没有形状。

);

end model\_neibujiegoutu\_9aa9d\_torole\_f876;

annotation(

localLabel="控制信息", //该关系的本地名称为“控制信息”。

shape="boldSolidLine", //该关系的形状为“boldSolidLine”。

endLocation=(397,301;869,302), //该关系的端点坐标为(397,301;869,302)。

polylineLocation="[0,0,-136,0]$[136,0,0,0];(1.0,0.4);(0.0,0.5)",//该关系的各折点坐标为“[0,0,-136,0]$[136,0,0,0];(1.0,0.4);(0.0,0.5)”。

color="136,136,136;0,0,0", //该关系的颜色为“136,136,136;0,0,0”。

labelDisplay=true, //是否显示标签“是”。

text=("Microsoft YaHei UI",8,Normal,Normal),

smoothness="None", //线的平滑度为“None”。

avoidObstructionsRouting=false, //避开障碍。

closestDistanceRouting =false, //线最短距离。

jumpLinkStatus="None", //跳转链接状态为“None”。

jumpLinkType="Semicircle", //跳转链接类型为“Semicircle”。

reverseJumpLink=false, // 反向跳转链接：否。

routingType="Manhattan", // 关系线型为Manhattan。

);

end model\_neibujiegoutu\_9aa9d\_relationship\_13d9;

Connect(connector(model\_neibujiegoutu\_9aa9d\_object\_d22d.model\_neibujiegoutu\_9aa9d\_object\_d22d\_undirected\_aa23,model\_neibujiegoutu\_9aa9d\_relationship\_13d9.model\_neibujiegoutu\_9aa9d\_fromrole\_cc74),connector(model\_neibujiegoutu\_9aa9d\_object\_956b.model\_neibujiegoutu\_9aa9d\_object\_956b\_undirected\_1714,model\_neibujiegoutu\_9aa9d\_relationship\_13d9.model\_neibujiegoutu\_9aa9d\_torole\_f876));//该关系与两个元素相连接：依附于model\_neibujiegoutu\_9aa9d\_object\_d22d对象的点model\_neibujiegoutu\_9aa9d\_object\_d22d\_undirected\_aa23；依附于model\_neibujiegoutu\_9aa9d\_object\_956b对象的点model\_neibujiegoutu\_9aa9d\_object\_956b\_undirected\_1714。

1. （资料性）  
   飞机机载维护与健康管理系统示例
   1. 建模需求

飞机机载维护与健康管理系统的主要功能是接收飞机机组告警系统的告警信息（CAS消息等）和其他系统计算机内置自检模块记录的故障数据，利用中央维护计算机或者中央维护功能软件将二者关联生成维护报告。维护人员通过维护报告查询手册确定故障源，但是飞机最终是否投入运行一般根据CAS消息和MMECL确定。同时，OMS还为维护人员提供开展地面测试，数据加载，查看构型，监控飞机状态等功能。

飞机机载维护与健康管理系统的主要使用者是航空公司地面维护人员和飞机运行监控人员，使用场景包括飞机制造检测、航线维护、机库维护、飞机运营状态监控、飞机运行数据分析等。系统的另一个潜在使用者是飞行机组，一定条件下准许机组人员查看维护系统采集的飞机健康状态数据以辅助了解飞机运行状态。OMHMS作为维护支持工具，高效地支持航空公司进行开展维护活动或健康监控活动。OMHMS的主要设计依据包括行业标准（如ARINC624-1）、公司规范（CDS2793）、航空公司调研信息和前置型号经验积累等。

图B.1展示的飞机机载维护与健康管理系统由中央维护子系统、飞机状态监控子系统及机载数据加载子系统组成，为飞机提供故障检测与隔离、系统健康数据收集与监控、机上软件或数据库升级更新等维护服务。本案例系统建模分为使命、运行、功能、逻辑和物理进行建模。



* 1. 案例研究对象
  2. 元模型

机载维护系统建模流程见图B.2。

图示

描述已自动生成

* 1. 建模方法论

机载维护系统通过使命分析、运行分析、功能分析、逻辑架构分析以及物理架构分析阶段完成基于模型的架构设计过程，指导后续设计方案的权衡以及仿真验证等工作，自定义的领域元模型库见表B.1。

* 1. 机载维护系统特定域元模型库

|  |  |  |  |
| --- | --- | --- | --- |
| 建模流程 | 图元模型 | 对象元模型 | 关系元模型 |
| 使命分析阶段 | 模块定义图 | 值类型、接口模块、包、视图、约束模块、模块、注释 | 关联、附属、组合、泛化、项目流、约束 |
| 活动图 | 动作概率、开始、节点、发送动作、栓、终止、分流、选择、时间事件、接受事件、动作、注释 | 连接注释、对象流、控制流 |
| 运行分析阶段 | 用例图 | 包、参与者、用例、边界系统、环境、外部系统、传感器、系统上下文 | 包含、关联、扩展、归纳、控制 |
| 活动图 | 动作概率、开始、节点、发送动作、栓、终止、分流、选择、时间事件、接受事件、动作、注释 | 连接注释、对象流、控制流 |
| 功能分析阶段 | 模块定义图 | 值类型、接口模块、包、视图、约束模块、模块、注释 | 关联、附属、组合、泛化、项目流、约束 |
| 活动图 | 动作概率、开始、节点、发送动作、栓、终止、分流、选择、时间事件、接受事件、动作、注释 | 连接注释、对象流、控制流 |
| 逻辑设计阶段 | 序列图 | 生命线、信息 | 信号 |
| 状态机图 | 结束、开始、动作、状态、历史标记、点、注释、叉、发送动作、选择、仿真参数模块、性能模型模块 | 转换 |
| 物理架构设计阶段 | 模块定义图 | 值类型、接口模块、包、视图、约束模块、模块、注释 | 关联、附属、组合、泛化、项目流、约束 |
| 内部模块图 | 值、接口模块、连接器、内部模块、注释、约束 | 附属、传递、对象流、项目流 |

模块定义图的图元模型描述如下，定义2个图属性，声明8个对象元模型，声明6个关系元模型，定义15个绑定关系，定义2个剖视。

partial Graph Block\_Defintion\_Diagram // 定义模块定义图。

initial Property Block\_Defintion\_Diagram\_Property\_To\_Do\_3ac03f = "null" extends Property\_To\_Do annotation(localLabel ="待办事项");

initial Property Block\_Defintion\_Diagram\_Property\_Context\_9794be = "null" extends Property\_Context annotation(localLabel ="上下文");

Object Object\_Package;

Object Object\_Block;

Object Object\_Interface\_Block;

Object Object\_Flow\_Specification;

Object Object\_Constraint\_Block;

Object Object\_Domain;

Object Object\_Subsystem;

Relationship Relationship\_Interface\_Realization;

Relationship Relationship\_Link;

Relationship Relationship\_Association\_Block;

Relationship Relationship\_Directed\_Association;

Relationship Relationship\_Association;

Relationship Relationship\_Directed\_Aggregation;

Constraint (

bind connector(Object\_Block,Relationship\_Interface\_Realization.Role\_Implementing\_Classifier),

bind connector(Object\_System,Relationship\_Interface\_Realization.Role\_Implementing\_Classifier),

bind connector(Object\_System\_Context,Relationship\_Interface\_Realization.Role\_Implementing\_Classifier),

bind connector(Object\_Domain,Relationship\_Interface\_Realization.Role\_Implementing\_Classifier),

bind connector(Object\_Subsystem,Relationship\_Interface\_Realization.Role\_Implementing\_Classifier),

bind connector(Object\_Constraint\_Block,Relationship\_Interface\_Realization.Role\_Implementing\_Classifier),

bind connector(Object\_Interface\_Block,Relationship\_Interface\_Realization.Role\_Implementing\_Classifier),

bind connector(Object\_External,Relationship\_Interface\_Realization.Role\_Implementing\_Classifier),

bind connector(Object\_Flow\_Specification,Relationship\_Interface\_Realization.Role\_Contract),

bind connector(Object\_Interface,Relationship\_Interface\_Realization.Role\_Contract),

bind connector(Object\_Interface,Relationship\_Containment.Role\_Part),

bind connector(Object\_Package,Relationship\_Containment.Role\_Part),

bind connector(Object\_Unit,Relationship\_Containment.Role\_Part),

bind connector(Object\_Block,Relationship\_Containment.Role\_Whole),

bind connector(Object\_System,Relationship\_Containment.Role\_Whole),

Object\_Interface\_Block.explode(Internal\_Block\_Diagram),

Object\_Constraint\_Block.explode(Internal\_Block\_Diagram),

);

annotation(

localLabel="模块定义图",

type = diagram,

description="",

);

end Block\_Defintion\_Diagram;

活动图的图元模型描述如下，包含两个图属性，声明12个对象元模型，声明3个关系元模型，定义12个绑定关系，定义5个剖视和4个分解。

partial Graph Activity\_Diagram

initial Property Activity\_Diagram\_Property\_To\_Do\_4f5d74 = "null" extends Property\_To\_Do annotation(localLabel ="待办事项");

initial Property Activity\_Diagram\_Property\_Context\_2475bb = "null" extends Property\_Context annotation(localLabel ="上下文");

Object Object\_Call\_Behavior\_Action;

Object Object\_Call\_Operation\_Action;

Object Object\_Opaque\_Action;

Object Object\_Central\_Buffer\_Node;

Object Object\_Data\_Store\_Node;

Object Object\_Activity\_Parameter\_Node;

Object Object\_Send\_Signal\_Action;

Object Object\_Accept\_Event\_Action;

Object Object\_Accept\_Time\_Event\_Action;

Object Object\_Accept\_Change\_Structural\_Feature\_Event\_Action;

Object Object\_Initial\_Node;

Object Object\_Activity\_Final\_Node;

Relationship Relationship\_Control\_Flow;

Relationship Relationship\_Object\_Flow;

Relationship Relationship\_Exception\_Handler;

Constraint (

bind connector(Object\_Opaque\_Action,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Initial\_Node,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Structured\_Activity\_Node,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Accept\_Event\_Action,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Decision\_Node,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Fork\_Node,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Accept\_Time\_Event\_Action,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Conditional\_Node,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Sequence\_Node,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Merge\_Node,Relationship\_Control\_Flow.Role\_Source),

bind connector(Object\_Send\_Signal\_Action,Relationship\_Control\_Flow.Role\_Source),

ence\_Node\_Point\_Input\_Pin\_8f74f4,Relationship\_Exception\_Handler.Role\_Exception\_Input),

bind connector(Object\_Structured\_Activity\_Node.Object\_Structured\_Activity\_Node\_Point\_Action\_Input\_Pin\_7d44d9,Relationship\_Exception\_Handler.Role\_Exception\_Input),

Object\_Conditional\_Node.explode(Activity\_Diagram),

Object\_Loop\_Node.explode(Activity\_Diagram),

Object\_Swimlanes.explode(Activity\_Diagram),

Object\_Call\_Behavior\_Action.explode(Activity\_Diagram),

Object\_Block.explode(Activity\_Diagram),

Object\_Interruptible\_Activity\_Region.decompose(Activity\_Diagram),

Object\_Structured\_Activity\_Node.decompose(Activity\_Diagram),

Object\_Expansion\_Region.decompose(Activity\_Diagram),

Object\_Sequence\_Node.decompose(Activity\_Diagram),

);

annotation(

localLabel="活动图",

type = diagram,

description="",

);

end Activity\_Diagram;

内部模块图部分描述如下，包含2个图属性，声明6个对象元模型，声明4个关系元模型，定义12个绑定关系，定义3个分解。

partial Graph Internal\_Block\_Diagram

initial Property Internal\_Block\_Diagram\_Property\_To\_Do\_c37486 = "null" extends Property\_To\_Do annotation(localLabel ="待办事项");

initial Property Internal\_Block\_Diagram\_Property\_Context\_9f574a = "null" extends Property\_Context annotation(localLabel ="上下文");

Object Object\_Value\_Property;

Object Object\_Part\_Property;

Object Object\_Reference\_Property;

Object Object\_Constraint\_Property;

Object Object\_Flow\_Property;

Object Object\_Participant\_Property;

Relationship Relationship\_Connector;

Relationship Relationship\_Binding\_Connector;

Relationship Relationship\_Item\_Flow\_Connector;

Relationship Relationship\_Item\_Flow\_Binding\_Connector;

Constraint (

bind connector(Object\_Reference\_Property.Object\_Reference\_Property\_Point\_Proxy\_Port\_0f0b10,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Constraint\_Property.Object\_Constraint\_Property\_Point\_Flow\_Port\_acfb89,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Bound\_Reference,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Reference\_Property.Object\_Reference\_Property\_Point\_Constraint\_Parameter\_aaf75b,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Part\_Property,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Classifier\_Behavior\_Property,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Reference\_Property.Object\_Reference\_Property\_Point\_Full\_Port\_06fd14,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Part\_Property.Object\_Part\_Property\_Point\_Constraint\_Parameter\_119082,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Reference\_Property.Object\_Reference\_Property\_Point\_Port\_b44640,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Constraint\_Property.Object\_Constraint\_Property\_Point\_Constraint\_Parameter\_850c76,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Constraint\_Property.Object\_Constraint\_Property\_Point\_Full\_Port\_2ca47c,Relationship\_Connector.Role\_Connectot\_End\_A),

bind connector(Object\_Constraint\_Property.Object\_Constraint\_Property\_Point\_Proxy\_Port\_d887a1,Relationship\_Connector.Role\_Connectot\_End\_A),

Object\_Part\_Property.decompose(Internal\_Block\_Diagram),

Object\_Reference\_Property.decompose(Internal\_Block\_Diagram),

Object\_Constraint\_Property.decompose(Internal\_Block\_Diagram),

);

annotation(

localLabel="内部模块图",

type = diagram,

description="",

);

end Internal\_Block\_Diagram;

时序图描述如下，包含2个图属性，声明2个对象元模型，声明1个关系元模型，定义2个绑定关系。

partial Graph Sequence\_Diagram

initial Property Sequence\_Diagram\_Property\_To\_Do\_d3ddca = "null" extends Property\_To\_Do annotation(localLabel ="待办事项");

initial Property Sequence\_Diagram\_Property\_Context\_847d79 = "null" extends Property\_Context annotation(localLabel ="上下文");

Object Lifeline;

Object Object\_Block;

Relationship Message;

Constraint (

bind connector(Lifeline,Message.Message\_Source),

bind connector(Lifeline,Message.Message\_Target),

);

annotation(

localLabel="时序图",

type = sequenceDiagram,

description="",

);

end Sequence\_Diagram;

状态机图描述如下，包含2个图属性，声明13个对象元模型，声明1个关系元模型，定义32个绑定关系，定义4个剖视。

partial Graph State\_Machine\_Diagram

initial Property State\_Machine\_Diagram\_Property\_To\_Do\_a9cf1e = "null" extends Property\_To\_Do annotation(localLabel ="待办事项");

initial Property State\_Machine\_Diagram\_Property\_Context\_43446a = "null" extends Property\_Context annotation(localLabel ="上下文");

Object Object\_State;

Object Object\_Composite\_State;

Object Object\_Orthogonal\_State;

Object Object\_Submachine\_State;

Object Object\_Initial\_Pseudo\_State;

Object Object\_Final\_State;

Object Object\_Terminate\_Pseudo\_State;

Object Object\_Deep\_History\_Pseudostate;

Object Object\_Shallow\_History\_Pseudo\_State;

Object Object\_Junction\_Pseudo\_State;

Object Object\_Choice\_Pseudo\_State;

Object Object\_Fork\_Pseudo\_State;

Object Object\_Join\_Pseudo\_State;

Relationship Relationship\_Transition;

Constraint (

bind connector(Object\_Terminate\_Pseudo\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Orthogonal\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Composite\_State.Object\_Composite\_State\_Point\_Pseudo\_State\_373e35,Relationship\_Transition.Role\_Source),

bind connector(Object\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Submachine\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Fork\_Pseudo\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Join\_Pseudo\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_State.Object\_State\_Point\_Pseudo\_State\_4d1a30,Relationship\_Transition.Role\_Source),

bind connector(Object\_Deep\_History\_Pseudostate,Relationship\_Transition.Role\_Source),

bind connector(Object\_Shallow\_History\_Pseudo\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Initial\_Pseudo\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Choice\_Pseudo\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Junction\_Pseudo\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Orthogonal\_State.Object\_Opaque\_Action\_Point\_Pseudo\_State\_981043,Relationship\_Transition.Role\_Source),

bind connector(Object\_Submachine\_State.Object\_Submachine\_State\_Point\_Connection\_Point\_Reference\_b39e28,Relationship\_Transition.Role\_Source),

bind connector(Object\_Composite\_State,Relationship\_Transition.Role\_Source),

bind connector(Object\_Terminate\_Pseudo\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_Orthogonal\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_Composite\_State.Object\_Composite\_State\_Point\_Pseudo\_State\_373e35,Relationship\_Transition.Role\_Target),

bind connector(Object\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_Submachine\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_Fork\_Pseudo\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_Join\_Pseudo\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_Final\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_State.Object\_State\_Point\_Pseudo\_State\_4d1a30,Relationship\_Transition.Role\_Target),

bind connector(Object\_Deep\_History\_Pseudostate,Relationship\_Transition.Role\_Target),

bind connector(Object\_Shallow\_History\_Pseudo\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_Choice\_Pseudo\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_Junction\_Pseudo\_State,Relationship\_Transition.Role\_Target),

bind connector(Object\_Orthogonal\_State.Object\_Opaque\_Action\_Point\_Pseudo\_State\_981043,Relationship\_Transition.Role\_Target),

bind connector(Object\_Submachine\_State.Object\_Submachine\_State\_Point\_Connection\_Point\_Reference\_b39e28,Relationship\_Transition.Role\_Target),

bind connector(Object\_Composite\_State,Relationship\_Transition.Role\_Target),

Object\_State.explode(State\_Machine\_Diagram),

Object\_Submachine\_State.explode(State\_Machine\_Diagram),

Object\_Composite\_State.explode(State\_Machine\_Diagram),

Object\_Orthogonal\_State.explode(State\_Machine\_Diagram),

);

annotation(

localLabel="状态机图",

type = diagram,

description="",

);

end State\_Machine\_Diagram;

用例图部分语言描述如下，包含2个图属性，声明8个对象元模型，声明5个关系元模型，定义28个绑定关系，定义4个剖视。

partial Graph Use\_Case\_Diagram

initial Property Use\_Case\_Diagram\_Property\_To\_Do\_da51d5 = "null" extends Property\_To\_Do annotation(localLabel ="待办事项");

initial Property Use\_Case\_Diagram\_Property\_Context\_afc85d = "null" extends Property\_Context annotation(localLabel ="上下文");

Object Object\_Actor;

Object Object\_Use\_Case;

Object Object\_Boundary\_System;

Object Object\_Environmental\_Effect;

Object Object\_External\_System;

Object Object\_Sensor;

Object Object\_System\_Context;

Object\_Package;

Relationship Relationship\_Include;

Relationship Relationship\_Extend;

Relationship Relationship\_Association;

Relationship Relationship\_Generalization;

Relationship Relationship\_Containment;

Constraint (

bind connector(Object\_Use\_Case,Relationship\_Include.Role\_Including\_Case),

bind connector(Object\_Use\_Case,Relationship\_Include.Role\_Addition),

bind connector(Object\_Use\_Case,Relationship\_Extend.Role\_Extension),

bind connector(Object\_Use\_Case,Relationship\_Extend.Role\_Extended\_Case),

bind connector(Object\_Boundary\_System,Relationship\_Association.Role\_Role\_B),

bind connector(Object\_External\_System,Relationship\_Association.Role\_Role\_B),

bind connector(Object\_System\_Context,Relationship\_Association.Role\_Role\_B),

bind connector(Object\_Actor,Relationship\_Association.Role\_Role\_B),

bind connector(Object\_Sensor,Relationship\_Association.Role\_Role\_B),

bind connector(Object\_Use\_Case,Relationship\_Association.Role\_Role\_B),

bind connector(Object\_Boundary\_System,Relationship\_Association.Role\_Role\_A),

bind connector(Object\_Boundary\_System,Relationship\_Generalization.Role\_General),

bind connector(Object\_External\_System,Relationship\_Generalization.Role\_General),

bind connector(Object\_System\_Context,Relationship\_Generalization.Role\_General),

bind connector(Object\_Actor,Relationship\_Generalization.Role\_General),

bind connector(Object\_Use\_Case,Relationship\_Generalization.Role\_General),

bind connector(Object\_Boundary\_System,Relationship\_Containment.Role\_Part),

bind connector(Object\_User\_System,Relationship\_Containment.Role\_Part),

bind connector(Object\_System\_Context,Relationship\_Containment.Role\_Part),

bind connector(Object\_Actor,Relationship\_Containment.Role\_Part),

bind connector(Object\_Use\_Case,Relationship\_Containment.Role\_Part),

bind connector(Object\_Package,Relationship\_Containment.Role\_Part),

bind connector(Object\_Boundary\_System,Relationship\_Containment.Role\_Whole),

bind connector(Object\_User\_System,Relationship\_Containment.Role\_Whole),

bind connector(Object\_System\_Context,Relationship\_Containment.Role\_Whole),

bind connector(Object\_Actor,Relationship\_Containment.Role\_Whole),

bind connector(Object\_Use\_Case,Relationship\_Containment.Role\_Whole),

bind connector(Object\_Package,Relationship\_Containment.Role\_Whole),

Object\_Block.explode(Internal\_Block\_Diagram),

Object\_System\_Context.explode(Internal\_Block\_Diagram),

Object\_Package.decompose(Use\_Case\_Diagram),

);

annotation(

localLabel="用例图",

type = diagram,

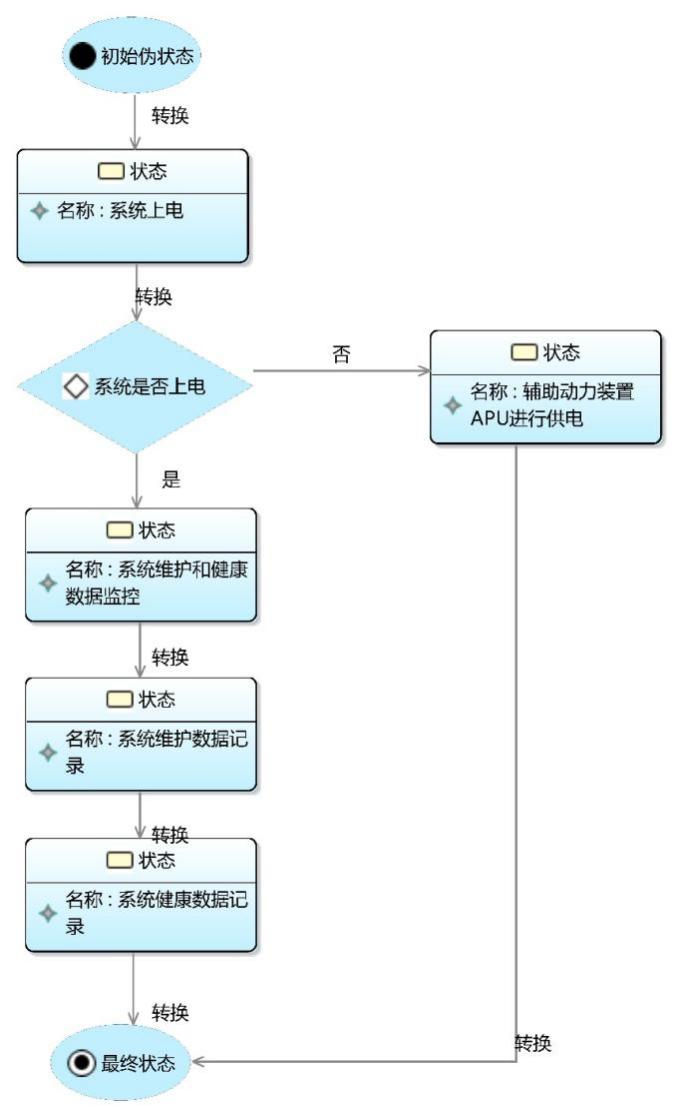
description="",

);

end Use\_Case\_Diagram;

* 1. 模型
     1. 使命分析阶段

在驾驶舱预先准备阶段，飞机首先进行系统上电，若上电失败，则进行辅助动力装置APU进行供电；若上电成功，则进行系统维护数据和健康数据的监控，并将数据进行记录。模型示意见图B.3：



* 1. 使命分析阶段模型示意图

模型中对象实例“初始伪状态”的语法如下：

partial Model model\_State\_Machine\_Diagram\_9594e instanceof State\_Machine\_Diagram

Property model\_State\_Machine\_Diagram\_9594e\_property\_54ad = "null" refer State\_Machine\_Diagram\_Property\_To\_Do\_a9cf1e annotation(localLabel = "待办事项",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_State\_Machine\_Diagram\_9594e\_property\_9c2d = "null" refer State\_Machine\_Diagram\_Property\_Context\_43446a annotation(localLabel = "上下文",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Object model\_State\_Machine\_Diagram\_9594e\_object\_649c refer Object\_Initial\_Pseudo\_State

Property model\_State\_Machine\_Diagram\_9594e\_object\_649c\_property\_f6d9 = "null" refer Object\_Initial\_Pseudo\_State\_Property\_Name\_8df0e9 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_State\_Machine\_Diagram\_9594e\_object\_649c\_property\_91cd = "null" refer Object\_Initial\_Pseudo\_State\_Property\_To\_Do\_0010ed annotation(localLabel = "待办事项",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

annotation(

initialLocation=(447,140,83,46),

localLabel="初始伪状态",

shape="circle",

color="194,239,255;0,0,0;0,0,0",

screenMode=blankMode,

imageAddress="",

SDIdentification="",

text=("Microsoft YaHei UI",8,Normal,Normal),

iconDisplay=true,

frame=(0,"solid"),

visualizationList="",

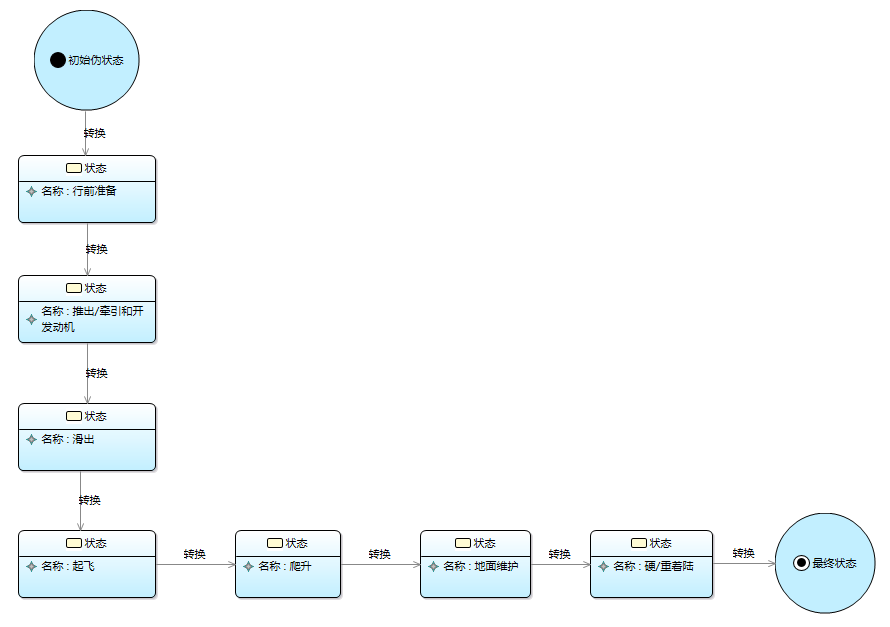
);

end model\_State\_Machine\_Diagram\_9594e\_object\_649c;

......

* + 1. 运行分析阶段

定义了机载维护与健康管理系统的使用和运行阶段，包括：飞机的行前准备阶段、推出/牵引和开发动机、画出、起飞、爬升、地面维护、硬/重着陆等阶段。模型示意见图B.4：



* 1. 运行分析阶段模型示意图

模型中对象实例“初始伪状态”和“最终状态”的语法如下：

partial Model model\_State\_Machine\_Diagram\_675e4 instanceof State\_Machine\_Diagram

Property model\_State\_Machine\_Diagram\_675e4\_property\_87d7 = "null" refer State\_Machine\_Diagram\_Property\_To\_Do\_a9cf1e annotation(localLabel = "待办事项",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_State\_Machine\_Diagram\_675e4\_property\_0c3a = "null" refer State\_Machine\_Diagram\_Property\_Context\_43446a annotation(localLabel = "上下文",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Object model\_State\_Machine\_Diagram\_675e4\_object\_5792 refer Object\_Initial\_Pseudo\_State

Property model\_State\_Machine\_Diagram\_675e4\_object\_5792\_property\_029d = "null" refer Object\_Initial\_Pseudo\_State\_Property\_Name\_8df0e9 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_State\_Machine\_Diagram\_675e4\_object\_5792\_property\_117c = "null" refer Object\_Initial\_Pseudo\_State\_Property\_To\_Do\_0010ed annotation(localLabel = "待办事项",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

annotation(

initialLocation=(74,50,106,101),

localLabel="初始伪状态",

shape="circle",

color="194,239,255;0,0,0;0,0,0",

screenMode=blankMode,

imageAddress="",

SDIdentification="",

text=("Microsoft YaHei UI",8,Normal,Normal),

iconDisplay=true,

frame=(0,"solid"),

visualizationList="",

);

end model\_State\_Machine\_Diagram\_675e4\_object\_5792;

......

Object model\_State\_Machine\_Diagram\_675e4\_object\_2762 refer Object\_Final\_State

Property model\_State\_Machine\_Diagram\_675e4\_object\_2762\_property\_8252 = "null" refer Object\_Final\_State\_Property\_Name\_d73887 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_State\_Machine\_Diagram\_675e4\_object\_2762\_property\_9c5d = "null" refer Object\_Final\_State\_Property\_To\_Do\_cd58a6 annotation(localLabel = "待办事项",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_State\_Machine\_Diagram\_675e4\_object\_2762\_property\_e4ff = "null" refer Object\_Final\_State\_Property\_Redefined\_Vertex\_8d4b3f annotation(localLabel = "重定义顶点",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_State\_Machine\_Diagram\_675e4\_object\_2762\_property\_66a7 = "null" refer Object\_Final\_State\_Property\_Entry\_Behavior\_Element\_b3affc annotation(localLabel = "入口行为元素",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_State\_Machine\_Diagram\_675e4\_object\_2762\_property\_89f6 = "null" refer Object\_Final\_State\_Property\_Do\_Activity\_Behavior\_Element\_3c6e7b annotation(localLabel = "执行活动行为元素",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_State\_Machine\_Diagram\_675e4\_object\_2762\_property\_116f = "null" refer Object\_Final\_State\_Property\_Exit\_Behavior\_Element\_207108 annotation(localLabel = "出口行为元素",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

annotation(

initialLocation=(815,553,101,101),

localLabel="最终状态",

shape="circle",

color="194,239,255;0,0,0;0,0,0",

screenMode=blankMode,

imageAddress="",

SDIdentification="",

text=("Microsoft YaHei UI",8,Normal,Normal),

iconDisplay=true,

frame=(0,"solid"),

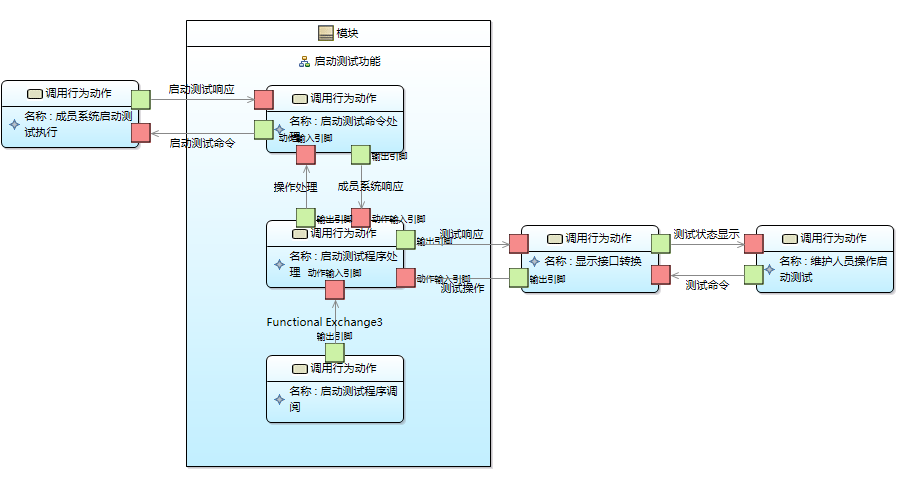
visualizationList="",

);

end model\_State\_Machine\_Diagram\_675e4\_object\_2762;

* + 1. 功能分析阶段

该模型为启动测试功能，成员系统启动测试功能，最后将测试结果给到维护人员，模型示意见图B.5。



* 1. 功能分析阶段模型示意图

模型中对象实例“模块”的语法如下：

partial Model h6Cj24q4 instanceof Activity\_Diagram

Property h6Cj24q4\_property\_0885 = "null" refer Activity\_Diagram\_Property\_To\_Do\_4f5d74 annotation(localLabel = "待办事项",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property h6Cj24q4\_property\_0114 = "null" refer Activity\_Diagram\_Property\_Context\_2475bb annotation(localLabel = "上下文",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Object h6Cj24q4\_object\_91f8 refer Object\_Block

this.explode(pgH6ENh3);

annotation(

initialLocation=(185,100,303,418),

localLabel="模块",

shape="rectangle",

color="0,0,0;0,0,0;0,0,0;194,239,255;255,255,255",

screenMode=decompositionMode,

imageAddress="",

SDIdentification="",

text=("Microsoft YaHei UI",8,Normal,Normal),

iconDisplay=true,

frame=(1,"solid"),

visualizationList="pgH6ENh3:(0,0,303,418)",

);

end h6Cj24q4\_object\_91f8;

......

* + 1. 逻辑架构分析阶段

图B.6所示为提供飞机状态信息功能场景时序图，先向机载维护与健康管理系统提供飞机的基本参数，随后机载维护与健康管理针对提供的基本参数进行计算，计算完成后将计算得到的状态消息输出给飞机，飞机接收相关状态消息。

图示

描述已自动生成

* 1. 逻辑架构分析阶段模型示意图

模型中对象实例“计算飞行状态消息参数”的语法如下：

partial Model sKIwwhQk instanceof Sequence\_Diagram

Property sKIwwhQk\_property\_71ea = "null" refer Sequence\_Diagram\_Property\_To\_Do\_d3ddca annotation(localLabel = "待办事项",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property sKIwwhQk\_property\_df6b = "null" refer Sequence\_Diagram\_Property\_Context\_847d79 annotation(localLabel = "上下文",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Object sKIwwhQk\_\_Execution\_Lifeline\_\_3445 refer Lifeline

annotation(

initialLocation=(324,117,10,35),

localLabel="计算飞行状态消息参数",

shape="rectangle",

color="209,209,209;0,0,0;0,0,0",

screenMode=propertyMode,

imageAddress="",

SDIdentification="Execution,relationship\_Message\_\_Message\_Target\_\_5c99,",

text=("Microsoft YaHei UI",7,Normal,Normal),

iconDisplay=true,

frame=(1,""),

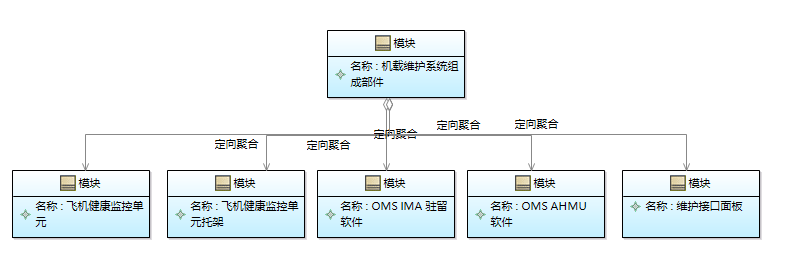
visualizationList="",

);

end sKIwwhQk\_\_Execution\_Lifeline\_\_3445;

......

如图B.7所示，机载维护与健康管理系统的硬件由飞机健康监控单元、飞机健康监控单元托架、OMS IMA驻留软件、OMS AHMU软件、维护接口面板组成。



* 1. 物理架构分析阶段模型示意图

模型中对象实例“模块”包含对象属性“机载维护系统组成部件”的语法如下：

partial Model model\_Block\_Defintion\_Diagram\_eeffd instanceof Block\_Defintion\_Diagram

Property model\_Block\_Defintion\_Diagram\_eeffd\_property\_87af = "null" refer Block\_Defintion\_Diagram\_Property\_To\_Do\_3ac03f annotation(localLabel = "待办事项",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Property model\_Block\_Defintion\_Diagram\_eeffd\_property\_c83e = "null" refer Block\_Defintion\_Diagram\_Property\_Context\_9794be annotation(localLabel = "上下文",color="0,0,0",text=("Microsoft YaHei UI",7,Normal,Normal),underline=false,strikeThrough=false);

Object model\_Block\_Defintion\_Diagram\_eeffd\_object\_174b refer Object\_Block

Property model\_Block\_Defintion\_Diagram\_eeffd\_object\_174b\_property\_19ba = "机载维护系统组成部件" refer Object\_Block\_Property\_Name\_525f94 annotation(localLabel = "名称",color="0,0,0",text=("Microsoft YaHei UI",8,Normal,Normal),underline=false,strikeThrough=false);

annotation(

initialLocation=(475,135,140,70),

localLabel="模块",

shape="rectangle",

color="0,0,0;0,0,0;0,0,0;194,239,255;255,255,255",

screenMode=decompositionMode,

imageAddress="",

SDIdentification="",

text=("Microsoft YaHei UI",8,Normal,Normal),

iconDisplay=true,

frame=(1,"solid"),

visualizationList="",

);

end model\_Block\_Defintion\_Diagram\_eeffd\_object\_174b;

......

1. （资料性）  
   单位

单位表达式跟随在属性声明、属性初始化或属性赋值表达式中属性实例的后面，通过“unit”关键字定义单位表达式。表C.1展示统一架构建模语言单位。

Int A(unit = “kg”);

Int A(unit = “kg”) = 1;

* 1. 统一架构建模语言单位

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 名称 | 单位 | 多架构建模语言语义 | 名称 | 单位 | 多架构建模语言语义 | 名称 | 单位 | 多架构建模语言语义 |
| 系数 | 1 | 1 | 电阻 | Ohm | ohm | 光量 | lm·s | lms |
| 角度 | deg | deg | 电阻率 | Ohm·m | ohmm | 亮度 | cd/m2 | cd/m^2 |
| 弧度角 | rad | rad | 传导率 | S/m | S/m | 光出射 | lm/m2 | lm/m^2 |
| 立体角 | sr | sr | 视在功率 | VA | va | 照度 | lx | lx |
| 长度 | m | m | 物质的量 | mol | mol | 曝光量 | lx·s | lxs |
| 面积 | m2 | m^2 | 摩尔质量 | kg/mol | kg/mol | 发光效率 | lm/W | lm/w |
| 体积 | m3 | m^3 | 摩尔体积 | m3/mol | m^3/mol | 摩尔吸收系数 | m2/mol | m^2/mol |
| 角速度 | rad/s | rad/s | 摩尔内能 | J/mol | j/mol | 声阻抗 | Pa·s/m3 | pas/m^3 |
| 角加速度 | rad/s2 | rad/s^2 | 摩尔热容 | J/(mol·K) | j/molk | 比声阻抗 | Pa·s/m | pas/m |
| 速度 | m/s | m/s | 摩尔速率 | mol/s | mol/s | 响度级 | phon | phon |
| 加速度 | m/s2 | m/s^2 | 跨导特性 | A/V2 | a/v^2 | 响度 | sone | sone |
| 频率 | Hz | hz | 反转电位 | 1/V | 1/v | 分子数密度 | 1/m3 | 1/m^3 |
| 质量 | kg | kg | 电力常数 | N/A | n/a | 浓度 | mol/m3 | mol/m^3 |
| 密度 | kg/m3 | kg/m^3 | 光谱辐射能密度 | J/m4 | j/m^4 | 化学势 | J/mol | j/mol |
| g/cm3 | g/cm^3 | 辐射强度 | W/sr | w/sr | 分子极化率 | C·m2/V | cm^2/v |
| 力 | N | n | 辐射亮度 | W/(sr·m2) | w/srm^2 | 扩散系数 | m2/s | m^2/s |
| 刚度 | N/m | n/m | 虚功率 | var | var | 法拉第常数 | C/mol | c/mol |
| 力矩 | N·m | nm | 波数 | 1/m | 1/m | 离子强度 | mol/kg | mol/kg |
| 压力 | Pa | pa | 循环波数 | rad/m | rad/m | 电导率 | S/m | s/m |
| bar | bar | 分贝 | dB | db | 摩尔电系数 | S·m2/mol | sm^2/mol |
| 能量 | J | j | 比容 | m3/kg | m^3/kg | 旋磁系数 | A·m2/(J·s) | Am^2/js |
| 能量密度 | J/m3 | j/m^3 | 线密度 | kg/m | kg/m | 活度 | Bq | Bq |
| 功率 | W | w | 面密度 | kg/m2 | kg/m^2 | 比活度 | Bq/kg | Bq/kg |
| 热力学温度 | K | k | 动量 | kg·m/s | kgm/s | 角截面 | m2/sr | m^2/sr |
| 温度斜率 | K/s | k/s | 冲量 | N·s | ns | 光谱截面 | m2/J | m^2/j |
| 线性温度因数 | 1/K | 1/k | 角动量 | kg·m2/s | kgm^2/s | 光谱角截面 | m2/(sr·J) | m^2/srj |
| 二次温度因数 | 1/K2 | 1/k^2 | 角冲量 | N·m·s | nms | 粒子注量 | 1/m2 | 1/m^2 |
| 压力因数 | Pa/K | pa/k | 转动惯量 | kg·m2 | Kgm^2 | 粒子注量率 | 1/(s·m2) | 1/sm^2 |
| 压缩率 | 1/pa | 1/pa | 平动阻尼常数 | N·s/m | ns/m | 能通量 | j/m2 | j/m^2 |
| 电流 | A | a | 电力矩 | N·m/A | nm/a | 能通量率 | W/m2 | w/m^2 |
| 电流斜率 | A/s | a/s | 旋转弹簧常数 | N·m/rad | nm/rad | 粒子电流密度 | 1/(m2·s) | 1/m^2s |
| 电荷 | C | c | 旋转阻尼常数 | N·m·s/rad | nms/rad | 质量衰减系数 | m2/kg | m^2/kg |
| 电荷体密度 | C/m3 | c/m^3 | 面积二阶矩 | m4 | m^4 | 线性功率 | J/m | j/m |
| 电荷面密度 | C/m2 | C/m^2 | 质量流量 | kg/s | kg/s | 总原子阻止力 | J·m2 | j/m^2 |
| 电场强度 | V/m | v/m | 体积流量 | m3/s | m^3/s | 总质量制动力 | J·m2/kg | j/m^2kg |
| 电势 | V | v | 热通量 | W/m2 | w/m^2 | 动力 | m2/(V·s) | m^2/vs |
| 电势斜率 | V/s | v/s | 热导 | W/K | w/k | 中子源密度 | 1/(m3·s) | 1/m^3s |
| 电容 | F | f | 热导率 | W/(m·K) | w/mk | 吸收剂量 | Gy | gy |
| 电容率 | F/m | f/m | 热传递系数 | W/(m2·K) | w/m^2k | 剂量当量 | Sv | sv |
| 电偶极矩 | C·m | cm | 热绝缘 | m2·K/W | m^2k/w | 吸收剂量率 | Gy/s | gy/s |
| 电流密度 | A/m2 | a/m^2 | 热扩散率 | m2/s | m^2/s | 照射 | C/kg | c/kg |
| 线性电流密度 | A/m | a/m | 热容 | J/K | j/k | 照射率 | C/(kg·s) | c/kgs |
| 磁通量密度 | T | t | 比热容 | J/(kg·K) | j/kgk | 光谱浓度 | s/m3 | s/m^3 |
| 磁通量 | Wb | wb | 熵流量 | J/(K·s) | j/ks | 状态密度 | 1/(J·m3) | 1/jm^3 |
| 磁矢势 | Wb/m | wb/m | 焓密度 | kg·s2/m5 | kgs^2/m^5 | 阻尼系数 | 1/s | 1/s |
| 磁导率 | H/m | h/m | 压力密度 | s2/m2 | s^2/m^2 | 对数减量 | 1/S | 1/sa |
| 电磁矩 | A·m2 | am^2 | 温度密度 | kg/(m3·K) | kg/m^3k | 衰减系数 | 1/m | 1/ma |
| 磁偶极矩 | Wb·m | wbm | 压力焓 | J·m·s2/kg2 | jms^2/kg^2 | 默认单位 | 1 | 1 |
| 磁阻 | 1/H | 1/h | 密度能量 | J·m3/kg | jm^3/kg | 时间 | s | s1 |
| 磁导 | H | h2 | 发光强度 | Cd | cd | 热阻 | K/W | k/w |
| 电导 | S | s | 光通量 | lm | lm | 压力能量 | J·m·s2/kg | jms^2/kg |
| 电感 | H | h |  |  |  |  |  |  |