ICS 35.240.50
L 67

中华人民共和国国家标准

GB/T 16656.203—1997
idt ISO 10303-203:1994

工业自动化系统与集成 产品数据的表达与交换 第203部分:应用协议:配置控制设计

**Industrial automation systems and integration—
Product data representation and exchange—
rt 203: Application protocol: Configuration controlled design**

1997-09-02发布　　1998-04-01实施

国家技术监督局 发布

目　　次

前言 …… V
ISO 前言 …… Ⅶ
引言 …… Ⅸ
1　范围 …… 1
2　引用标准 …… 2
3　定义和缩写 …… 2
3.1　GB/T 16656.1 中定义的术语 …… 2
3.2　GB/T 16656.31 中定义的术语 …… 3
3.3　GB/T 16656.42 中定义的术语 …… 3
3.4　GB/T 16656.43 中定义的术语 …… 4
3.5　GB/T 16656.44 中定义的术语 …… 4
3.6　其他定义 …… 4
3.6.1　边界表示模型 …… 4
3.6.2　设计阶段 …… 4
3.6.3　机械零件 …… 4
3.6.4　实体模型 …… 4
3.6.5　子装配件 …… 4
3.6.6　线框模型 …… 5
3.7　缩写 …… 5
4　信息需求 …… 5
4.1　功能单元 …… 5
4.1.1　高级的边界表示 …… 6
4.1.2　授权 …… 6
4.1.3　物料清单 …… 6
4.1.4　设计活动控制 …… 6
4.1.5　设计信息 …… 6
4.1.6　有效性 …… 7
4.1.7　截止项标识 …… 7
4.1.8　小平面边界表示 …… 7
4.1.9　拓扑流形曲面 …… 7
4.1.10　无拓扑曲面与线框 …… 7
4.1.11　零件标识 …… 7
4.1.12　形状 …… 8
4.1.13　资源控制 …… 8
4.1.14　拓扑线框 …… 8

4.2 应用对象 …… 8
4.3 应用要求 …… 18
5 应用解释模型 …… 21
5.1 映射表 …… 21
5.2 AIM EXPRESS 简表 …… 51
5.2.1 基本概念和假设 …… 58
5.2.2 配置控制设计类型 …… 60
5.2.3 配置控制设计实体 …… 62
5.2.4 配置控制设计规则 …… 95
5.2.5 配置控制设计函数 …… 136
6 一致性要求 …… 161
6.1 一致性级别 1 的实体 …… 161
6.2 一致性级别 2 的实体 …… 164
6.3 一致性级别 3 的实体 …… 169
6.4 一致性级别 4 的实体 …… 173
6.5 一致性级别 5 的实体 …… 178
6.6 一致性级别 6 的实体 …… 182

附 录

附录 A(标准的附录) 应用解释模型 EXPRESS 详表 …… 188
附录 B(标准的附录) 实体的应用解释模型短名 …… 319
附录 C(标准的附录) 协议实现一致性声明问卷 …… 326
附录 D(标准的附录) 实现方法规定要求 …… 327
附录 E(标准的附录) 信息对象注册 …… 328
E1 文档标识 …… 328
E2 模式标识 …… 328
E2.1 配置控制设计详细模式 …… 328
E2.2 配置控制设计简单模式 …… 328
附录 F(提示的附录) 应用活动模型 …… 328
F1 应用活动模型定义 …… 328
F2 应用活动模型图 …… 330
附录 G(提示的附录) 应用参考模型 …… 333
附录 H(提示的附录) 应用解释模型 EXPRESS-G …… 341
附录 J(提示的附录) 应用解释模型 EXPRESS 列表 …… 380
附录 K(提示的附录) 应用协议使用指南 …… 380
K1 使用测试目的 …… 380
K2 示例部分 …… 381
附录 L(提示的附录) 参考文献 …… 386
实体、属性名中英文对照 …… 388

图

图 F1 IDEF0*)方法的基本标志法 …… 330
图 F2 用 IDEF0 方法描述的第 A0 层管理产品开发模块 …… 331
图 F3 用 IDEF0 方法描述的第 A3 层开发产品设计模块 …… 332
图 G1 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-1 …… 334
图 G2 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-2 …… 335
图 G3 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-3 …… 336
图 G4 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-4 …… 337
图 G5 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-5 …… 338
图 G6 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-6 …… 339
图 G7 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-7 …… 340
图 H1 应用相关环境-应用解释模型(AIM)EXPRESS-G 图 39-1 …… 341
图 H2 产品定义-应用解释模型(AIM)EXPRESS-G 图 39-2 …… 342
图 H3 产品目录-应用解释模型(AIM)EXPRESS-G 图 39-3 …… 343
图 H4 参数定义-应用解释模型(AIM)EXPRESS-G 图 39-4 …… 344
图 H5 参数表达-应用解释模型(AIM)EXPRESS-G 图 39-5 …… 345
图 H6 形状表示关系-应用解释模型(AIM)EXPRESS-G 图 39-6 …… 346
图 H7 表示-应用解释模型(AIM)EXPRESS-G 图 39-7 …… 347
图 H8 几何表示项-应用解释模型(AIM)EXPRESS-G 图 39-8 …… 348
图 H9 拓扑表示项-应用解释模型(AIM)EXPRESS-G 图 39-9 …… 349
图 H10 点-应用解释模型(AIM)EXPRESS-G 图 39-10 …… 350
图 H11 几何有向-应用解释模型(AIM)EXPRESS-G 图 39-11 …… 351
图 H12 曲线-应用解释模型(AIM)EXPRESS-G 图 39-12 …… 352
图 H13 圆锥曲线-应用解释模型(AIM)EXPRESS-G 图 39-13 …… 353
图 H14 有界曲线-应用解释模型(AIM)EXPRESS-G 图 39-14 …… 354
图 H15 曲面曲线-应用解释模型(AIM)EXPRESS-G 图 39-15 …… 355
图 H16 B-样条曲线-应用解释模型(AIM)EXPRESS-G 图 39-16 …… 356
图 H17 曲面-应用解释模型(AIM)EXPRESS-G 图 39-17 …… 357
图 H18 基本曲面-应用解释模型(AIM)EXPRESS-G 图 39-18 …… 358
图 H19 有界曲面-应用解释模型(AIM)EXPRESS-G 图 39-19 …… 359
图 H20 B-样条曲面-应用解释模型(AIM)EXPRESS-G 图 39-20 …… 360
图 H21 拓扑-应用解释模型(AIM)EXPRESS-G 图 39-21 …… 361
图 H22 壳-应用解释模型(AIM)EXPRESS-G 图 39-22 …… 362
图 H23 实体模型-应用解释模型(AIM)EXPRESS-G 图 39-23 …… 363
图 H24 曲面和线框模型-应用解释模型(AIM)EXPRESS-G 图 39-24 …… 364
图 H25 文档-应用解释模型(AIM)EXPRESS-G 图 39-25 …… 365
图 H26 批准-应用解释模型(AIM)EXPRESS-G 图 39-26 …… 366

*)IDEF 是集成计算机辅助制造(ICAM)定义(Definition)方法的缩写。

图 H27 人员和组织-应用解释模型(AIM)EXPRESS-G 图 39-27 …… 367
图 H28 人员和组织赋值-应用解释模型(AIM)EXPRESS-G 图 39-28 …… 368
图 H29 日期和时间-应用解释模型(AIM)EXPRESS-G 图 39-29 …… 369
图 H30 日期和时间赋值-应用解释模型(AIM)EXPRESS-G 图 39-30 …… 370
图 H31 工作和更改文档-应用解释模型(AIM)EXPRESS-G 图 39-31 …… 371
图 H32 认证-应用解释模型(AIM)EXPRESS-G 图 39-32 …… 372
图 H33 合同-应用解释模型(AIM)EXPRESS-G 图 39-33 …… 373
图 H34 安全分类-应用解释模型(AIM)EXPRESS-G 图 39-34 …… 374
图 H35 单位-应用解释模型(AIM)EXPRESS-G 图 39-35 …… 375
图 H36 有单位度量-应用解释模型(AIM)EXPRESS-G 图 39-36 …… 376
图 H37 度量-应用解释模型(AIM)EXPRESS-G 图 39-37 …… 377
图 H38 产品结构-应用解释模型(AIM)EXPRESS-G 图 39-38 …… 378
图 H39 配置-应用解释模型(AIM)EXPRESS-G 图 39-39 …… 379

表

表 1 映射表—高级的边界表示功能单元 …… 23
表 2 映射表—授权功能单元 …… 23
表 3 映射表—物料清单功能单元 …… 26
表 4 映射表—设计活动控制功能单元 …… 30
表 5 映射表—设计信息功能单元 …… 35
表 6 映射表—有效性功能单元 …… 36
表 7 映射表—截止项标识功能单元 …… 38
表 8 映射表—面边界表示功能单元 …… 39
表 9 映射表—拓扑流形曲面功能单元 …… 39
表 10 映射表—无拓扑曲面和线框功能单元 …… 39
表 11 映射表—零件标识功能单元 …… 40
表 12 映射表—形状功能单元 …… 45
表 13 映射表—源控制功能单元 …… 47
表 14 映射表—拓扑线框功能单元 …… 49
表 15 一致性选择 …… 161
表 B1 应用解释模型的实体短名 …… 319

前　言

为了适应信息技术国际化的发展趋势，并贯彻我国标准化工作的双采方针，本国家标准等同采用国际标准 ISO 10303-203:1994《工业自动化系统与集成　产品数据的表达与交换　第 203 部分：应用协议：配置控制设计》(注：原国际标准的名称前后不一致，为便于查询，本标准采用了统一的名称)。我国对应的标准编号为 GB/T 16656.203，本标准是 GB/T 16656 国家标准系列中的一个分标准。

ISO 10303 是适用于计算机对产品数据进行表达与交换的一个系列国际标准。其目的是在产品整个生命周期中提供一种独立于任何特定系统且具有描述产品数据能力的途径。其描述功能不仅适合文件交换，而且也是实现、共享产品数据库以及建档的基础。

在 ISO 10303 系列标准中每一个标准被称为一个 Part(部分)，并单独发表。现已正式发布了如下各部分：

——第 1 部分　概述与基本原理
——第 11 部分　描述方法：EXPRESS 语言参考手册
——第 21 部分　实现方法：交换文件结构纯正文编码
——第 22 部分　实现方法：标准数据访问接口规范
——第 31 部分　一致性测试方法论与框架：基本概念
——第 32 部分　一致性测试方法论与框架：测试实验室与客户需求
——第 41 部分　集成通用资源：产品描述与支持原理
——第 42 部分　集成通用资源：几何与拓扑表达
——第 43 部分　集成通用资源：表达结构
——第 44 部分　集成通用资源：产品结构配置
——第 45 部分　集成通用资源：材料
——第 46 部分　集成通用资源：可视化表示
——第 47 部分　集成通用资源：形变公差
——第 49 部分　集成通用资源：处理结构和性质
——第 101 部分　集成应用资源：制图
——第 104 部分　集成应用资源：有限元分析
——第 105 部分　集成应用资源：运动学
——第 201 部分　应用协议：显式绘图
——第 202 部分　应用协议：相关绘图
——第 203 部分　应用协议：配置控制设计
——第 207 部分　应用协议：钣金计划与设计
——第 210 部分　应用协议：印刷电路装配产品设计数据
——第 213 部分　应用协议：加工零件的数控工艺计划

其中第 1、第 11、第 21、第 31、第 41、第 43、第 44、第 101 和第 203 部分(即本标准)已纳入我国国家标准的采用制定计划。第 1、11、41、43、44、101 部分的国标转化工作已于 1993 年完成；第 11 部分的国标制定工作已于 1995 年完成；第 21、31、203 部分的国标转化工作将于 1996 年底完成。

该系列标准分为描述方法、集成资源、应用协议、抽象测试套件、实现方法及一致性测试六大类。第 203 部分属于应用协议类。

ISO 10303-203 是一个应用协议。它详细阐述了用于交换产品定义和数据的规范。它仅与产品生命

周期的设计阶段有关，并且只有机械零件和装配件的设计可以使用此规范进行交换。

该规范的主要焦点是跟踪和管理产品的控制数据。其数据包括：

——对某组织成员提供产品标识和检索该产品组成零部件的设计标识；

——产生产品设计、发放和正式更改的文档；

——经历了项目正式启动、更改和发放过程的产品发展历史；

——产品的每个零部件与整体的关系结构；

——关于产品的材料、工艺、精加工及其他设计要求的补充材料；

——关于产品或产品设计的合格供应商标识。

按照我国国家标准的具体编写制定原则，本标准的主要编辑性改动如下：

本标准根据 GB/T 1.1—1993 的 4.2.3.3 的规定，保留了原国际标准的前言，并将其作为本标准的“ISO 前言”；又增加一个“前言”，作为本标准的本国前言。

对于带下划线的用于 EXPRESS 语言描述的各英文实体名，为了即要维护其英文原意又要便于了解其名称代表的意思，在此标准中，当其作为标题出现时，我们按 GB/T 1.1 的要求只有中文译名；但在正文中，我们以英文为主，当其在第一次出现或必要时，我们才将中文译名括起来放在英文原名后，且中文译名仅供参考。

通过本标准与国际标准 ISO 10303-203 在技术内容上等同，同时遵守了我国国家标准的具体制定要求，从而保证了本标准的技术国际先进性和与我国国家标准的统一性。

本标准的附录 A、附录 B、附录 C、附录 D、附录 E 是标准附录。

本标准的附录 F、附录 G、附录 H、附录 J、附录 K、附录 L 是提示附录。

本标准由中国标准化与信息分类编码研究所提出并归口。

本标准起草单位：中国标准化与信息分类编码研究所。

本标准主要起草人：秦光里、郭广旭、洪岩。

ISO 前言

国际标准化组织(ISO)是世界标准化机构(ISO 成员)的联合体。国际标准的制定工作是通过 ISO 的技术委员会正式完成的。对已经建立技术委员会的学科感兴趣的每一个成员机构都有权在该委员会作为代表。与 ISO 相联系的国际组织,官方的或非官方的,也参与其工作。ISO 与国际电工委员会(IEC)在电工技术标准化的所有领域也有着紧密的合作。

ISO 10303-203 的制定是由国际标准化组织(ISO)第 184 技术委员会(工业自动化系统与集成)第 4 分技术委员会(工业数据)的第 4 工作组(修饰与集成)(即 ISO TC 184/SC4/WG4)负责完成的。

ISO 10303 的总标题为"工业自动化系统与集成　产品数据的表达与交换",它包括以下部分(分标准):

——第 1 部分　概述与基本原理
——第 11 部分　描述方法:EXPRESS 语言参考手册
——第 21 部分　实现方法:交换文件结构纯正文编码
——第 22 部分　实现方法:标准数据访问接口规范
——第 31 部分　一致性测试方法论与框架:基本概念
——第 32 部分　一致性测试方法论与框架:测试实验室与客户需求
——第 41 部分　集成通用资源:产品描述与支持原理
——第 42 部分　集成通用资源:几何与拓扑表达
——第 43 部分　集成通用资源:表达结构
——第 44 部分　集成通用资源:产品结构配置
——第 45 部分　集成通用资源:材料
——第 46 部分　集成通用资源:可视化表示
——第 47 部分　集成通用资源:形变公差
——第 49 部分　集成通用资源:处理结构和性质
——第 101 部分　集成应用资源:制图
——第 104 部分　集成应用资源:有限元分析
——第 105 部分　集成应用资源:运动学
——第 201 部分　应用协议:显式绘图
——第 202 部分　应用协议:相关绘图
——第 203 部分　应用协议:配置控制设计
——第 207 部分　应用协议:钣金计划与设计
——第 210 部分　应用协议:印刷电路装配产品设计数据
——第 213 部分　应用协议:加工零件的数控工艺计划

本国际标准的结构在 ISO 10303-1 中进行说明。国际标准的编号反映出它的结构:

——第 11 部分规定其描述方法;
——第 21 和 22 部分规定其实现方法;
——第 31 和 32 部分规定其一致性测试方法和框架;
——第 41 到 49 部分规定其集成通用资源;
——第 101 到 105 部分规定其集成应用资源;
——第 201 到 213 部分规定其应用协议;

还有另外的部分(分标准)将发表,它们会遵守同样的编号规则。

附录 A、附录 B、附录 C、附录 D、附录 E 是本标准(ISO 10303-203)的标准附录。附录 F、附录 G、附录 H、附录 J、附录 K、附录 L 仅作为提示的附录。

引　言

ISO 10303 是适用于计算机对产品数据进行表达与交换的一个系列国际标准。其目的是对整个产品生命周期提供一种独立于任何特定系统、能够描述产品数据的中性机制。其描述功能不仅适合文件交换，而且也是实施数据库共享和存档的基础。

在 ISO 10303 系列标准中每一个标准被称为一个 Part(部分)，并单独发表。它们按描述方法、集成资源、应用协议、抽象测试套件、实现方法及一致性测试等类别进行分类。在 ISO 10303-1 中对这些类别进行了说明。第 203 部分属于应用协议类。

本标准(ISO 10303-203)规定了在限定条件下产品数据使用的应用协议，该限定条件满足对配置控制三维产品设计数据交换的工业要求。为了详细阐明产品的构成，各企业组织使用了许多计算机应用系统。完整地定义一个产品包括详细说明它的形状、它的配置及其针对某一特定配置的尽可能多样化的定义。其数据可以存贮在某组织里的一个或多个应用系统上。将产品的形状定义数据与其配置说明数据集成一起对于某组织在完全不同的应用系统中不依赖更多的数据就有能力完整地定义其产品是必不可少的。为了完整地定义被交流的产品设计，企业组织必须能与合作者、供应商及用户共享它的产品数据。

本标准定义了具有三维形状表示的产品定义的交换和定义、及控制那些产品定义的配置数据。本标准仅与产品生命周期的设计阶段有关。只有机械零件和装配件的设计可以使用本标准进行交换。在本标准中可以通过五种不同类型的几何表示的任一种来详细描述一机械零件或装配件的三维形状。

尽管产品设计是重要的，但在本标准中产品形状并不是主要焦点。本标准的主要焦点是跟踪和管理产品的控制数据。该数据包括：

——对某组织成员提供产品标识和检索该产品组成零部件的设计标识；

——产生产品设计、发放和正式更改的文档；

——经历了项目正式启动、更改和发放过程的产品发展历史；

——产品的每个零部件与整体的关系结构；

——关于产品的材料、工艺、精加工及其他设计要求的补充材料；

——关于产品或产品设计的合格供应商标识。

为了机械零件和装配件配置控制的三维设计交换，本标准定义了其相关环境、范围和信息等要求；以及详细描述了满足这些要求所必须的集成资源。

应用协议为 ISO 10303 的开发实现提供基础，也为应用协议实现的一致性测试提供抽象测试套件。

第 1 章定义了本标准的范围，并概括说明了本标准涉及的功能数据。在附录 F 中提供了一个应用活动模型，它是定义本标准范围的基础。在第 4 章中用与本应用相称的专门术语对本应用的信息需求进行了详细描述。在附录 G 中给出了该信息需求的图形表示，它被称之为应用参考模型。

为满足信息需求解释了源结构，并由此产生了应用解释模型(AIM)。5.1 中给出的这个解释表明了信息需求和应用解释模型之间的一致性。应用解释模型的简表在 5.2 中给出，它规定了集成资源的交界面。对于应用解释模型使用的结构，在集成资源中提供了相应定义和 EXPRESS 表达，注意该定义和 EXPRESS 表达可以包括各选择项和没有被引入应用解释模型的各子类。在附录 A 中给出的详表包括该应用解释模型没有注释的全部的 EXPRESS 表达。在附录 H 中给出了应用解释模型的图形表示。在附录 D 中给出了对于特定的实现方法的附加要求。

中华人民共和国国家标准

工业自动化系统与集成 产品数据的表达与交换 第203部分：应用协议：配置控制设计

GB/T 16656.203—1997
idt ISO 10303-203:1994

**Industrial automation systems and integration—
Product data representation and exchange—
Part 203: Application protocol: Configuration controlled design**

1 范围

为了机械零件和装配件配置控制三维设计应用系统间的信息交换，本标准规定了其范围和信息需求所必须的集成资源。在此相关环境下的配置只包括控制三维产品设计数据的过程和数据。这里使用的交换概念就其范围而言，面很窄，只包括作为三维产品定义的那些被交换的数据。在本标准范围内组织交换数据可以有契约关系，而其细节在本标准范围之外。

注：在附录E中应用活动模型提供了过程和信息流的图形表示，此过程和信息流是本标准范围的定义基础。

本标准范围内的内容如下：

a) 产品指机械零件和装配件；

b) 产品开发设计阶段所固有的产品定义数据和配置控制数据；

c) 与更改过程的文档相关的一个设计和数据的更改；

d) 一零件的五种形状表示包括：无拓扑线框和曲面、拓扑线框几何、拓扑流形曲面、小平面边界表示、以及边界表示；

e) 在产品生命周期设计阶段中不同专业所采用的数据其他表达形式；

f) 政府、企业、公司对于某产品设计、工艺、表面加工及设计者制定的有关材料等的标识；

g) 其作用包含在某一产品设计中的政府、企业、公司或其他标准的标识；

h) 跟踪任一设计发放所必需的数据；

i) 对于跟踪一产品的一个设计、一种设计情况或一种配置情况的批准所必需的数据；

j) 产品供应商或设计供应商、以及某一组织认为必要时对该供应商资格信息的标识数据；

k) 一零件如果按合同进行设计，其设计开发依据的合同及其合同的标识；

l) 单个零件或装配件中组成零件的安全分类的标识；

m) 在设计分析或测试中所用、或从中产生的数据，该数据为考虑设计更改提供依据。

本标准范围以外的内容如下：

a) 在设计分析或测试中所用、或从中产生的数据，该数据不为考虑设计更改提供依据；

b) 在设计发放之前的初始开发期间，导致该设计更改的数据；

c) 为产品开发生命周期的各阶段所固有的，而不是设计阶段所固有的产品定义数据和配置控制数据；

d) 面向设计项目控制的业务数据；

国家技术监督局1997-09-02批准　　1998-04-01实施

e) 在设计阶段之外(如:制造),不同专业采用的数据其他表达形式;

f) 物体的 CSG(构造实体几何)表示法;

g) 为产品形状或配置控制数据的可视化表示所固有的数据。

2 引用标准

下列标准所包含的条文,通过在本标准中引用而构成为本标准的条文。本标准出版时,所示版本均为有效。所有标准都会被修订,使用本标准的各方应探讨使用下列标准最新版本的可能性。

GB/T 16656.1 工业自动化系统与集成 产品数据的表达与交换 第1部分:概述与基本原理

GB/T 16656.11—1997 工业自动化系统与集成 产品数据的表达与交换 第11部分:描述方法:EXPRESS 语言参考手册

GB/T 16656.21—1997 工业自动化系统与集成 产品数据的表达与交换 第21部分:交换文件结构纯正文编码

GB/T 16656.31—1997 工业自动化系统与集成 产品数据的表达与交换 第31部分:一致性测试方法论与框架:基本概念

GB/T 16656.41 工业自动化系统与集成 产品数据的表达与交换 第41部分:集成通用资源:产品描述与支持原理

GB/T 16656.42 工业自动化系统与集成 产品数据的表达与交换 第42部分:集成通用资源:几何与拓扑表达

GB/T 16656.43 工业自动化系统与集成 产品数据的表达与交换 第43部分:集成通用资源:表达结构

GB/T 16656.44 工业自动化系统与集成 产品数据的表达与交换 第44部分:集成通用资源:产品结构配置

ISO 31:1992 量和单位

ISO 1000:1992 国际单位及其复合单位,以及推荐使用的其他单位

ISO/IEC 8824-1:1995 信息技术开放系统互连 抽象语法表示法1(ASN.1) 第1部分:基本记法规范

3 定义和缩写

本标准使用了下列术语、定义和缩写。

3.1 GB/T 16656.1 中定义的术语

本标准使用了 GB/T 16656.1 中定义的下列术语:

——抽象测试套件 abstract test suite;

——应用 application;

——应用活动模型 application activity model;

——应用相关环境 application context;

——应用解释模型 application interpreted model;

——应用对象 application object;

——应用协议 application protocol;

——应用参考模型 application reference model;

——装配件 assembly;

——构件 component;

——一致性级别 conformance class;

——一致性需求 conformance requirement;

——数据 data；
——数据交换 data exchange；
——实现方法 implementation method；
——信息 information；
——集成资源 integrated resource；
——解释 interpretation；
——协议实现一致性声明问卷 PICS proforma；
——产品 product；
——产品数据 product data；
——协议实现一致性声明 protocol implementation conformance statement；
——结构 structure；
——功能单元 unit of functionality。

3.2 GB/T 16656.31 中定义的术语

本标准使用了 GB/T 16656.31 中定义的下列术语：
——一致性测试 conformance testing；
——前置处理程序 preprocessor；
——后置处理程序 postprocessor；

3.3 GB/T 16656.42 中定义的术语

本标准使用了 GB/T 16656.42 中定义的下列术语：
——弧连通的 arcwise connected；
——轴对称 axi-symmetric；
——界限 bounds；
——边界 boundary；
——边界表示实体模型 boundary representation solid model；
——闭曲线 closed curve；
——闭曲面 closed surface；
——连通的 connected；
——连通部分 connected component；
——曲线 curve；
——循环 cycle；
——维数 dimensionality；
——域 domain；
——量度 extent；
——有限的 finite；
——几何坐标系 geometric coordinate system；
——图 graph；
——句柄 handle；
——同胚 homeomorphic；
——表 list；
——有边界的 d 维-流形 d-manifold with boundary；
——开曲线 open curve；
——开曲面 open surface；
——可定向的 orientable；

——重叠 overlap；
——参数范围 parameter range；
——参数空间 parameter space；
——定位坐标系 placement coordinate system；
——自交 self-intersect；
——封闭环 self-loop；
——集合 set；
——空间维数 space dimensionality；
——曲面 surface；
——拓扑方向 topological sense。

3.4 GB/T 16656.43 中定义的术语

本标准使用了 GB/T 16656.43 中定义的下列术语：
——坐标空间 coordinate space；
——基于几何的 geometrically founded；
——几何相关的 geometrically related；

3.5 GB/T 16656.44 中定义的术语

本标准使用了 GB/T 16656.44 中定义的下列术语：
——前辈节点 ancestor node；
——物料清单结构 bill-of-material structure；
——子节点 child node；
——后辈节点 descendent node；
——指定非循环图 directed acyclic graph；
——产品的形状、配合与功能 form,fit and function of a product；
——叶节点 leaf node；
——链 link；
——批 lot；
——节点 node；
——父节点 parent node；
——零件表结构 parts list strucutre；
——约定用法 promissory use；
——根节点 root node。

3.6 其他定义

本标准使用了下列术语和定义。

3.6.1 边界表示模型 boundary representation model

实体模型用划定它们内外边界的壳集合来表示。约束条件保证能明确定义相关联的几何实体，并且保证连接该模型中顶点、边、面、环和壳个数满足欧拉公式。

3.6.2 设计阶段 design phase

产品工程描述的动态时期。

3.6.3 机械零件 machanical part

由材料形成有固定形状的物理对象。

3.6.4 实体模型 solid model

明确定义的具有用二维边界区分内部和外部的三维对象。

3.6.5 子装配件 sub-assembly

一组件，其本身是个装配件。

3.6.6 线框模型 wireframe model

一个模型，其包括全部的点、线以及表示一形状的那些边的面。

3.7 缩写

本标准采用了以下缩写：

AAM：应用活动模型(application activity model)

AIM：应用解释模型(application interpreted model)

AP：应用协议(application protocol)

ARM：应用参考模型(application reference model)

BOM：物料清单(bill-of-material)

CAD：计算机辅助设计(computer aided design)

CM：配置控制(configuration management)

DAG：有向非循环图(directed acyclic graph)

ICAM：集成计算机辅助制造(integrated computer-aided manufacturing)

ID：标识(identification)

IDEF0：集成计算机辅助制造定义语言 0(ICAM definition language 0)

IDEF1X：集成计算机辅助制造定义语言 1—扩展(ICAM definition language 1-extended)

PICS：协议资料和一致性声明(Protocol information and conformance statement)

UoF：功能单元(Units of Functionality)

4 信息需求

本章规定了用于交换机械零件或装配件的配置控制三维设计的信息需求。

把这些信息需求规定为功能单元、应用对象和应用要求的一个集合。这些要求适合于单个应用对象及各应用对象间的相关关系。这些被规定的信息需求使用了本标准学科领域的术语。

注

1 在附录 G 中给出了该信息需求的 A 图形表示。

2 该信息需求与附录 F 中作为本标准范围内被标识了活动范围的那些信息需求相符。

3 在 5.1 中规定了映射表，该映射表显示了使用本标准的集成资源怎样去满足这些信息需求。该集成资源的使用引进了附加要求，这对于应用协议是常见的。

4.1 功能单元

本条规定了对于机械零件或装配件的配置控制三维设计应用协议的功能单元。本标准规定了如下功能单元：

——advanced _boundary _representation(高级的边界表示)；

——authorization(授权)；

——bill _of _material(物料清单)；

——design _activity _control(设计活动控制)；

——design _information(设计信息)；

——effectivity(有效性)；

——end _item _identification(截止项标识)；

——facetted _boundary _representation(小平面边界表示)；

——manifold _surface _with _topology(拓扑流形曲面)；

——non _topological _surface _and _wireframe(无拓扑曲面与线框)；

——part _identification(零件标识)；

——shape(形状)；

——source _control(资源控制)；

——wireframe _with _topology(拓扑线框)。

下面给出这些功能单元(UoF)及每个功能单元所支持的功能描述。4.2 中定义了这些功能单元所包括的应用对象。

4.1.1 高级的边界表示

advanced _boundary _representation UoF(高级的边界表示功能单元)相当于使用高级的边界表示实体模型来表示零件的形状。此表示便于定义曲线、曲面及约束它们的拓扑。仅通过拓扑，清楚地定义了边界。定义零件形状的所有几何都必须与拓扑有关。

advanced _boundary _representation UoF 使用了下列应用对象：

——Advanced _B _rep(高级的 B-rep)。

4.1.2 授权

authorization UoF(授权功能单元)相当于在特定日期，在特定状态下，由一组对该授权具有准许作用的人作指示对一条产品数据进行验收。

authorization UoF 使用了下列应用对象：

—Approval(批准)；

——Person _organization(人员组织)。

4.1.3 物料清单

bill _of _material UoF(物料清单功能单元)包括必须表示制造一零件所要求的材料或构件的结构表。

bill _of _material UoF 使用了下列应用对象：

——Alternate _part(备用零件)；

——Component _assembly _position(构件装配位置)；

——Engineering _assembly(工程装配)；

——Engineering _make _from(工程参照件)；

——Engineering _next _higher _assembly(工程下一层高级装配)；

——Engineering _promissory _usage(工程约定使用)；

——Substitute _part(代用零件)。

4.1.4 设计活动控制

design _activity _control UoF(设计活动控制功能单元)包括描述零件版本历史的信息。它能标识初期零件的设计要求以及为修改零件版本的更改要求。它为在这些初期的或更改了的零件版本的开发工作提供指导。

design _activity _control UoF 使用了下列应用对象：

——Change _order(更改单)；

——Change _request(更改请求)；

——Start _order(开始单)；

——Start _request(开始请求)；

——Work _order(工作单)；

——Work _request(工作请求)。

4.1.5 设计信息

design _information UoF(设计信息功能单元)相当于一零件的信息集合，该信息集合是从该零件的标识和形状信息中分离出来的，它定义了完整零件必须具备的特定要求。这信息能标识与该零件的设计和制造相关的规范以及对那些规范应用的限制。

design _information UoF 使用了下列应用对象：

——Additional _design _information(附加设计信息)；

——Design _specification(设计规范)；

——Material _specification(材料规范)；

——Process _specification(工艺规范)；

——Specification(规范)；

——Surface _finish _specification(表面精加工规范)；

——Usage _constraint(使用约束)。

4.1.6 有效性

effectivity UoF(有效性功能单元)相当于在一产品模型中关于各构件计划使用的信息。

effectivity UoF 使用了下列应用对象：

——Planned _date _effectivity(计划日期有效性)；

——Planned _effectivity(计划有效性)；

——Planned _lot _effectivity(计划批量有效性)；

——Planned _sequence _effectivity(计划工序有效性)。

4.1.7 截止项标识

end _item _identification UoF(截止项标识功能单元)包括关于不同产品的信息，该产品是由某组织经销给其用户的。它为某组织控制这些产品的配置提供信息结构。

end _item _identification UoF 使用了下列应用对象：

——Product _configur ation(产品配置)；

——Product _model(产品模型)。

4.1.8 小平面边界表示

faceted _boundary _representation UoF(小平面边界表示功能单元)包括一零件的形状表示，该实体模型的边界曲面是二维曲面。在此表示中仅使用了点和平面多边形。对于此表示许多拓扑信息是隐含的。

facetted _boundary _representation UoF 使用了下列应用对象：

——Facetted _B _rep(小平面 B-rep)。

4.1.9 拓扑流形曲面

manifold _surface _with _topology UoF(拓扑流形曲面功能单元)相当于是用拓扑流形曲面进行零件的形状表示。用三维曲线、曲面和拓扑来定义该零件的外边界。

manifold _surface _with _topology UoF 使用了下列应用对象：

——Manifold _surface _with _topology(拓扑流形曲面)。

4.1.10 无拓扑曲面与线框

non _topological _surface _and _wireframe UoF(无拓扑曲面与线框功能单元)相当于是用无拓扑几何曲面或线框进行零件的形状表示。这些表示只用点、曲线和曲面来构成。这些曲线的边界由该曲线上的点与点和曲线之间的关系直接定义。这些曲面的边界由该曲面上的曲线与曲线和曲面之间的关系直接定义。应该明确裁剪曲面和曲线，除非它们是闭合的。

non _topological _surface _and _wireframe UoF 使用了下列应用对象：

——Non _topological _surface _and _wireframe(无拓扑曲面与线框)。

4.1.11 零件标识

part _identification UoF(零件标识功能单元)相当于结构，通过该结构能够定义一零件、一零件的特殊版本以及一零件不同设计的描述视图。

part _identification UoF 使用了下列应用对象：

——Design _discipline _product _definition(设计科目产品定义)；

——Part(零件)；

——Part _version(零件版本)。

4.1.12 形状

shape UoF(形状功能单元)相当于零件的几何和拓扑定义。这个功能单元允许该零件的各部分形状有不同类型的几何表示。当零件的这些部分集中在一起时就构成该零件的完整形状。

shape UoF 使用了下列应用对象：

——Geometric _model _representation(几何模型表示)；

——Shape(形状)；

——Shape _aspect(形状特征)

4.1.13 资源控制

source _control UoF(资源控制功能单元)包括关于某组织的信息，该组织具有决定生产某特定零件的资格。

source control UoF 使用了下列应用对象：

Supplier(供应商)；

——Supplied _part _version(供应零件版本)。

4.1.14 拓扑线框

wireframe _with _topology UoF(拓扑线框功能单元)相当于零件的形状表示，此零件的形状表示使用边拓扑定义的线框。其包括三维曲线和拓扑。

wireframe _with _topology UoF 使用了下列应用对象：

Wireframe _with _topology(拓扑线框)。

4.2 应用对象

本条规定了对于机械零件或装配件的配置控制三维设计应用协议的应用对象。每个应用对象都是一个原子元素，其与唯一的应用概念连接在一起，并且包括详细说明该对象数据元素的属性。下面给出了这些应用对象及其定义。

4.2.1 附加设计信息

Additional _design _information(附加设计信息)是与产品设计有关的技术规范集合。

例1：在产品要求的形式上，透射率、吸收率和反射率的概念是附加设计信息。通过参考与该零件相关的带应用限制的详细说明能够使我们了解后续的各工艺，诸如：机械加工和测试。

4.2.2 高级的边界表示

Advanced B _rep(高级的边界表示)是一种 Geometric _model _representation(见 4.2.15)的类型。它通过高级边界表示实体模型表示一零件的形状或一零件形状的外形。此表示考虑到曲线和曲面的定义以及约束它们的拓扑，并仅通过拓扑直接定义边界。所有定义零件形状或零件形状特征的几何都必须与拓扑相关。

4.2.3 备用零件

Alternate _part(备用零件)是一个在形状、配合以及功能上可以与其他零件相互交换的零件。

注：组织对其拥有的备用零件的使用不感兴趣，则不追踪它。

例2：设计要求一个一定尺寸的钣金螺纹。钣金螺纹由不同的组织制造，但是该零件尺寸的形状、配合与功能是相同的。不同的钣金螺纹由制造者给予不同的零件编号。设计组织不跟踪在实际装配中被使用的钣金螺纹。该钣金螺纹就是备用零件。

4.2.4 批准

Approval(批准)是在一个组织内对一条产品数据同意或不同意的标识。

与 Approval 有关的数据如下：

——Date(日期)；

——Purpose(目的)；

——Status(状态)。

4.2.4.1 日期

date(日期)规定了批准生效或将生效的时间。

4.2.4.2 目的

purpose(目的)规定了批准的原因。

4.2.4.3 状态

status(状态)规定了应用一条产品数据或多条产品数据之间关系的同意状态。批准状态的有效值是批准和不批准。批准意味着同意该要求；不批准意味着不同意该要求。

4.2.5 更改单

Change_order(更改单)是 Work_order(见 4.2.40)的一种类型，它授权一零件的更改设计的开发，并导致产生一新的零件版本。

与 Change_order 有关的数据如下：

——adopted_solution(采用方案)；

——change_date(更改日期)。

4.2.5.1 采用方案

adopted_solution(采用方案)对于一个 Change_request(见 4.2.6)规定了从提出的方案集合中选择一种方案。

4.2.5.2 更改日期

change_date(更改日期)规定了 Change_order 的生效日期。

4.2.6 更改请求

Change_request(更改请求)是 Work_request(见 4.2.40)的一种类型，它标识了修改一零件设计应做工作的建议范围。

与 Change_request 有关的数据如下：

——consequence(结果)；

——recommended_solution(推荐方案)；

——version(版本)。

4.2.6.1 结果

consequence(结果)规定了零件有效版本方面在操作、功能性或一特定推荐方案的形状中的有效作用。对于一个 Change_request 可以有多个结果。

例 3：对一设计的一个更改可能是加厚一架飞机机翼上的一个支承。这增加厚度的结果将使该机翼现在能承受 x 量的增加负荷，该飞机将能多携带 y 量的重量。

4.2.6.2 推荐方案

通过 Work_request 描述(见 4.2.41)recommended_solution(推荐方案)规定了一潜在的可能方案，它使在 Change_request 中描述的要求得到满足。对于一个 Change_request 可以有多个 recommended_solution。

4.2.6.3 版本

version(版本)对一更改请求的每次反复规定了一唯一标识符。

4.2.7 构件装配位置

Component_assembly_position(构件装配位置)是装配件中其构件位置的标识。

与 Component_assembly_position 有关的数据如下：

——transformation(变换)。

4.2.7.1 变换

transformation(变换)规定了在装配件几何空间中其构件的移动和方向。

4.2.8 设计科目产品定义

一个 Design_discipline_product_definition(设计科目产品定义)是一 Part_version(见 4.2.20)的组织定义之一或其视图之一。

与 Design_discipline_product_definition 有关的数据如下:

——CAD_filename(CAD 文件名);

——creation_date(开发日期);

——description(描述);

——discipline_id(科目标识)。

注:用该实体可以捕捉一特定的 Part_version 的其开发的任何中间阶段的定义。在 Part_version 等级上,某组织没有正式追踪那里的定义,用它可以捕捉一产品定义循环中的各个阶段。

例 4:定义循环的各阶段可以是一个物料清单视图,一个有限元模型视图,一个功能系统视图或一个制造视图。

4.2.8.1 CAD 文件名

CAD_filename(CAD 文件名)规定了在一计算机辅助设计系统中包含有几何零件描述的文件名。对于一特定的 Design_discipline_product_definition 不必规定此 CAD_filename。如果提供了它的名称,则假定该文件包括该零件形状的外部定义,其仅在设计该零件的组织内有意义。

4.2.8.2 创建日期

creation_date(创建日期)规定了第一次定义 Design_discipline_product_definition 的日期和时间。

4.2.8.3 描述

description(描述)规定了该产品一特定定义的目的。

4.2.8.4 科目标识

discipline_id(科目标识)规定了与产品定义匹配的视图或阶段。

4.2.9 设计规范

Design_specification(设计规范)是建立零件设计要求的 Specification(见 4.2.31)的一种类型。这些要求不是由其他设计或构造特点或参考数据决定的。

例 5:在一 Design_specification 要求集合中出现的可以是重量限制、尺寸限制、反射能力、色彩、透明度、触觉和外观。

4.2.10 工程装配

Engineering_assembly(工程装配)是一装配件和一构件或子装配件间的 parent_child(父子)关系。每个 Engineering_assembly 可以是下列情况之一:—Engineering_next_higher_assembly(见 4.2.12),或—Engineering_promissory_usage(见 4.2.13)。

与 Engineering_assembly 有关的数据如下:

——security_code(安全代码);

注:本标准的范围限定在产品设计定义物料清单的交换方面。它不包括某组织要求的所有物料清单。

4.2.10.1 安全代码

security_code(安全代码)规定了在一装配件中一构件的安全分类。

注:在下面的一些情况下,这个构件的安全代码与 Part_version(见 4.2.20)相关的特定 security_code 不同,即当零件作为与此安全代码有关的某装配件中的构件时,因其与一装配件相关,从而改变了它自己的安全级别。该零件当其独立存在时,即当它使用在任何特定装配件之外时,可以有一安全代码。然而,此处规定的安全代码定义了在这非常特殊的相关环境中的零件的安全代码。

例6:一轮子可以用在一些汽车产品中,其本身的安全分类没有定级。同样的轮子在一装配件中使用,然后该装配件又用在一试验的、高灵敏度的、处于开发阶段的新汽车中。尽管该轮子其本身的安全分类没有定级,但是在新汽车中该装配件被定为最高级,事实上该装配件中的这个特定轮子也必须定为最高级。当用该轮子作为正在开发的这个特定汽车的装配件中的一个构件时,这个特性重新定义了该轮子的安全分类为最高级。

4.2.11 工程参照件

一个 Engineering_make_from(工程参照件)是指两个零件间的关系,其中一零件被用来作为另一零件设计的基础。这个关系由设计给定,因此它可以通过该产品的生命周期跟踪。

例7:A 公司设计一个装配件使用了 B 公司的一个杯座,并对 B 公司设计的杯座,它添加了两个额外的配合孔。在这种情况下,A 公司把 B 公司设计的杯座标识为一个 Engineering_make_from。

4.2.12 工程下一层更高装配

Engineering_next_higher_assembly(工程下一层更高装配)是 Engineering_assembly(见 4.2.10)的一种类型。它是在装配谱系中一零件与它直接父件的关系。

与 Engineering_next_higher_assembly 有关的数据如下:

——as_required(按所需);

——component_quantity(构件数量);

——reference_designator(引用标志);

——unit_of_measure(度量单位)。

4.2.12.1 按所需

as_required(按所需)属性规定构件数量是否被明确地定义或在一特定使用中是否必须定量。

例8:如果要标定制造一零件消耗的金属板量,可以规定一卷,在这里,一是量,卷是度量单位。当要求时,也可使用多卷金属板。在这种情况下,要求的是量,而卷还是度量单位。

4.2.12.2 构件数量

如果 as_required 是不可靠的,则由 component_quantity(构件数量)规定在紧接的更高装配件中可见到的某一构件的数量。

4.2.12.3 引用标志

reference_designator(引用标志)规定了一个唯一标识符。该标识符用于区别一构件在一装配件中的一种特定情况,在那个装配件中使用了多个特定的零构件。

例9:如果在一汽车部件中使用了四个同样的轮子,然后作为例子,用 reference_designator 把左前轮与其他轮子区分开。

4.2.12.4 度量单位

在表达 component_quantity(构件数量)术语中,unit_of_measure 规定了固定的量。

4.2.13 工程约定使用

Engineering_promissory_usage(工程约定使用)是 Engineering_assembly(见 4.2.10)的一种类型。它是一零件与一更高装配的关系,在那里还没有定义该零件的下一层更高装配。

例10:在飞机机身里为了支承一梁结构需要一些支柱。设计者正在设计其中一个支柱,设计者需要规定此支柱在该产品结构中的预计使用。但该设计者还不知道下一层更高装配的情况,因为在该产品结构中还没有定义必须配合有该支柱的梁。然而,设计者了解要使用这些支柱的机身。使用这个对象,设计者能够规定在机身支承的子装配中这些约定支柱。

4.2.14 小平面 B_rep

Facetted_B_rep(小平面 B_rep)是 Geometric_model_representation(见 4.2.15)的一种类型,它借助一小平面边界表示实体模型来表示一零件的形状或一零件形状的外形。该表示允许形状的定义通过二维曲面作其定界曲面来表示。在这个表示中仅使用了点和平面封闭折线。对于此表示许多拓扑信息

是隐含的。壳只用封闭折线限定的面组成。

4.2.15 几何模型表示

Geometric_model_representation(几何模型表示)是一零件形状或某一部分形状的定义。每个Geometric_model_representation可以是以下的之一:一个Advanced_B_rep(见4.2.2);一个Facetted_B_rep(见4.2.14);一个Non_topological_surface_and_wireframe(见4.2.18);一个Manifold_surface_with_topology(见4.2.16);或一个Wireframe_with_topology(见4.2.39)。

4.2.16 拓扑流形曲面

Manifold_surface_with_topology(拓扑流形曲面)是Geometric_model_representation(见4.2.15)的一种类型,它使用拓扑流形曲面表示一零件的形状或一零件形状的外形。用三维曲线、曲面和拓扑来定义零件的外部边界。

4.2.17 材料规范

Material_specification(材料规范)是Specification(见4.2.31)的类型之一,它规定了在一产品加工中所用原材料、配料或半加工材料的使用性能。

4.2.18 无拓扑曲面与线框

Non_topological_surface_and_wireframe(无拓扑曲面与线框)是Geometric_model_representation(见4.2.15)的一种类型,它使用无拓扑的曲面或线框几何表示一零件的形状或零件形状的各部分。这些表示仅仅通过使用点、曲线和曲面来构成。实际上,曲线的边界由该曲线上的各点和这些点与这些点所限定的曲线之间的明确关系定义。曲面的边界由该曲面上的曲线和这些曲线与这些曲线所限定的曲面之间的明确关系来定义。除非曲面和曲线是闭合的,否则应明确剪裁。

4.2.19 零件

Part(零件)是计划生产的或在一产品加工中使用的单元。

与Part有关的数据如下:

part classification(零件分类);

part_nomenclature(零件名称);

part number(零件号);

part type(零件类型);

standard part indicator(标准件标志)。

4.2.19.1 零件分类

part_classification(零件分类)规定了相关零件的一个家族,该零件家族在制造中使用相同的加工工艺。对于一特定零件没必要规定其Part_classification。

例11:机加工的、冲加工的或车削加工的零件可分成独立的零件级别。

4.2.19.2 零件名称

part_nomenclature(零件名称)规定了一个名字,在组织内一般通过此名字了解该零件。

注:该名字也能在任何单个组织外有意义。例如,一个国家或企业对同样零件可能有不同的命名习惯。

4.2.19.3 零件号

对于一特定组织,该part_number(零件号)规定了一零件的唯一标识符。

4.2.19.4 零件类型

part type(零件类型)规定了零件种类的一个集合。本标准标识的零件种类是:元件、装配件、外购件或不可拆装配件。那些Part_type是元件的零件指的是物料清单中底层的零件。其Part_type是装配件的零件指的是为满足一特定功能把其他零件装在一起所组成的零件。其Part_type是外购件的零件指的是对于来自一个支配源的设计已经被加工完成的零件。其Part_type是不可拆装配件的零件是指一种装配件,该装配件在装配以后不能再被拆装了,否则至少会给组成该装配件的构件之一带来物理损伤。

4.2.19.5 标准件标志

standard_part_indicator(标准件标志)规定了在定义某装配件的物料清单中一零件是否有外定设计。

注：对于一个公司、企业或其他组织，一个零件可以是一个标准件。

4.2.20 零件版本

Part_version(零件版本)是在一零件的设计已经过正式发布或更改后，对该零件的描述标识。只有由负责该零件形状定义的组织正式跟踪的更改才会导致形成此应用对象的一个新实例。由负责该零件形状定义的某组织不正式跟踪的重复设计不会导致形成此应用对象的一个新实例。但其将由 Design_discipline_product_definition 应用对象跟踪。

与 Part_version 有关的数据如下：

——contract_number(合同号)；

——make_or_buy_code(制造或购买代码)；

——release_status(发放状态)；

——revision_letter(修改文字)；

——security_code(安全代码)。

4.2.20.1 合同号

contract_number(合同号)规定了零件进行设计所依据的商业合同。对于一个唯一的 Part_version 不必规定其 contract_number。如果依据一合同已经开发了该设计，那么此属性将规定是那个合同的唯一标识。

4.2.20.2 制造或购买代码

make_or_buy_code(制造或购买代码)规定了设计部门为获得该零件的计划。它将永远是制造或购买这两个值的其中之一。如果这属性的值是制造，那么零件设计部门已经计划好在内部制造该零件；如果这属性值是购买，那么零件设计部门就计划从一卖主购买该零件。

4.2.20.3 发放状态

此 release_status(发放状态)规定了与设计信息发布相关联的一零件的版本状态。它将永远是发放或未发放这两个值的其中之一。发放是指那些版本已经被重新审查并被批准进一步使用。未发放是指那些版本没经过重新审查或没被批准继续使用。

4.2.20.4 修改文字

此 revision_letter(修改文字)规定了一零件特定版本的唯一标识。

4.2.20.5 安全代码

此 security_code(安全代码)规定了一零件特定版本的安全分类。

4.2.21 人员组织

Person_organization(人员组织)是独立个人和其特定组织的标识，在该特定组织中其独立个人起作用。

与 Person_organization 有关的数据如下：

——address(地址)；

——organization(组织)；

——person(人员)。

——person_organization_id(人员组织标识)

4.2.21.1 地址

address(地址)规定了书面的和(或)电子的邮件发送路线，以及在某些情况下一人员和组织的实际地址的发送路线。对于一特定的 Person_organization 不必规定其 address。

4.2.21.2 组织

organization(组织)规定了一个为了一个或更多的共同目的组织在一起的人民团体。

4.2.21.3 人员

person(人员)规定了一独立存在的人。

4.2.21.4 人员组织标识

person_organization_id(人员组织标识)规定了该 person_organization 的唯一标识。

4.2.22 计划日期有效性

Planned_date_effectivity(计划日期有效性)是 Planned_effectivity(见 4.2.23)的一种。它是个规范,该规范由在 Product_configuration 中对一产品的使用预先计划好的设计组织规定。通过一个或两个相关日期确定在该产品配置中此零件的预期使用。

与 Planned_date_effectivity 有关的数据如下:

——end_date(截止日期);

——start_date(开始日期)。

4.2.22.1 截止日期

end_date(截止日期)规定了由设计组织计划在 Product_configuration 中使用该零件的最后日期。对于一特定的 Planned_date_effectivity 不必规定其 end_date。在变更的时间上无 end_date 意味着也没必要标识 end_date。

4.2.22.2 开始日期

start_date(开始日期)规定了在 Product_configuration 中使用该零件的初始日期。

4.2.23 计划有效性

Planned_effectivity(计划有效性)是在一产品的特定配置中一零件的预定使用。每个 Planned_effectivity 不是一个 Planned_date_effectivity(见 4.2.22)、Planned_lot_effectivity(见 4.2.24),就是一个 Planned_sequence_effectivity(见 4.2.25)。

4.2.24 计划批量有效性

Planned_lot_effectivity(计划批量有效性)是 Planned_effectivity(见 4.2.23)的一种。它是在一 Product_configuration 中一零件的使用规范,在产品配置中该零件作为一组中的一个被生产。当零件被分批生产时,或者在生产过程中重要的指标可能改变时使用批量。

与 Planned_lot_effectivity 有关的数据如下:

——lot_numb er(批量号);

——lot_size(批量大小);

—— lot_size_unit_of_measure(批量单位)。

4.2.24.1 批量号

lot_number(批量号)规定了构成一个批次的零件组的标识。

4.2.24.2 批量大小

lot_size(批量大小)规定了该批次中零件的数量。

4.2.24.3 批量单位

lot_size_unit_of_measure(批量单位)规定了表达 lot_size 的固定量值单位。

4.2.25 计划顺序有效性

Planned_sequence_effectivity(计划顺序有效性)是 Planned_effectivity(见 4.2.23)的一种。它是在根据计划的顺序号码标识的 Product_configurations 范围中由一零件的设计组织预期使用的规范。

与 Planned_sequence_effectivity 有关的数据如下:

——component_quantity(构件数量);

——from_effectivity_id(有效性起始标识);

——quantity_unit_of_measure(数量度量单位)；

——thru_effectivity_id(有效性结束标识)。

4.2.25.1 构件数量

component_quantity(构件数量)规定了在一配置中对一特定零件有效的元件数量。

4.2.25.2 有效性起始标识

from_effectivity_id(有效性起始标识)规定了 Planned_sequence_effectivity 范围的开始计划的顺序号码。

4.2.25.3 数量度量单位

quantity_unit_of_measure(数量度量单位)规定了表达 component_quantity 的度量单位。

例 12:quantity_unit_of_measure 可以是卷、张或条,以及其他东西之一。

4.2.25.4 有效性结束标识

thru_effectivity_id(有效性结束标识)规定了 Planned_sequence_effectivity 范围的结束计划的顺序号码。对于一特定的 Planned_sequence_effectivity 不必规定其 thru_effectivity_id。

4.2.26 工艺规范

Process_specification(工艺规范)是 Specification(见 4.2.31)的一种,它是对一产品或材料完成的操作所作的标识。

例 13:典型加工类型指热处理、焊接、电镀、包装和作标记。

4.2.27 产品配置

Product_configuration(产品配置)是 Product_model(见 4.2.28,4.2.28.1)的一个变种。配置控制以这个应用对象为基础。

与 Product_configuration 有关的数据如下:

——item_id(标识项)；

——phase_of_product(产品阶段)。

例 14:F14 军用飞机的一个 Product_configuration 是 F14 的 D 配置。该组织已经定义了 F14 的四种配置 A,B,C 和 D。在此情况下该配置是 D。一单个产品模型可以有许多不同的配置以便该组织标识它的不同变化。

4.2.27.1 标识项

item_id(标识项)规定了一个 Product_model 变体的唯一标识。

4.2.27.2 产品阶段

phase_of_product(产品阶段)规定了产品生命周期中的一个阶段,在此阶段中该设计的特定版本正在变换。

例 15:例如,在一组织内一产品的四个阶段可以是技术与基本原理开发、开发与改良、制造与装配批准以及生产。

4.2.28 产品模型

Product_model(产品模型)是组织提供给其用户的产品。在一产品的设计阶段为了计划需要标识该 Product_model。

与 Product_model 有关的数据如下:

——model_name(模型名)。

4.2.28.1 模型名

model_name(模型名)规定了由一组织对其提供给用户的一产品所给定的唯一标识。

例 16:F14 是一军用飞机这一 Product_model 的标识。

4.2.29 形状

Shape(形状)是一零件形态的数学描述。

4.2.30 形状特征

Shape_aspect(形状特征)是一零件形状的一明显部分。一零件的形状由一个或多个 Shape_aspects 组成。

注：在设计中常常根据该零件不同区域形状的复杂性用一些不同类型的几何模型定义零件形状。这每个区域都可标识为一个 Shape_aspect。

4.2.31 规范

Specification(规范)是一个文档，它包括与零件的半成品或成品必须具备的特定品质有关的定义、工艺和法规。每个 Specification 不是一 Design_specification(设计规范)(见 4.2.9)、一 Material_specification(材料规范)(见 4.2.17)、一 Process_specification(工艺规范)(见 4.2.26)，就是一 Surface_finish_specification(表面精加工规范)(见 4.2.37)。

与 Specification 有关的数据如下：

——specification_code(规范代码)；

——specification_source(规范源)。

例 17：参考其他文档可知：材料、表面精加工和工艺规范这些事物的所有示例其应用时每次都能够用隐含描述，而不是用明确描述。

4.2.31.1 规范代码

specification_code(规范代码)规定了一 Specification 的唯一标识。该码在源组织中是唯一的。

4.2.31.2 规范源

specification_source(规范源)规定了对该 specification 负责的组织。

4.2.32 初始单

Start_order(初始单)是 Work_order(见 4.2.40)的一种，其授权开发一零件的初期设计并导致产生一个零件原始版本。

4.2.33 初始请求

Start_request(初始请求)是 Work_request(见 4.2.41)的一种，其标识了一零件初期设计所要作的工作。

4.2.34 代用零件

Substitute_part(代用零件)是一装配件中的一个构件，其形状、配合和功能可以与备用件不同，但它能够满足该装配相关环境中其他零件的要求。

注

1 不要求一 substitute_part 具有和原零件相同的形状或数量。

2 用此应用对象也可以消除版本积累，即当创造一低层构件的一新版本时，所有更高装配要重新标识。

3 此应用对象只允许一种方式代用。在给定的相关环境下，如果 A 是 B 的一个代用件，那么装配时 B 不能作为 A 的代用件，除非这个实体在另一情况中确实存在。如果 A 是 C 相关环境中 B 的代用件，那么在 C 的一最后建立的实物单元中必须使用 A 或 B，但二者不能同时使用。

例 18：120 磅/每平方英寸±12 磅/每平方英寸的一量具能够用 125 磅/每平方英寸±6.25 磅/每平方英寸的一量具代替，其提供的增量相等并且除了它测量的物品之外，其输入和输出尺寸相同。

4.2.35 外购件版本

Supplied_part_version(外购件版本)是一个通过它的零件号和它的供应商所定义的零件。

与 Supplied_part_version 有关的数据如下：

——certification_required(所需认证)；

——supplier_part_number(外购零件号)。

4.2.35.1 所需认证

certification_required(所需认证)规定从供应商手中获得一零件之前其供应商本身要先被批准。

4.2.35.2 外购零件号

supplier_part_number(外购零件号)规定了在生产组织中使用的该零件号码。对于一特定的 Supplied_part_version 不必规定其 Supplier_part_number。

例 19:A 公司使用 1234567-1 号零件作为一装配件的一个零件。这零件实际上从 B 公司购买,但是为保证其遇到的该装配件的各个要求,对于每个接收到的附加测试都要完成。B 公司把这个零件标识为 AABBCCD-E。A 公司用供应商号码 52088 标识 B 公司。在公司 A 里,标识该零件的零件号是 1234567-1。其供应商是 B 公司,它的号码是 52088,而其供应的零件号码是 AABBCCD-E。

4.2.36 供应商

Supplier(供应商)是生产该零件或设计该零件的组织。

与 Supplier 有关的数据如下:

——supplier_id(供应商标识);

此 supplier_id 规定了该生产组织的唯一标识。这个标识可以是一组织的唯一号码、一企业的唯一号码或国际上公认的号码。

4.2.37 表面精加工规范

Surface_finish_specification(表面精加工规范)是 specification(见 4.2.31)的一种,它规定了适用于半成品或成品零件的表面组织或保护层的特性。

4.2.38 使用约束

Usage_constraint(使用约束)是一 specification(见 4.2.31)应用的一个限制。它关系到具有详细说明信息的一规范的一个特定题目或者关系到对于一特定实例在其应用中使用的文本。

与 Usage_constraint 有关的数据如下:

——element(元素);

——value(值)。

4.2.38.1 元素

element(元素)规定了正受限制或正被明确的一规范中的特定单元或标题。

4.2.38.2 值

value(值)规定了在一个使用约束的一个元素中与一特定单元或标题有关的限定或特定的数。

例 20:A 部分可以要求绘制每一个写表面精加工规范的公司。该总体词是:Specification_code=ABCD-1;Specification_source=XYZ 公司;Usage_constraint 的配对词是:Element=颜色;Value=绿;Element=层;Value=3。

4.2.39 拓扑线框

Wireframe_with_topology(拓扑线框)是 Geometric_model_representation(见 4.2.15)的一种,它使用线框表示了一零件的形状或零件形状的一部分,该线框定义了一隐含体积或通过边拓扑被剪裁。它包括三维曲线和定义一组顶点和边的图的拓扑。

4.2.40 工作单

Work_order(工作单)是一个文档,它授权了一零件初始的或改良的设计开发工作。每个 Work_order 不是一个 Start_order(见 4.2.32),就是一个 Change_order(见 4.2.5)。它是进行一个或多个 Work_request(见 4.2.41)的结果。

与 Work_order 有关的数据如下:

——additional_data(附加数据);

——analysis_data(分析数据);

——work_order_id(工作单标识)。

4.2.40.1 附加数据

additional_data(附加数据)规定了在工作请求提出的相关集的开发期间曾被编辑的附加适当信

息。对于一特定的 Work_order 不必规定其 additional_data。

4.2.40.2 分析数据

analysis_data(分析数据)规定了在一工作请求提出的相关集中为使设计或改良合法化所完成的评定结果。对于一特定的 Work_order 不必规定其 analysis_data。

4.2.40.3 工作单标识

work_order_id(工作单标识)规定了由 Work_order 所授权的工作的唯一标识。

4.2.41 工作请求

Work_request(工作请求)是一个文档,该文档标识了对于一零件的初始设计或重新定义设计所作的工作。每个 Work_request 不是一 Start_request,就是一 Change_request。

与 Work_request 有关的数据如下:

——description(描述);

——reason(理由);

——request_date(请求日期);

——status(状态);

——work_request_id(工作请求标识)。

4.2.41.1 描述

description(描述)规定了对开发或重新定义一零件的设计所作工作的说明。

4.2.41.2 理由

reason(理由)规定了产生 Work_request 的目的。

4.2.41.3 请求日期

request_date(请求日期)规定了建立 Work_request 的日期。

4.2.41.4 状态

status(状态)规定了该工作请求完成的当前状态。一 Work_request 的状态有效值是:提出、进行、发布及保持有效。

4.2.41.5 工作请求标识

work_request_id(工作请求标识)规定了关于一零件设计方面提出的工作的唯一标识。

4.3 应用要求

本条规定了机械零件或装配件配置控制三维设计应用协议的应用要求。应用要求规定了应用对象间的所有关系、该关系的重点、以及这些应用对象和功能单元的完整性和有效性所要求的法规。应用要求及其定义由下面给出。

4.3.1 附加设计信息与规范

每个 Additional_design_information(附加设计信息)是一个或多个 Specification(规范)对象的集合。每个 Specification 正好通过一个 Additional_design_information 来标识。

4.3.2 批准与人员组织

由一个或多个 Person_organization(人员组织)对象认可每个 Approval(批准)。每个 Person_organization 对象认可零个、一个或多个 Approval 对象。

4.3.3 设计科目产品定义与附加设计信息

每个 Design_discipline_product_definition(设计科目产品定义)有零个、一个或多个 Additional_design_information(附加设计信息)对象。每个 Additional_design_information 适用于一个或多个 Design_discipline_product_definition 对象。

4.3.4 设计科目产品定义与批准

每个 Design_discipline_product_definition(设计科目产品定义)正好通过一个 Approval(批准)被批准。每个 Approval 批准一个或多个 Design_discipline_product_definition 对象。

4.3.5 设计科目产品定义与工程装配

用每个 Design_discipline_product_definition(设计科目产品定义)作为零个、一个或多个 Engineering_assembly(工程装配)对象中的一个构件。每个 Engineering_assembly 构件正好通过一 Design_discipline_Product_definition 来定义。

每个 Design_discipline_product_definition 是零个、一个或多个 Engineering_assembly 对象的装配件。每个 Engineering_assembly 正好通过一 Design_discipline_product_definition 来定义。

4.3.6 设计科目产品定义与工程参照

每个 Design_discipline_product_definition(设计科目产品定义)是零个、一个或多个 Engineering_make_from(工程参照)对象中的基础设计。每个 Engineering_make_from 正好有一 Design_discipline_product_definition 的结果设计。

每个 Design_discipline_product_definition 是零个、一个或多个 Engineering_make_from 对象中的结果设计。每个 Engineering_make_from 正好有一 Design_discipline_product_definition 的结果设计。

4.3.7 工程装配与计划有效性

每个 Engineering_assembly(工程装配)在零个、一个或多个 Planned_effectivity(计划有效性)对象中是有效的。每个 Planned_effectivity 正好作用于一个 Engineering_assembly。

4.3.8 工程装配与代用零件

每个 Engineering_assembly(工程装配)有零个、一个或多个 Substitute_part(代用零件)对象的代用件。每个 Substitute_part 必须正好是一个 Engineering_assembly 中一零件的代用件。

4.3.9 工程下一层更高装配与构件装配位置

每个 Engineering_next_higher_assembly(工程下一层更高装配)设置在零个或一个 Component_assembly_position(构件装配位置)上。每个 Component_assembly_position 必须正好有一个 Engineering_next_higher_assembly。

4.3.10 几何模型表示与构件装配位置

每个 Geometric_model_representation(几何模型表示)表示了零个、一个或多个 Component_assembly_position(构件装配位置)对象中的构件。每个 Component_assembly_position 正好是一 Geometric_model_representation 中的一个构件。

每个 Geometric_model_representation 表示了零个、一个或多个 Component_assembly_position 对象中的一个装配件。每个 Component_assembly_position 是正好在一 Geometric_model_representation 中的一个装配件。

4.3.11 零件与备用零件

每个 Part(零件)有零个、一个或多个 Alternate_part(备用零件)对象的备用件。每个 Alternate_part 是一个或多个 part 对象的备用件。

4.3.12 零件与零件版本

每个 Part(零件)有一个或多个 Part_version(零件版本)对象。一零件的每个 Part_version 应定义一 Part 的实际变化。

4.3.13 零件与代用零件

每个 Part(零件)在一特定使用中都有零个、一个或多个 Substitute_part(代用零件)对象的代用件。在一特定使用中每个 Substitute_part 是一个零件对象并且代用了一个或多个 Part。

4.3.14 零件版本与批准

每个 Part_version(零件版本)正好通过一个 Approval(批准)被批准。每个 Approval 批准一个或多个 Part_version 对象。

4.3.15 零件版本与设计学科产品定义

每个 Part_version(零件版本)通过一个或多个 Design_discipline_product_definition(设计科目产品定义)对象来定义。每个 Design_discipline_product_definition 正好是一个 Part_version 的一个定义。

4.3.16 零件版本与外购零件版本

每个 Part_version(零件版本)作为一个或多个 Supplied_part_version(外购零件版本)对象被标识。每个 Supplied_part_version 正好与一个 Part_version 相对应。

4.3.17 人员组织与设计科目产品定义

每个 Person_organization(人员组织)是零个、一个或多个 Design_discipline_product_definition(设计科目产品定义)对象的创造者。每个 Design_discipline_product_definition 正好通过一个 Person_organization 创造。

4.3.18 人员组织与零件

每个 Person_organization(人员组织)是零个、一个或多个 Part(零件)对象的拥有者。每个 Part 由正好一个 Person_organization 所拥有。

4.3.19 人员组织与零件版本

每个 Person_organization(人员组织)是零个、一个或多个 Part_version(零件版本)对象的创造者。每个 Part_version 正好通过一个 Person_organization 创造。

4.3.20 人员组织与供应商

每个 Person_organization(人员组织)标识了零个、一个或多个 Supplier(供应商)对象。每个 Supplier 正好通过一个 Person_organization 来标识。

4.3.21 计划有效性与批准

每个 Planned_effectivity(计划有效性)正好通过一个 Approval(批准)被批准。每个 Approval 批准一个或多个 Planned_effectivity 对象。

4.3.22 产品配置与批准

每个 Product_configuration(产品配置)正好通过一个 Approval(批准)被批准。每个 Approval 批准一个或多个 Product_configuration 对象。

4.3.23 产品配置与零件

每个 Product_configuration(产品配置)通过零个、一个或多个 Part(零件)对象来满足。每个 Part 满足零个、一个或多个 Product_configuration 对象。

4.3.24 产品配置与计划有效性

每个 Product_configuration(产品配置)有零个、一个或多个 Planned_effectivity(计划有效性)对象。每个 Planned_effectivity 正好标识一个 Product_configuration。

4.3.25 产品模型与产品配置

每个 Product_model(产品模型)有一个或多个 Product_configuration(产品配置)对象。每个 Product_configuration 正好是一个 Product_model 的配置。

4.3.26 形状与设计科目产品定义

每个 Shape(形状)定义了一个或多个 Design_discipline_product_definition(设计科目产品定义)的几何特征。每个 Design_discipline_product_definition 有通过零个或一个 Shape 定义的几何特征。

4.3.27 形状与几何模型表示

每个 Shape(形状)表示了零个、一个或多个 Geometric_model_representation(几何模型表示)对象。每个 Geometric_model_representation 是零个、一个或多个 Shape 对象的表示。

4.3.28 形状与形状特征

每个 Shape(形状)由零个、一个或多个 Shape_aspect(形状特征)对象组成。每个 Shape_aspect 正好是一个 Shape 的一部分。

4.3.29 形状特征与几何模型表示

每个 Shape_aspect(形状特征)通过一个或多个 Geometric_model_representation(几何模型表示)对象来表示。每个 Geometric_model_representation 正好表示了一个 Shape_aspect。

4.3.30 形状特征与规范

每个 Shape_aspect(形状特征)有由零个、一个或多个 Specification(规范)对象描述的特征。每个 Specification 规定了零个、一个或多个 Shape_aspect 对象的一个特征。

4.3.31 规范与使用约束

每个 Specification(规范)通过零个、一个或多个 Usage_constraint(使用约束)来限定。每个 Usage_constraint 正好适用于一个 Specification。

4.3.32 外购零件版本与批准

每个 Supplied_part_version(外购零件版本)正好通过一个 Approval(批准)被批准。每个 Approval 批准零个、一个或多个 Supplied_part_version 对象。

4.3.33 供应商与外购零件版本

每个 Supplier(供应商)生产了零个、一个或多个 Supplied_part_version(外购零件版本)对象。每个 Supplied_part_version 正好通过一个 Supplier 生产。

4.3.34 工作单与批准

每个 Work_order(工作单)正好通过一个 Approval(批准)被批准。每个 Approval 批准一个或多个 Work_order 对象。

4.3.35 工作单与零件版本

每个 Work_order(工作单)适用于一个或多个 Part_version(零件版本)对象。每个 Part_version 是零个、一个或多个 Work_order 对象的结果。

4.3.36 工作单与工作请求

每个 Work_order(工作单)合并了一个或多个 Work_request(工作请求)对象。每个 Work_request 通过零个或一个 Work_order 对象来合并。

4.3.37 工作请求与批准

每个 Work_request(工作请求)正好通过一个 Approval(批准)被批准。每个 Approval 批准一个或多个 Work_request 对象。

4.3.38 工作请求与零件版本

每个 Work_request(工作请求)以一个或多个 Part_version(零件版本)对象为基础。每个 Part_version 通过零个、一个或多个 Work_request 对象被引用。

如果一 Work_request 要求比一个 Part_version 更多的工作,那么这些 Part_version 对象应该是独立 Part 对象。

4.3.39 工作请求与人员组织

每个 Work_request(工作请求)注意事项通过一个或多个 Person_organization(人员组织)对象来接收。把零个、一个或多个 Work_request 对象通知给每个 Person_organization。

5 应用解释模型

5.1 映射表

映射表表示了本标准每个功能单元及应用对象怎样映射到一个或几个 AIM(应用解释模型)源构件上(见 5.2)。

映射表由五列构成。这五列的内容是:

列 1)应用元素:应用元素的名字同它在 4.2 条应用对象定义中出现的一样。应用对象名用大写字母书写。属性名列在它所属的应用对象之后,并用小写字母书写。

列 2)AIM 元素:一个 AIM 元素名同它在 AIM(5.2)、术语"IDENTICAL MAPPING"或术语"

PATH"中出现的一样。AIM 实体用小写字母书写。AIM 实体的属性名被引用为〈实体名〉.〈属性名〉。一个应用元素可映射到几个相关的 AIM 元素。这些 AIM 元素中的每一个在映射表中占用一行。术语"IDENTICAL MAPPING"表示一个应用要求的两个应用对象映射到相同的 AIM 元素上。术语"PATH"表示应用要求映射到整个引用路径上。

列 3)源:对于那些来自集成资源的 AIM 元素,它表示 GB/T 16656 相应部分的序号。对于那些为本标准创建的 AIM 元素,则是被写部分的序号。

列 4)规则:给出一个或几个序号表示应用到当前 AIM 元素上的规则。所有涉及 AIM 元素的映射项引用从应用对象间关系所派生出来的规则。规则的扩展名列在表后。

列 5)引用路径:为了完整地描述一个 AIM 元素的映射,可能必须通过几个相关的 AIM 元素规定一个引用路径。一个 AIM 元素在引用路径列中占一行,并通过一个符号定义它与该列后续行 AIM 元素的关系。所以引用路径列表示一个 AIM 元素相对其后续行 AIM 元素的作用。如果提供一个引用路径,那么两个或两个以上相关 AIM 元素定义了该集成资源的解释,其满足 ARM 元素规定的要求。

对本标准创建的每一个 AIM 元素,都规定了一个从 GB/T 16656 集成资源到其 SUPERTYPE(超类型)的引用路径。在一个 AIM 元素的两个属性双向引用的情况下,每个属性产生一个引用路径,每个引用路径用一对括号括起来。作为两个引用路径根的 AIM 元素或者位于两对括号之间,或者位于每对括号之前。

为了表示引用路径,使用下列文法约定:

a) []:为了满足信息要求,需要多个 AIM 元素或引用路径部分。

b) ():为了满足信息要求,在映射中多个 AIM 元素或引用路径部分被标识为可选。

c) {}:括起来的部分约束引用路径以满足信息要求。

d) ->:属性引用实体或选择类型在下一行给出。

e) <-:实体或选择类型被下一行的属性所引用。

f) [i]:属性是一个聚集,它的一个属性在下一行给出。

g) [n]:属性是一个聚集,它的第 n 个成员在下一行给出。

h) =>:实体是下一行实体的超类。

i) <=:实体是下一行实体的子类。

j) =:限定字符串、选择或枚举类型为一个选项或值。

表 1　映射表—高级的边界表示功能单元

应用元素	应用解释模型元素	源	规则	引　用　路　径
ADVANCED_B_REP	advanced_brep_shape_representation	203	73	advanced_brep_representation<= shape_representation

表 2　映射表—授权功能单元

应用元素	应用解释模型元素	源	规则	引　用　路　径
APPROVAL	cc_design_approval	203	1,2,3	cc_design_approval<= approval_assignment
date	date_and_time	41	1,21	cc_design_approval<= approval_assignment approval_assignment. assigned_approval-> approval<- approval_date_time. dated_approval approval_date_time approval_date_time. date_time-> date_time_select=date_and_time date_and_time
purpose	approval. level	41		cc_design_approval<= approval_assignment approval_assignment. assigned_approval-> approval approval. level
status	approval_status	41	49,18	cc_design_approval<= approval_assignment approval_assignment. assigned_approval-> approval approval. status-> approval_status
approval to person_organization	PATH		2	cc_design_approval<= approval_assignment approval_assignment. assigned_approval-> approval<- approval_person_organization. authorized_approval

表 2(续)

应用元素	应用解释模型元素	源	规则	引 用 路 径
approval to person_organization	PATH		2	approval_person_organization approval_person_organization. person_organization->person_organization_select=person_and_organization person_and_organization<- person_and_organization_assignment. assigned_person_and_organization person_and_organization_assignment=> cc_design_person_and_organization_assignment
PERSON_ORGANIZATION	cc_design_person_and_organization_assignment	203	54,26	cc_design_person_and_organization_assignment<= person_and_organization_assignment
person_organization_id	(person. id) (organization. id)	41 41		cc_design_person_and_organization_assignment<= person_and_organization_assignment person_and_organization_assignment. assigned_person_and_organization->person_and_organization (person_and_organization. the_person->person person. id) (person_and_organization. the_organization->organization organization. id)
address	address	41		cc_design_person_and_organization_assignment<= person_and_organization_assignment person_and_organization_assignment. assigned_person_and_organization->person_and_organization (person_and_organization. the_person->person<- personal_address. people[i] personal_address<= address) (person_and_organization. the_organization->organization<- organizational_address. organizations[i] organizational_address<= address)

表 2(续)

应用元素	应用解释模型元素	源	规则	引 用 路 径
organization	organization	41		cc_design_person_and_organization_assignment<= person_and_organization_assignment person_and_organization_assignment. assigned_person_and_organization->>person_and_organization person_and_organization. the_organization->>organization
person	person	41		cc_design_person_and_organization_assignment<= person_and_organization_assignment person_and_organization_assignment. assigned_person_and_organization->>person_and_organization person_and_organization. the_person->>person
person_organization to part	PATH		42	cc_design_person_and_organization_assignment {cc_design_person_organization_assignment<= person_and_organization_assignment person_and_organization_assignment. role->>person_and_organization_role person_and_organization_role. name='design_owner'} cc_design_person_and_organization_assignment. items[i]->>person_organization_item=product product
person_organization to design_discipline_product_definition	PATH		41	cc_design_person_and_organization_assignment {cc_design_person_and_organization_assignment<= person_organization_assignment person_and_organization_assignment. role->>person_and_organization_role person_and_organization_role. name='creator'} cc_design_person_and_organization_assignment. items[i]->>person_organization_item=product_definition product_definition
person_organization to part_version	PATH		46	cc_design_person_and_organization_assignment {cc_design_person_and_organization_assignment<= person_organization_assignment person_and_organization_assignment. role->

表 2(完)

应用元素	应用解释模型元素	源	规则	引 用 路 径
person_organization to part_version	PATH		46	person_and_organization_role person_and_organization_role. name='creator'} cc_design_person_and_organization_assignment. items[i]->person_organization_item=product_definition_formation product_definition_formation
person_organization to supplier	PATH			cc_design_person_and_organization_assignment<= person_and_organization_assignment person_and_organization_assignment. assigned_person_and_organization->person_and_organization person_and_organization. the_organization->organization

表 3 映射表—物料清单功能单元

应用元素	应用解释模型元素	源	规则	引 用 路 径
ALTERNATE_PART	product	41		product {product<- alternate_product_relationship. alternate}
COMPONENT_ASSEMBLY_POSITION	([shape_representation_relationship] [representation_relationship_with_transformation]) (mapped_item)	43	16 73	
transformation	(transformation) ([mapped_item. mapping_target] [representation_map. mapping_origin])	43		(shape_representation_relationship<= representation_relationship=> representation_relationship_with_transformation representation_relationship_with_transformation. transformation_operator->transformation) ([mapped_item mapped_item. mapping_target] [mapped_item mapped_item. mapping_source->representation_map representation_map. mapping_origin])

表 3(续)

应用元素	应用解释模型元素	源	规则	引 用 路 径
ENGINEERING_ASSEMBLY	assembly_component_usage	44	16,70	
security_code	security_classification_level	44	28,37,56	assembly_component_usage classified_liem=assembly_component_usage classified_item<- cc_design_security_classification. items[i] cc_design_security_classification<= security_classification_assignment security_classification_assignment. assigned_security_classification->; security_classification security_classification. security_level->; security_classification_level
engineering_assembly to planned_effectivity	PATH			assembly_component_usage<= product_definition_usage<= product_definition_relationship<- product_definition_effectivity. usage
engineering_assembly to substitute_part	assembly_component_usage_substitute. substitute	44		assembly_component_usage<- assembly_component_usage_substitute. substitute assembly_component_usage_substitute
ENGINEERING_MAKE_FROM	design_make_from_relationship	203	35	design_make_from_relationship<= product_definition_relationship
ENGINEERING_NEXT_HIGHER_ASSEMBLY	next_assembly_usage_occurrence	44	70	
as_required	descriptive_measure	41	4	next_assembly_usage_occurrence<= assembly_component_usage=>; quantified_assembly_component_usage quantified_assembly_component_usage. quantity_>; measure_with_unit measure_with_unit. value_component_>; measure_value measure_value=descriptive_measure {descriptive_measure='as_required'}

表 3(续)

应用元素	应用解释模型元素	源	规则	引 用 路 径
component_quantity	measure_with_unit. value_component	41		next_assembly_usage_occurrence<= assembly_component_usage=> quantified_assembly_component_usage quantified_assembly_component_usage. quantity_> measure_with_unit measure_with_unit. value_component
reference_designator	assembly_component_usage. reference_designator	44		next_assembly_usage_occurrence<= assembly_component_usage assembly_component_usage. reference_designator
unit_of_measure	unit	41		next_assembly_usage_occurrence<= assembly_component_usage=> quantified_assembly_component_usage quantified_assembly_component_usage. quantity_> measure_with_unit measure_with_unit. unit_component_> unit
engineering_next_higher_assembly to	(context_dependent_shape representation)	41	16	(next_assembly_usage_occurrence<= assembly_component_usage<=
component_assembly_position	(mapped_item)	43	73	product_definition_usage<= product_definition_relationship characterized_product_definition=product_definition_relationship characterized_definition=characterized_product_definition characterized_definition<- property_definition. definition property_definition=> product_definition_shape<- context_dependent_shape_representation. represented_product_relation context_dependent_shape_representation) (next_assembly_usage_occurrence<= assembly_component_usage<= product_definition_usage<= product_definition_relationship

表 3(续)

应用元素	应用解释模型元素	源	规则	引 用 路 径
				product_definition_relationship. relating_product_definition->product_definition characterized_product_definition=product_definition characterized_definition=characterized_product_definition characterized_definition<- property_definition. definition property_definition{=>product_definition_shape}<- property_definition_representation. definition property_definition_representation{=>shape_definition_representation}<- property_definition_representation. used_representation->
engineering_next_higher_assembly to component_assembly_position(continued)	representation. items[i]	43		property_definition_representation. used_representation property_definition_representation{=>shape_definition_representation} property_definition_representation. definition->property_definition{=>product_definition_shape} property_definition. definition->characterized_definition=characterized_product_definition characterized_product_definition=product_definition product_definition<- product_definition_relationship. related_product_definition product_definition_relationship=> product_definition_usage=> assembly_component_usage=> next_assembly_usage_occurrence})
ENGINEERING_PROMISSORY_USAGE	promissory_usage_occurrence	44	70	
SUBSTITUTE_PART	product	41		product {product<- product_definition_formation. of_product product_definition_formation<- product_definition. formation product_definition<- product_definition_relationship. related_product_definition product_definition_relationship=>

表 3(完)

应用元素	应用解释模型元素	源	规则	引 用 路 径
SUBSTITUTE_PART	product	41		product_definition_usage=> assembly_component_usage<- assembly_component_uasge_substitute. substitute}

表 4 映射表—设计活动控制功能单元

应用元素	应用解释模型元素	源	规则	引 用 路 径
CHANGE_ORDER	change	203	10,11,74	change<= action_assignment {action_assignment. assigned_action->} action=> executed_action=> directed_action}
adopted_solution	action_method	41		change<= action_assignment action_assignment. assigned_action->} action {action=> executed_action=> directed_action} action. chosen_method->} action_method
change_date	date_and_time	41	52,22	change - date_time_item=change date_time_item<- cc_design_date_and_time_assignment. items[i] cc_design_date_and_time_assignment<= date_and_time_assignment {date_and_time_assignment. role->} date_time_role date_time_role. name='change_date'} date_and_time_assignment. assigned_date_and_time->} date_and_time

表 4(续)

应用元素	应用解释模型元素	源	规则	引 用 路 径
CHANGE_REQUEST	change_request	203	7,8,9,75	change_request<= action_request_assignment
recommended_solution	action_method	41	76	change_request<= action_request_assignment action_request_assignment. assigned_action_request->> versioned_action_request<- action_request_solution. request action_request_solution action_request_solution. method->> action_method
version	versioned_action_request. version	41		change_request<= action_request_assignment action_request_assignment. assigned_action_request->> versioned_action_request versioned_action_request. version
consequence	action_method. consequence	41		change_request<= action_request_assignment action_request_assignment. assigned_action_request->> versioned_action_request<- action_request_solution. request action_request_solution action_request_solution. method->> action_method action_method. consequence
START_ORDER	start_work	203	64,65	start_work<= action_assignment {action_assignment. assigned_action->> action=> action_execution=> directed_action}
START_REQUEST	start_request	203	61,62,63	start_request<= action_request_assignment

表 4(续)

应用元素	应用解释模型元素	源	规则	引 用 路 径
WORK_ORDER	directed_action	41	66,17	
work_order_id	action_directive. name	41		directed_action directed_action. directive->> action_directive action_directive. name
additional_data	action_directive. comment	41		directed_action directed_action. directive->> action_directive action_directive. comment
analysis_data	action_directive. analysis	41		directed_action directed_action. directive->> action_directive action_directive. analysis
work_order to approval	PATH		7,61	directed_action<= executed_action<= action<- action_assignment. assigned_action action_assignment=> (start_work approved_item=start_work) (change approved_item=change) approved_item<- cc_design_approval. items[i] cc_design_approval
work_order to part_version	PATH		66	directed_action<= executed_action<= action<- action_assignment. assigned_action action_assignment=> (start_work start_work. items[i]->> work_item work_item=product_definition_formation)

表 4(续)

应用元素	应用解释模型元素	源	规则	引 用 路 径
work_order to part_version	PATH		66	(change change.items[i]->) work_item work_item=product_definition_formation) product_definition_formation=> product_definition_formation_with_specified_source
work_order to work_request	action_directive.requests[i]	41		directed_action directed_action.directive->) action_directive action_directive.requests[i]->) versioned_action_request
WORK_REQUEST	versioned_action_request	41	74	
description	versioned_action_request.description	41		
reason	versioned_action_request.purpose	41		
request_date	date_and_time	41	52,22	versioned_action_request<- action_request_assignment.assigned_action_request action_request_assignment=> (start_request date_time_item=start_request) (change_request date_time_item=change_request) date_time_item<- cc_design_date_and_time_assignment.items[i] cc_design_date_and_time_assignment<= date_and_time_assignment {date_and_time_assignment.role->) date_time_role.name='request_date'} date_and_time_assignment.assigned_date_and_time->) date_and_time
status	action_request_status	41	48	versioned_action_request<- action_request_status.assigned_request action_request_status

表 4(完)

应用元素	应用解释模型元素	源	规则	引 用 路 径
work_request_id	versioned_action_request.id	41		
work_request to approval	PATH		7,61	versioned_action_request<- action_request_assignment.assigned_action_request action_request_assignment=>(start_request approved_item=start_request)(change_request approved_item=change_request) approved_item<- cc_design_approval.items[i] cc_design_approval
work_request to part_version	PATH		74	versioned_action_request<- action_request_assignment.assigned_action_request action_request_assignment=>(start_request start_request.items[i]->start_request_item start_request_item=product_definition_formation) (change_request change_request.items[i]->change_request_item change_request_item=product_definition_formation) product_definition_formation=> product_definition_formation_with_specified_source
work_request to person_organization	PATH		9,63	versioned_action_request<- action_request_assignment.assigned_action_request action_request_assignment=>(start_request person_organization_item=start_request) (change_request person_organization_item=change_request) person_organization_item<- cc_design_person_and_organization_assignment.items[i] cc_design_person_and_organization_assignment

表 5　映射表—设计信息功能单元

应用元素	应用解释模型元素	源	规　则	引　用　路　径
ADDITIONAL_DESIGN_INFORMATION	document	41	31	document {document<- document_relationship. relating_document}
additional_design_information to specification	PATH		31	document<- document_relationship. relating_document document_relationship document_relationship. related_document->document<- document_reference. assigned_document document_reference=> cc_design_specification_reference
DESIGN_ SPECIFICATION	document	41	53,23	document {document. kind->document_type document_type. product_data_type='design_specification'}
MATERIAL_ SPECIFICATION	document	41	53,23	document {document. kind->document_type document_type. product_data_type='material_specification'}
PROCESS_ SPECIFICATION	document	41	53,23	document {document. kind->document_type document_type. product_data_type='process_specification'}
SPECIFICATION	cc_design_specification_reference	203		cc_design-specification_reference<= document_reference
specification_code	document. id	41		cc_design_specification_reference<= document_reference document_reference. assigned_document->document document. id
specification_source	document_reference. source	41		cc_design_specification_reference<= document_reference document_reference. source

表 5(完)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
specification to usage_constraint	PATH			cc_design_specification_reference<= document_reference document_reference. assigned_document->>document<- document_usage_constraint. source document_usage_constraint
SURFACE_FINISH_SPECIFICATION	document	41	53,23	document {document. kind->>document_type document_type. product_data_type='surface_finish_specification'}
USAGE_CONSTRAINT	document_usage_constraint	41		
element	document_usage_constraint. subject_element	41		
value	document_usage_constraint. subject_element_value	41		

表 6 映射表—有效性功能单元

应用元素	应用解释模型元素	源	规 则	引 用 路 径
PLANNED_DATE_EFFECTIVITY	dated_effectivity	41	67	
end_date	date_and_time	41	52,22	dated_effectivity dated_effectivity. effectivity_end_date->>date_and_time
start_date	date_and_time	41	52,22	dated_effectivity dated_effectivity. effectivity_start_date->>date_and_time
PLANNED_EFFECTIVITY	[product_definition_effectivity] [configuration_effectivity]	41 44	67,32	
planned_effectivity to approval	PATH		32	[product_definition_effectivity] [configuration_effectivity]<-

表 6(续)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
planned_effectivity to approval	PATH		32	approved_item=configuration_effectivity approved_item<- cc_design_approval. items[i] cc_design_approval
PLANNED_LOT_EFFECTIVITY	lot_effectivity	41	67	
lot_number	lot_effectivity. effectivity_lot_id	41		
lot_size	measure_value	41		lot_effectivity lot_effectivity. effectivity_lot_size->> measure_with_unit measure_with_unit. value_component->> measure_value
lot_size_unit_of_measure	unit	41	24	lot_effectivity. effectivity_lot_size->> measure_with_unit measure_with_unit. unit_component->> unit
PLANNED_SEQUENCE_EFFECTIVITY	serial_numbered_effectivity	41	67	
component_quantity	measure_value	41		serial_numbered_effectivity<<= effectivity=> product_definition_effectivity product_definition_effectivity. usage->> product_definition_relationship=> product_definition_usage=> assembly_component_usage=> quantified_assembly_component_usage quantified_assembly_component_usage. quantity->> measure_with_unit measure_with_unit. value_component->> measure_value

表 6(完)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
from_effectivity_id	serial_numbered_effectivity. effectivity_start_id	41		
quantity_unit_of_measure	unit	41	24	serial_numbered_effectivity<= effectivity=> product_definition_effectivity product_definition_effectivity. usage->> product_definition_relationship=> product_definition_usage=> assembly_component_usage=> quantified_assembly_component_usage quantified_assembly_component_usage. quantity->> measure_with_unit measure_with_unit. unit_component->> unit
thru_effectivity_id	serial_numbered_effectivity. effectivity_end_id	41		

表 7 映射表—截止项标识功能单元

应用元素	应用解释模型元素	源	规 则	引 用 路 径
PRODUCT_CONFIGURATION	configuration_item	44	12,13	
item_id	configuration_item identification	44		
phase_of_product	configuration_item. purpose	41		
product_configuration to approval	PATH		12	configuration_item approved_item=configuration_item approved_item<- cc_design_approval. items[i] cc_design_approval
product_configuration to part	configuration_design. design	44		configuration_item<- configuration_design. configuration configuration_design

表 7(完)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
product_configuration to part	configuration_design. design	44		configuration_design. design->product_definition_formation{=>product_definition_formation_with_specified_source}product_definition_formation. of_product->product
product_configuration to planned_effectivity	PATH			configuration_item<-configuration_design. configuration configuration_design<-configuration_effectivity. configuration [configuration_effectivity] [product_definition_effectivity]
PRODUCT_MODEL	product_concept	44	38	
model_name	product_concept. identification	44		
product_model to product_configuration	configuration_item. item_concept	44		product_concept<-configuration_item. item-concept configuration_item

表 8 映射表—面边界表示功能单元

应用元素	应用解释模型元素	源	规 则	引 用 路 径
FACETTED_B_REP	faceted_brep_shape representation	203	73	faceted_brep_shape_representation<=shape_representation

表 9 映射表—拓扑流形曲面功能单元

应用元素	应用解释模型元素	源	规 则	引 用 路 径
MANIFOLD_SURFACE_WITH_TOPOLOGY	manifold_surface_shape_representation	203	73	manifold_surface_shape_representation<=shape_representation

表 10 映射表—无拓扑曲面和线框功能单元

应用元素	应用解释模型元素	源	规 则	引 用 路 径
NON_TOPOLOGICAL_SURFACE_AND_WIREFRAME	(geometrically_bounded_wireframe_shape_representation)	203	73	(geometrically-bounded_wireframe_shape_representation<=shape_representation)

表 10(完)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
NON_TOPOLOGICAL_SURFACE_AND_WIREFRAME	(geometrically_bounded_surface_shape_representation)		73	(geometrically_bounded_surface_shape_representation<= shape_representation)

表 11 映射表—零件标识功能单元

应用元素	应用解释模型元素	源	规 则	引 用 路 径
DESIGN_DISCIPLINE_PRODUCT_DEFINITION	product_definition	41	30,31, 39,40, 41	
cad_filename	document.id	41	53,23	product_definition=> product_definition_with_associated_documents product_definition_with_associated_documents.documentation_ids-> document {document.kind-> document_type document_type.product_data_type='cad_filename'} document.id
creation_date	date_and_time	41	52,22	product_definition date_time_item=product_definition date_time_item<- cc_design_date_and_time_assignment.items[i] cc_design_date_and_time_assignment<= date_and_time_assignment {date_and_time_assignment.role-> date_time_role date_time_role.name='creation_date'} date_and_time_assignment.assigned_date_and_time-> date_and_time
description	product_definition.description	41		
discipline_id	[product_definition_context.life_cycle_stage] [application_context_element.name]	41 41	30	product_definition product_definition.frame_of_reference-> product_definition_context [product_definition_context.life_cycle_stage]

表 11(续)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
discipline_id	[product_definition_context. life_cycle_stage] [application_context_element. name]	41 41	30	[product_definition_context<= application_context_element application_context_element. name]
design_discipline_product_definition to additional_design_information	PATH		31	product_definition specified_item=product_definition specified_item<- cc_design_specification_reference. items[i] cc_design_specification_reference<= document_reference document_reference. assigned_document->document<- document_relationship. relating_document
design_discipline_product_definition to approval	PATH		39	product_definition approved_item=product_definition approved_item<- cc_design_approval. items[i] cc_design_approval
design_discipline_product_definition to engineering_assembly (is the assembly for)	product_definition_relationship. relating_product_definition	41		product_definition<- product_definition_relationship. relating_product_definition product_definition_relationship=> product_definition_usage=> assembly_component_usage
design_discipline_product_definition to engineering_assembly(is used sa a component in)	product_definition_relationship. related_product_definition	41		product_definition<- produst_definition_relationship. related_product_definition product_definition_relationship=> product_definition_usage=> assembly_component_usage
design_discipline_product_definition to engineering_make_from(is the base design in)	PATH			product_definition<- product_definition_relationship. relating_product_definition product_definition_relationship=> design_make_from_relationship

表 11(续)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
design_ discipline_ product_definition to engineering_make_from(is the resultant design in)	PATH			product_definition<- product_definition_relationship. related_product_definition product_definition_relationship=> design_make_from_relationship
PART	product	41	42,43, 44,68	
part_classification	product_related_product_category	41	55	product<- product_related_product_category. products[i] product_related_product_category
part_nomenclature	product. name	41		
part_number	product. id	41		
part_type	product_related_product_category	41	43,55	product<- product_related_product_category. products[i] product_related_product_category
standard_part_indicator	product_category. name	41		product<- product_related_product_category. products[i] product_related_product_category<= product_category<- (product_category_relationship. related_product_category->) (product_category_relationship. relating_product_category->) product_category product_category. name {product_category. name='standard_part'}
part to alternate_part	PATH			product<- alternate_product_relationship. base alternate_product_relationship alternate_product_relationship. alternate->product

表 11(续)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
part to part-version	product_definition_formation. of_product	41	44	product<- product_definition_formation. of_product product_definition_formation=> product_definition_formation_with_specified_source
part to substitute_part	PATH			product<- product_definition_formation. of_product product_definition_formation{=>product_definition_formation_with_specified_source}<- product_definition. formation product_definition<- product_definition_relationship. relating_product_definition product_definition_relationship=> product_definition_usage=> assembly_component_usage<- assembly_component_usage_substitute. substitute assembly_component_usage_substitute
PART_VERSION	product_definition_formation_with_specified_source	41	44,45,46,47,69,74	
contract_number	contract. name	41	14,15,20,51	product_definition_formation_with_specified_source<= product_definition_formation contracted_item=product_definition_formation contracted_item<- cc_design_contract. items[i] cc_design_contract<= contract_assignment contract_assignment. assigned_contract->contract contract. name
make_or_buy_code	product_definition_formation_with_specified_source. make_or_buy	41		

表 11(续)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
release_status	approval_status	41	49,18	product_definition_formation_with_specified_source<= product_definition_formation approved_item=product_definition_formation approved_item<- cc_design_approval.items[i] cc_design_approval<= approval_assignment approval_assignment.assigned_approval->>approval approval.status->>approval_status
revision_letter	product_definition_formation.id	41		product_definition_formation_with_specified_source<= product_definition_formation product_definition_formation.id
security_code	security_classification_level	41	28,56,57,59,58,60	product_definition_formation_with_specified_source<= product_definition_formation classified_item=product_definition_formation classified_item<- cc_design_security_classification.items[i] cc_design_security_classification<= security_classification_assignment security_classification_assignment.assigned_security_classification->>security_classification security_classification.security_level->>security_classification_level
part_version to approval	PATH			product_definition-formation_with_specified_source<= product_definition_formation approved_item=product_definition_formation approved_item<- cc_design_approval.items[i] cc_design_approval
part_version to design_discipline_product_definition	product_definition.formation	41		product_definition_formation_with_specified_source<= product_definition_formation<- product_definition.formation product_definition

表 11(完)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
part_version to supplied_part_version	PATH		36	product_definition_formation_with_specified_source<= product_definition_formation<- product_definition.formation product_definition<- product_definition_relationship.relating_product_definition product_definition_relationship=> supplied_part_relationship

表 12 映射表—形状功能单元

应用元素	应用解释模型元素	源	规 则	引 用 路 径
GEOMETRIC_MODEL_REPRESENTATION	shape_representation	41	33,34,25,27,29,71,72,73	
geometric_model_representation to component_assembly_position (represents components in)	(shape_representation_relationship.rep_2) (representation_map.mapped_representation)	41 43	16	(shape_representation<= representation<- shape_representation_relationship.rep_2) (shape_representation<= representation<- representation-map.mapped_representation representation-map<- mapped_item.mapping_source mapped_item<- representation.items[i])
geometric_model_representation to component_assembly_position (represents assembly in)	(context_dependent_shape_representation) (mapped_item)	41 43	16	(shape_representation<= representation<- shape_representation_relationship.rep_1) (shape_representation<= representation representation.items[i]{->mapped-item})

表 12(续)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
SHAPE	shape_definition_representation	41		
shape to design_discipline_product_definition	PATH	41		shape_definition_representation<= property_definition_representation property_definition_representation.definition->property_definition{=>product_definition_shape} property_definition.definition->characterized_definition characterized_definition=characterized_product_definition characterized_product_definition=product_definition product_definition
shape to geometric_model_representation	PATH	41		shape_definition_representation<= property_definition_representation property_definition_representation.used_representation->representation=> shape_representation
shape to shape_aspect	PATH	41		shape_definition_representation<= property_definition_representation property_definition_representation.definition->property_definition property_definition.definition->characterized_definition=shape_definition shape_definition=shape_aspect shape_aspect
SHAPE_ASPECT	shape_aspect	41		
shape_aspect to geometric_model_representation	PATH	41	73	shape_aspect shape_definition=shape_aspect characterized_definition=shape_definition characterized_definition<- property_definition.definition property_definition<- property_definition_representation.definition property_definition_representation{=>shape_definition_representation} property_definition_representation.used_representation->representation=> shape_representation

表 12(完)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
shape_aspect to specification	specified_item =shape_aspect	203	31	shape_aspect specified_item=shape_aspect specified_item<- cc_design_specification_reference. items[i] cc_design_specification_reference

表 13 映射表—源控制功能单元

应用元素	应用解释模型元素	源	规 则	引 用 路 径
SUPPLIED_PART_VERSION	product_definition_formation_with_specified_source	41	69	
certification_required	cc_design_certification	203	5,6, 19,50	product_definition_formation_with_specified_source<= product_definition_formation<- product_definition. formation product_definition<- product_definition_relationship. related_product_definition product_definition_relationship=>~ supplied_part_relationship certified_item=supplied_part_relationship certified_item<- cc_design_certification. items[i] cc_design_certification
supplier_part_number	product. id	41		product_definition_formation_with_specified_source<= product_definition_formation<- product_definition. formation product_definition<- product_definition_relationship. relating_product_definition product_definition_relationship=> supplied_part_relationship<= product_definition_relationship product_definition_relationship. related_product_definition->

表 13(完)

应用元素	应用解释模型元素	源	规 则	引 用 路 径
supplier_part_number	product.id	41		product_definition product_definition.formation->product_definition_formation{=>product_definition_formation_with_specified_source} product_definition_formation.of_product->product product.id
supplied_part_version to approval	PATH		45	product_definition_formation {product_definition_formation<-product_definition.formation product_definition<-product_definition_relationship.related_product_definition product_definition_relationship=>supplied_part_relationship} approved_item=product_definition_formation approved_item<-cc_design_approval.items[i] cc_design_approval
SUPPLIER	organization	41		
supplier_id	organization.id	41		
supplier to supplied_part_version	PATH			organization<-person_and_organization.the_organization person_and_organization<-person_and_organization_assignment.assigned_person_and_organization person_and_organization_assignment=>cc_design_person_and_organization_assignment cc_design_person_and_organization.items[i]->person_organization_item person_organization_item=product_definition_formation product_definition_formation

表 14　映射表—拓扑线框功能单元

应用元素	应用解释模型元素	源	规　则	引　用　路　径
WIREFRAME_WITH_TOPOLOGY	(edge_based_wireframe_shape_representation)	203	73	(edge_based_wireframe_shape_representation<=shape_representation)
	(shell_based_wireframe_shape_representation)		73	(shell_based_wireframe_shape_representation<=shape_representation)

前表所引用的规则：

1) approval_requires_approval_date_time（批准需要批准日期时间）
2) approval_requires_approval_person_organization（批准需要批准人员组织）
3) approvals_are_assigned（批准被赋值）
4) as_required_quantity（按所需数量）
5) certification_requires_approval（认证需要批准）
6) certification_requires_date_time（认证需要日期时间）
7) change_request_requires_approval（更改请求需要批准）
8) change_request_requires_date_time（更改请求需要日期时间）
9) change_request_requires_person_organization（更改请求需要人员组织）
10) change_requires_approval（更改需要批准）
11) change_requires_date_time（更改需要日期时间）
12) configuration_item_requires_approval（配置项需要批准）
13) configuration_item_requires_person_organization（配置项需要人员组织）
14) contract_requires_approval（合同需要批准）
15) contract_requires_person_organization（合同需要人员组织）
16) coordinated_assembly_and_shape（协调的装配和形装）
17) dependent_instantiable_action_directive（依赖可实例化活动命令）
18) dependent_instantiable_approval_status（依赖可实例化批准状态）
19) dependent_instantiable_certification_type（依赖可实例化认证类型）
20) dependent_instantiable_contract_type（依赖可实例化合同类型）
21) dependent_instantiable_date（依赖可实例化日期）
22) dependent_instantiable_date_time_role（依赖可实例化日期时间职能）
23) dependent_instantiable_document_type（依赖可实例化文档类型）
24) dependent_instantiable_named_unit（依赖可实例化命名单位）
25) dependent_instantiable_parametric_representation_context（依赖可实例化参数表达相关环境）
26) dependent_instantiable_person_and_organization_role（依赖可实例化人员和组织职能）
27) dependent_instantiable_representation_item（依赖可实例化表示项）
28) dependent_instantiable_security_classification_level（依赖可实例化安全分类等级）
29) dependent_instantiable_shape_representation（依赖可实例化形状表示）
30) design_context_for_property（设计相关环境目的）
31) document_to_product_definition（文档与产品定义）
32) effectivity_requires_approval（有效性需要批准）
33) geometric_representation_item_3d（三维几何表示项）
34) global_unit_assignment（全局单位赋值）
35) no_shape_for_make_from（无参照形状）
36) no_shape_for_supplied_part（无供应零件形状）
37) acu_requires_security_classification（装配构件使用需要安全分类）
38) product_concept_requires_configuration_item（产品概念需要配置项）
39) product_definition_requires_approval（产品定义需要批准）
40) product_definition_requires_date_time（产品定义需要日期时间）
41) product_definition_requires_person_organization（产品定义需要人员组织）
42) product_requires_person_organization（产品需要人员组织）

43) product_requires_product_category (产品需要产品目录)
44) product_requires_version (产品需要版本)
45) product_version_requires_approval (产品版本需要批准)
46) product_version_requires_person_organization (产品版本需要人员组织)
47) product_version_requires_security_classification (产品版本需要安全分类)
48) restrict_action_request_status (限制活动请求状态)
49) restrict_approval_status (限制批准状态)
50) restrict_certification_type (限制认证类型)
51) restrict_contract_type (限制合同类型)
52) restrict_date_time_role (限制日期时间职能)
53) restrict_document_type (限制文档类型)
54) restrict_person_organization_role (限制人员组织职能)
55) restrict_product_category_value (限制产品目录值)
56) restrict_security_classification_level (限制安全分类等级)
57) security_classification_optional_date_time (安全分类可选的日期时间)
58) security_classification_requires_approval (安全分类需要批准)
59) security_classification_requires_date_time (安全分类需要日期时间)
60) security_classification_requires_person_organization (安全分类需要人员组织)
61) start_request_requires_approval (开始请求需要批准)
62) start_request_requires_date_time (开始请求需要日期时间)
63) start_request_requires_person_organization (开始请求需要人员组织)
64) start_work_requires_approval (开始工作需要批准)
65) start_work_requires_date_time (开始工作需要日期时间)
66) subtype_mandatory_action (子类型强制的活动)
67) subtype_mandatory_effectivity (子类型强制的有效性)
68) subtype_mandatory_product_context (子类型强制的产品相关环境)
69) subtype_mandatory_product_definition_formation (子类型强制的产品定义功能)
70) subtype_mandatory_product_definition_usage (子类型强制的产品定义使用)
71) subtype_mandatory_representation (子类型强制的表示)
72) subtype_mandatory_representation_context (子类型强制的表示相关环境)
73) subtype_mandatory_shape_representation (子类型强制的形状表示)
74) unique_version_change_order_rule (单个版本更改指令规则)
75) versioned_action_request_requires_solution (该版活动请求需要方案)
76) versioned_action_request_requires_status (该版活动请求需要状态)

5.2 AIM EXPRESS 简表

本条描述了使用集成资源元素的 EXPRESS 模式，它包括实体、实体说明、规则以及本标准的特定函数。本条也描述了对从集成资源输入的结构正文部分的修改。AIM 的集成资源结构所提供的定义和 EXPRESS 包括选择列表项和子类，它们不输入到 AIM 中。引用这些项和子类的集成资源所指出的要求只能应用于那些输入到 AIM 中的项。

EXPRESS 描述

```
*)
SCHEMA config_control_design;
```

```
USE FROM application_context_schema   --GB/T 16656.41
    (application_context,
    application_protocol_definition,
    product_context,
    product_definition_context,
    product_concept_context);

USE FROM product_definition_schema   --GB/T 16656.41
    (product,
    product_definition,
    product_definition_formation,
    product_definition_formation_with_specified_source,
    product_definition_relationship,
    product_category,
    product_category_relationship,
    product_related_product_category,
    product_definition_with_associated_documents);

USE FROM product_structure_schema   --GB/T 16656.44
    (product_definition_usage,
    assembly_component_usage,
    next_assembly_usage_occurrence,
    promissory_usage_occurrence,
    quantified_assembly_component_usage,
    specified_higher_usage_occurrence,
    assembly_component_usage_substitute,
    alternate_product_relationship);

USE FROM configuration_management_schema   --GB/T 16656.44
    (configuration_item,
    configuration_design,
    configuration_effectivity);

USE FROM product_concept_schema   --GB/T 16656.44
    (product_concept);

USE FROM product_property_definition_schema   --GB/T 16656.41
    (product_definition_shape,
    shape_aspect,
    shape_aspect_relationship);

USE FROM product_property_representation_schema   --GB/T 16656.41
```

```
    (context_dependent_shape_representation,
    shape_representation,
    shape_representation_relationship,
    shape_definition_representation);

USE FROM representation_schema   --GB/T 16656.43
    (functionally_defined_transformation,
    item_defined_transformation,
    global_uncertainty_assigned_context,
    mapped_item,
    representation,
    representation_context,
    parametric_representation_context,
    representation_item,
    representation_map,
    representation_relationship_with_transformation);

USE FROM geometry_schema   --GB/T 16656.42
    (axis1_placement,
    axis2_placement_2d,
    axis2_placement_3d,
    b-spline_curve_with_knots,
    b-spline_surface_with_knots,
    bezier_curve,
    bezier_surface,
    boundary_curve,
    cartesian_point,
    cartesian_transformation_operator_3d,
    cartesian_transformation_operator_2d,
    circle,
    composite_curve,
    composite_curve_on_surface,
    composite_curve_segment,
    conical_surface,
    curve,
    curve_bounded_surface,
    curve_replica,
    cylindrical_surface,
    degenerate_pcurve,
    dimension_count,
    direction,
    ellipse,
```

evaluated_degenerate_pcurve,
geometric_representation_context,
geometric_representation_item,
hyperbola,
intersection_curve,
line,
offset_curve_2d,
offset_curve_3d,
offset_surface,
outer_boundary_curve,
parabola,
pcurve,
plane,
point,
point_on_curve,
point_on_surface,
polyline,
quasi_uniform_curve,
quasi_uniform_surface,
rational_b_spline_curve,
rational_b_spline_surface,
rectangular_composite_surface,
rectangular_trimmed_surface,
reparametrised_composite_curve_segment,
seam_curve,
spherical_surface,
surface,
surface_curve,
surface_of_linear_extrusion,
surface_of_revolution,
surface_replica,
toroidal_surface,
trimmed_curve,
uniform_curve,
uniform_surface,
vector);

USE FROM topology_schema --GB/T 16656.42
(clsoed_shell,
connected_edge_set,
connected_face_set,
edge_curve,

```
    edge_loop,
    face_bound,
    face_outer_bound,
    face_surface,
    open_shell,
    oriented_face,
    poly_loop,
    vertex_loop,
    vertex_point,
    vertex_shell,
    wire_shell);

USE FROM geometric_model_schema  --GB/T 16656.42
    (edge_based_wireframe_model,
    facetted_brep,
    geometric_curve_set,
    geometric_set,
    manifold_solid_brep,
    shell_based_surface_model,
    shell_based_wireframe_model);

USE FROM action_schema  --GB/T 16656.41
    (action,
    action_method,
    action_request_solution,
    action_request_status,
    action_status,
    action_directive,
    directed_action,
    versioned_action_requset);

USE FROM certification_schema  --GB/T 16656.41
    (certification,
    certification_type);

USE FROM approval_schema  --GB/T 16656.41
    (approval_date_time,
    approval_person_organization,
    approval,
    approval_status,
    approval_relationship);
```

```
USE FROM contract_schema  --GB/T 16656.41
   (contract,
   contract_type);

USE FROM security_classification_schema  --GB/T 16656.41
   (security_classification,
   security_classification_level);

USE FROM person_organization_schema  --GB/T 16656.41
   (person_and_organization,
   organization_relationship,
   personal_address,
   organizational_address,
   organization_project,
   person_and_organization_role;

USE FROM date_time_schema  --GB/T 16656.41
   (date_and_time,
   date,
   calendar_date,
   ordinal_date,
   week_of_year_and_day_date,
   date_time_role);

USE FROM document_schema  --GB/T 16656.41
   (document_with_class,
   document_usage_constraint,
   document_type,
   document_relationship);

USE FROM effectivity_schema  --GB/T 16656.41
   (effectivity,
   serial_numbered_effectivity,
   dated_effectivity,
   lot_effectivity);

USE FROM management_resources_schema  --GB/T 16656.41
   (approval_assignment,
   certification_assignment,
   contract_assignment,
   date_and_time_assignment,
   person_and_organization_assignment,
```

```
    document_reference,
    security_classification_assignment,
    action_assignment,
    action_request_assignment);

USE FROM measures_schema  --GB/T 16656.41
    (area_measure,
    name_unit,
    context_dependent_measure,
    context_dependent_unit,
    conversion_based_unit,
    count_measure,
    descriptive_measure,
    si_unit,
    area_unit,
    length_unit,
    mass_measure,
    mass_unit,
    plane_angle_unit,
    solid_angle_unit,
    volume_unit,
    measure_with_unit,
    area_measure_with_unit,
    length_measure_with_unit,
    mass_measure_with_unit,
    plane_angle_measure_with_unit,
    positive_plane_angle_measure,
    solid_angle_measure,
    solid_angle_measure_with_unit,
    volume_measure,
    volume_measure_with_unit,
    global_unit_assigned_context);
(*
  说明---- 上面引用的模式可以在GB/T 16656下面的部分找到:
application_context_schema(应用相关环境模式)          GB/T 16656.41
product_definition_schema(产品定义模式)               GB/T 16656.41
product_structure_schema(产品结构模式)                GB/T 16656.44
configuration_management_schema(配置管理模式)         GB/T 16656.44
product_concept_schema(产品概念模式)                  GB/T 16656.44
product_property_definition_schema(产品属性定义模式)  GB/T 16656.41
product_property_representation_schema(产品属性表示模式) GB/T 16656.41
representation_schema(表示模式)                       GB/T 16656.43
```

geometry_schema(几何模式)	GB/T 16656.42
topology_schema(拓扑模式)	GB/T 16656.42
geometric_model_schema(几何模型模式)	GB/T 16656.42
action_schema(活动模式)	GB/T 16656.41
certification_schema(认证模式)	GB/T 16656.41
approval_schema(批准模式)	GB/T 16656.41
contract_schema(合同模式)	GB/T 16656.41
security_classification_schema(安全分类模式)	GB/T 16656.41
person_organization_schema(人员组织模式)	GB/T 16656.41
date_time_schema(日期时间模式)	GB/T 16656.41
document_schema(文档模式)	GB/T 16656.41
effectivity_schema(有效性模式)	GB/T 16656.41
management_resources_schema(管理资源模式)	GB/T 16656.41
measures_schema(度量模式)	GB/T 16656.41

5.2.1 基本概念和假设

本标准用于三维产品设计数据的配置控制。该模式的基本概念是由组织来控制这些产品设计类型的配置。该模式不适合从绘图开始的产品设计。它是为控制三维设计的配置而制定的。

5.2.1.1 产品形状与其配置数据的关系

在本标准中产品形状是由 shape_representation 实体表示的。该实体和它的子类定义了构成特定表示类型的几何和拓扑实体。每个机械零件或装配件将通过 product 实体的实例给出。每个产品至少有一个版本是通过 product_definition_formation 实体的实例给出的。通过 product_definition 实体可以给每个版本一个或多个定义。每个定义可以有它自己的形状表示。这可以通过使用一些 AIM 实体来实现，这些 AIM 实体与 product_definition 实例有关，而该实例又与合适的 shape_representation 实例有关。product_definition 的形状概念通过 product_definition_shape 的一个实例给出。产品的这种形状再通过 shape_definition_representation 的一个实例连到 shape_representation 实体上。shape_definition_representation 从它的超类 property_definition_representation 继承属性，该超类引用 shape_representation 实体，包括零件形状的几何和拓扑，以及一个 characterized_definition 选择类型。该类型可以表示为 shape_aspect，shape_aspect_relationship 或 characterized_product_definition，后者也是一个选择类型。为了指出感兴趣的性能是 product_definition 的形状，应当使用 property_definition 的子类 product_definition_shape。限定 product_definition_shape 实体使用 characterized_product_definition 选择类型。characterized_product_definition 选择类型可以通过其选择列表的引用来表示 product_definition 或 product_definition_relationship。这种引用将保证用 product_definition_shape 实体定义一个 product_definition 的形状。如果感兴趣的性能是 product_definition 的形状的一个特性，或 product_definition 的形状的两个特性之间的关系，那么将使用 property_definition，并且它的 definition 的属性将引用 shape_aspect 或 shape_aspect_relationship 的一个实例。为了指出一个产品的形状，在这里将引用 product_definition 实体。约束条件 subtype_mandatory_shape_representation(见5.2.4.73)规定用 shape_representation 的一个子类型规定零件的形状。可以通过 representation_items 的任何集合给出 shape_aspect 或 shape_aspect_relationship 的表示。

5.2.1.2 构件的形状与其装配件形状之间的关系

可以用两种方法建立起组成装配件的构件形状与其装配件形状间的关系。第一种方法，通过变换定义构件相对于装配件的方向，包括定义每种零件(构件和装配件)的形状，然后建立两种形状之间的关系并提供信息。第二种方法是定义每种零件(构件和装配件)，然后把构件的形状直接结合到装配件的形状

中。在不同表示类型的形状之间建立关系将使用第一种方法。如果它们的类型相同，可以用第二种方法把一个构件的表示结合到其装配件的表示中。

两种方法都采用 shape_representation 和 product_definition 实体。第一种方法也采用 product_definition，shape_representation_relationship AND(和)representation_relationship_with_transformation(每个实体的一个实例通过 AND(和)关系形成两个实体的一个复杂实例)和 context_dependent_shape_representation 实体。第二种方法采用 mapped_item 和 representation_map 实体。

当使用第一种方法在构件形状与装配件形状之间建立联系时，定义构件和装配件 product definition 形状的每个 shape_representation 实体通过在 shape_representation_relationship 实体中的引用建立联系。在这种情况下，将通过 shape_representation_relationship AND(和)represenation_relationship_with_transformation 实体的一个复杂实例的格式提供方向信息。represenation_relationship_with_transformation 实体引用一个 transformation，后者是一个选择类型，它定义方向，对 item_defined_transformation 用 axis2_placement_3d 实体表示，对 functionally_defined_transformation 用 cartesian_transformation_operator 实体表示。另外，在 shape_representation_relationship 与 product_definition_relationship 之间建立联系必须显式地给出 context_dependent_shape_representation 的一个实例。shape_representation_relationship 定义了两个形状之间的关系，product_definition_relationship 定义了两个 product_definition 之间的装配件-构件关系。

当使用第二种方法在构件形状与装配件形状之间建立联系时，representation_map 实体的一个实例引用了定义构件形状几何和拓扑的 shape_representation 实体。mapped_item 实体的一个实例的 mapping_source 属性引用了 representation_map 实体的实例。representation_map 实体的 mapped_representation 属性将引用 shape_representation 的子类型，该子类型定义了形状的几何和拓扑表示。这样 mapped_item 实体的实例将被加到 shape_representation 实例的 items 集合中，该 shape_representation 实体定义装配件的几何和拓扑。

5.2.1.3 形状表示的类型

本标准定义了八类零件形状的表示：基于边和壳模型的线框表示、有几何有界的线框表示、流形曲面模型、有几何有界的流形曲面、面边界表示实体模型及边界表示实体模型。这些表示类型中的每一个都是自包含的，即一个类型不能包含另一个类型。每个类型都通过 shape_representation 实体的子类型给出。每个子类型包括局部规则，它用于控制其中几何和拓扑实体的类型。每个 shape_representation 实体必须是这些子类型之一，除非用它作上面第一种方法中的装配件形状表示。在那种情况下，为了定义其中的构件表示方向，shape_representation 实例在它的 items 集合中将只包含 axis2_placement_3d 实体。由于每个子类型的规则互相冲突，任何被 representation_map 引用以实现上面第二种方法的 shape_representation 类型将和一个 shape_representation 类型有相同的类型，该 shape_representation 有一个 mapped_item，它引用在它的 items 集合中的 representation_map。

5.2.1.4 全局规则的使用

在 GB/T 16656集成资源部分，不同实体之间的许多关系都是通过两个相关实体之间最通用的0个或多个基来规定的。这个基意味着关系是可选的，或可能有一个或几个被关联实体与关联实体的一个实例有关。本标准使用全局规则约束这个基。在某些情况下，该约束是一对一的，在某些情况下，至少有一个。这些规则的例子包括 contract_requires_person_organization，approval_requires_approval_date_time 和 certification_requires_approval，它们是一一对应的，而 change_request_requires_person_organization 和 product_requires_version 对应的是一个或多个。由全局规则限制 STRING 类型的属性值只能用于机械零件和装配件的配置控制三维设计，这些规则的例子是 restrict_approval_status 和 restrict_person_organization_role.

5.2.1.5 单位赋值

基于全局的形状表示赋值单位。这通过创建一个 global_unit_assigned_context 实例来实现。该实体

包含一个属性，它允许将单位的一集合赋值给一个 representation_context。每个 shape_representation 实体有它的表示范围。如果度量单位是 shape_representation 的一个实例所必需的，那么该实例在它的 context_of_items 属性中将有一个 global_unit_assigned_context 实体。

5.2.2 配置控制设计类型

5.2.2.1 工作项

work_item(工作项)标识 product_definition_formation，它是初始设计活动或对设计进行修改的结果。

EXPRESS 描述：

```
*)
TYPE work_item=SELECT (product_definition_formation);
END_TYPE;
(*
```

5.2.2.2 更改请求项

change_request_item(更改请求项)是零件的 product_definition_formation，它受 change_request 的影响。

EXPRESS 描述：

```
*)
TYPE change_request_item=SELECT (product_definition_formation);
END_TYPE;
(*
```

5.2.2.3 开始请求项

start_request_item(开始请求项)是零件的 product_definition_formation，它是 start_request 创建的。

EXPRESS 描述：

```
*)
TYPE start_request_item=SELECT (product_definition_formation);
END_TYPE;
(*
```

5.2.2.4 认证项

certified_item(认证项)将 certification 应用到一个零件，该零件由一个外部组织提供。certification 指明该外部组织有资格生产该零件。

EXPRESS 描述：

```
*)
TYPE certified_item=SELECT (supplied_part_relationship);
END_TYPE;
(*
```

5.2.2.5 批准项

approval_item(批准项)赋值 approval 给 product_definition_formation，planned_effectivity，configuration_item，security_classification，change_request，change，start_request，start_work，certification，或 contract 以标识所选设计部分的批准状态。

EXPRESS 描述：

```
*)
```

```
TYPE approved_item=SELECT
    (product_definition_formation,
    product_definition,
    configuration_effectivity,
    configuration_item,
    security_classification,
    change_request,
    change,
    start_request,
    start_work,
    certification,
    contract);
END_TYPE;
(*
```

5.2.2.6 合同项

contracted_item(合同项)使一特定的 product_definition_formation 与 contracted 建立联系。

EXPRESS 描述：

```
*)
TYPE contracted_item=SELECT (product_definition_formation);
END_TYPE;
(*
```

5.2.2.7 分类项

classified_item(分类项)将 security_classification 应用到一特定的 product_definition_formation 或应用到某个特定用途的两个 product_definition 之间的关系上。

EXPRESS 描述：

```
*)
TYPE classified_item=SELECT
  (product_definition_formation,
  assembly_component_usage);
END_TYPE;
(*
```

5.2.2.8 人员组织项

person_organization_item(人员组织项)赋值 person_and_organization 给 change_request, start_request, approval, configuration_item, product, product_definition_formation, product_definition, contract, 或 security_classification. person_and_organization 的职能由5.2.5.3条给出的 cc_design_person_and_organization_correlation 函数来控制。

EXPRESS 描述：

```
*)
TYPE person_organization_item=SELECT
  (change,
  start_work,
  change_request,
```

```
  start_request,
  configuration_item,
  product,
  product_definition_formation,
  product_definition,
  contract,
  security_classification);
END_TYPE;
(*
```

5.2.2.9 日期时间项

date_time_item(日期时间项)赋值 date_and_time 给 product_definition,change_request,start_request,change,start_work,approval_person_organization,contract,security_classification 或 certification。date_and_time 的作用由5.2.5.4条给出的 cc_design_date_time_correlation 函数来控制。

EXPRESS 描述:

```
*)
TYPE date_time_item=SELECT
  (product_definition,
  change_request,
  start_request,
  change,
  start_work,
  approval_person_organization,
  contract,
  security_classification,
  certification);
END_TYPE;
(*
```

5.2.2.10 规定项

specified_item(规定项)将 specification 赋值给每个 product_definition 或 shape_aspect.

EXPRESS 描述:

```
*)
TYPE specified_item=SELECT
  (product_definition,
  shape_aspect);
END_TYPE;
(*
```

5.2.3 配置控制设计实体

5.2.3.1 配置控制设计实体定义

5.2.3.1.1 基于边的线框形状表示

edge_based_wireframe_shape_representation(基于边的线框形状表示)是一个 shape_representation,它用线框表示一个 product 的形状或形状的一部分,该线框以拓扑边为其边框。它包括所有3D(三维)曲线和拓扑实体,该实体定义了由顶点和边构成的图。

EXPRESS 描述：

```
*)
ENTITY edge_based_wireframe_shape_representation
 SUBTYPE OF (shape_representation);
WHERE
  WR1:SIZEOF(QUERY(it<*SELF.items|
      NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL',
      'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
      'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D']*TYPEOF(it))=1
      )))=0;
  WR2:SIZEOF(QUERY(it<*SELF\representation.items|
      SIZEOF(['CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL',
      'CONFIG_CONTROL_DESIGN.MAPPED_ITEM']*TYPEOF(it))=1
      ))>=1;
  WR3:SIZEOF(QUERY(ebwm<*
      QUERY(it<*SELF\representation.items|
      'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
      NOT(SIZEOF(QUERY(eb<*
      ebwm\edge_based_wireframe_model.ebwm_boundary|
      NOT(SIZEOF(QUERY(edges<*eb.ces_edges|
      NOT('CONFIG_CONTROL_DESIGN.EDGE_CURVE'IN TYPEOF(edges))))=0
      )))=0)))=0;
  WR4:SIZEOF(QUERY(ebwm<*
      QUERY(it<*SELF\representation.items|
      'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
      NOT(SIZEOF(QUERY(eb<*
      ebwm\edge_based_wireframe_model.ebwm_boundary|
      NOT(SIZEOF(QUERY(pline_edges<*QUERY(edges<*eb.ces_edges|
      'CONFIG_CONTROL_DESIGN.POLYLINE'IN
      TYPEOF(edges\edge_curve.edge_geometry))|
      NOT(SIZEOF(pline_edges\edgeedges_curve.edge_geometry\polyline.points)
      >2)))=0)))=0)))=0;
  WR5:SIZEOF(QUERY(ebwm<*
      QUERY(it<*SELF\representation.items|
      'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
      NOT(SIZEOF(QUERY(eb<*
      ebwm\edge_based_wireframe_model.ebwm_boundary|
      NOT(SIZEOF(QUERY(edges<*eb.ces_edges|
      NOT(('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN
      TYPEOF(edges.edge_start))AND
      ('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN
      TYPEOF(edges.edge_end)))))=0)))=0)))=0;
  WR6:SIZEOF(QUERY(ebwm<*
```

```
        QUERY(it<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
        NOT (SIZEOF(QUERY(eb<*
        ebwm\edge_based_wireframe_model.ebwm_boundary|
        NOT(SIZEOF(QUERY(edges<*eb.ces_edges|
        NOT(valid_wireframe_edge_curve
        (edges\edge_curve.edge_geometry,'CONFIG_CONTROL_DESIGN'))))=
        0)))=0)))=0;
  WR7:SIZEOF(QUERY(ebwm<*
        QUERY(it<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
        NOT (SIZEOF(QUERY(eb<*
        ebwm\edge_based_wireframe_model.ebwm_boundary|
        NOT(SIZEOF(QUERY(edges<*eb.ces_edges|
        NOT((valid_wireframe_vertex_point
        (edges.edge_start\vertex_point.vertex_geometry,
        'CONFIG_CONTROL_DESIGN'))AND
        (valid_wireframe_vertex_point
        (edges.edge_end\vertex_point.vertex_geometry,
        'CONFIG_CONTROL_DESIGN')))))=0)))=0)))=0;
  WR8:SIZEOF(QUERY(ebwm<*
        QUERY(it<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
        NOT (SIZEOF(QUERY(eb<*
        ebwm_based_wireframe_model.ebwm_boundary|
        NOT(SIZEOF(QUERY(con_edges<*QUERY(edges<*eb.ces_edges|
        'CONFIG_CONTROL_DESIGN.CONIC'IN
        TYPEOF (edges\edge_curve.edge_geometry))|
        NOT('CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'IN
        TYPEOF(con_edges\edge_curve.edge_geometry\conic.position))))=0
        )))=0)))=0;
  WR9:SIZEOF(QUERY(mi<*QUERY(item<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'IN TYPEOF(item))|
        NOT('CONFIG_CONTROL_DESIGN'.+
        'EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION'
        IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
        =0;
END_ENTITY;
(*
```

形式限制：

WR1：在 edge_based_wireframe_shape_representation 中的 items 应当是 edge_based_wireframe_model，mapped_item 或 axis2_placement_3d。

WR2：在 edge_based_wireframe_shape_representation 的 items 中至少有一个应当是 edge_based_wireframe_model 或 mapped_item。

WR3：在 edge_based_wireframe_model 中的每个 edge 都应当是一个 edge_curve。

WR4：构成 edge_based_wireframe_model 的 edge 的基础的每个 polyline 应当由两个以上的不同 points 来定义。

WR5：为 edge_based_wireframe_model 定义的每个 vertex 应当是一个 vertex_point。

WR6：构成 edge_based_wireframe_model 的 edge 的基础的每个 edge_geometry 应当是一个 line，circle，ellipse，parabola，hyperbola，b_spline_curve，offset_curve_3d，polyline 或 curve_replica，并且这些曲线有一个由其他曲线定义的始终如一的基准。

WR7：构成 edge_based_wireframe_model 的 edges 的基础的 vertex_geometry 应当是一个 cartesian_point 或 point_replica，并且 point_replica 应当重复另一个 point_replica 或 cartesian_point。

WR8：所有的 conics 都应当由 axis2_placement_3d 来定位。

WR9：如果在 edge_based_wireframe_shape_representation 中有一个 mapped_item，那么 mapped_item 的源应当是一个 edge_based_wireframe_shape_representation。

5.2.3.1.2 基于壳的线框形状表示

shell_based_wireframe_shape_representation（基于壳的线框形状表示）是一个 shape_representation。它通过线框表示一个 product 的形状或一部分形状。该线框定义了一个内在体积。它包括所有3D（三维）曲线以及定义了由顶点、边和环构成的图的拓扑实体。

EXPRESS 描述：

```
*)
ENTITY shell_based_wireframe_shape_representation
 SUBTYPE OF (shape_representation);
WHERE
  WR1:SIZEOF(QUERY(it<*SELF\representation.items|
       NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL',
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
       'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D']*TYPEOF(it))=1
       )))=0;
  WR2:SIZEOF(QUERY(it<*SELF\representation.items|
       SIZEOF(['CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREERAME_MODEL',
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM']*TYPEOF(it))=1))>=1;
  WR3:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
       'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
       NOT(SIZEOF(QUERY(ws<*QUERY(sb<*
       sbwm\shell_based_wireframe_model.sbwm_boundary|
       'CONFIG_CONTROL_DESIGN.WIRE_SHELL'IN TYPEOF (sb))|
       NOT(SIZEOF(QUERY(eloop<*QUERY(wsb<*
       ws_shell.wire_shell_extent|
       'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF (wsb))|
       NOT(SIZEOF(QUERY(el<*eloop\path.edge_list|
       NOT('CONFIG_CONTROL_DESIGN.EDGE_CURVE'IN
       TYPEOF (el.edge_element))))=0))))=0))))=0))))=0;
```

```
WR4:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
    'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
    NOT(SIZEOF(QUERY(ws<*QUERY(sb<*
    sbwm\shell_based_wireframe_model.sbwm_boundary|
    'CONFIG_CONTROL_DESIGN.WIRE_SHELL'IN TYPEOF (sb))|
    NOT(SIZEOF(QUERY(eloop<*QUERY(wsb<*
    ws\wire_shell.wire_shell_extent|
    'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF (wsb))|
    NOT(SIZEOF(QUERY(pline_el<*QUERY(el<*eloop.edge_list|
    'CONFIG_CONTROL_DESIGN.POLYLINE'IN TYPEOF
    (el.edge_element\edge_curve.edge_geometry))|
    NOT(SIZEOF(pline_el.edge_element\edge_curve.
    edge_geometry\polyline.points)>2)))
    =0)))=0)))=0)))=0;
WR5:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
    'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
    NOT(SIZEOF(QUERY(ws<*QUERY(sb<*
    sbwm\shell_based_wireframe_model.sbwm_boundary|
    'CONFIG_CONTROL_DESIGN.WIRE_SHELL'IN TYPEOF (sb))|
    NOT(SIZEOF(QUERY(eloop<*QUERY(wsb<*
    ws\wire_shell.wire_shell_extent|
    'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF (wsb))|
    NOT(SIZEOF(QUERY(el<*eloop\path.edge_list|
    NOT(valid_wireframe_edge_curve
    (el.edge_element\edge_curve.edge_geometry,
    'CONFIG_CONTROL_DESIGN'))))=0)))=0)))=0)))=0;
WR6:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
    'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
    NOT(SIZEOF(QUERY(ws<*QUERY(sb<*
    sbwm\shell_based_wireframe_model.sbwm_boundary|
    'CONFIG CONTROL_DESIGN.WIRE_SHELL'IN TYPEOF (sb))|
    NOT(SIZEOF(QUERY(eloop<*QUERY(wsb<*
    ws\wire_shell.wire_shell_extent|
    'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF (wsb))|
    NOT(SIZEOF(QUERY(el<*eloop\path.edge_list|
    NOT(('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN
    TYPEOF (el.edge_element.edge_start))
    AND
    ('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN
    TYPEOF(el.edge_element.edge_end)))))=0)))=0)))=0)))=0;
WR7:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
    'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
```

```
    NOT(SIZEOF(QUERY(ws<*QUERY(sb<*
    sbwm\shell_based_wireframe_model.sbwm_boundary|
    'CONFIG_CONTROL_DESIGN.WIRE_SHELL'IN TYPEOF (sb))|
    NOT(SIZEOF(QUERY(eloop<*QUERY(wsb<*
    ws\wire_shell.wire_shell_extent|
    'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF (wsb))|
    NOT(SIZEOF(QUERY(el<*eloop\path.edge_list|
    NOT((valid_wireframe_vertex_point
    (el.edge_element.edge_start\vertex_point.vertex_geometry,
    'CONFIG_CONTROL_DESIGN'))
    AND
    (valid_wireframe_vertex_point
    (el.edge_element.edge_end\vertex_point.vertex_geometry,
    'CONFIG_CONTROL_DESIGN')))))=0)))
    =0)))=0)))=0;
WR8:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
    'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
    NOT(SIZEOF(QUERY(ws<*QUERY(sb<*
    sbwm\shell_based_wireframe_model.sbwm_boundary|
    'CONFIG_CONTROL_DESIGN.WIRE_SHELL'IN TYPEOF (sb))|
    NOT(SIZEOF(QUERY(eloop<*QUERY(wsb<*
    ws\wire_shell.wire_shell_extent|
    'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF (wsb))|
    NOT(SIZEOF(QUERY(con_edges<*QUERY(el<*eloop\path.edge_list|
    'CONFIG_CONTROL_DESIGN.CONIC'IN
    TYPEOF (el.edge_element\edge_curve.edge_geometry))|
    NOT('CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'IN
    TYPEOF(con_edges.edge_element\edge_curve.edge_geometry\conic.position))))=0)))=
    0)))=0)))=0;
WR9:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
    'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
    NOT(SIZEOF(QUERY(ws<*QUERY(sb<*
    sbwm\shell_based_wireframe_model.sbwm_boundary|
    'CONFIG_CONTROL_DESIGN.WIRE_SHELL'IN TYPEOF(sb))|
    NOT(SIZEOF(QUERY(vloop<*QUERY(wsb<*
    ws\wire_shell.wire_shell_extent|
    'CONFIG_CONTROL_DESIGN.VERTEX_LOOP'IN TYPEOF(wsb))|
    NOT('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN
    TYPEOF(vloop\vertex_loop.loop_vertex))))=0)))=0)))=0;
WR10:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
     'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF
     (it))|
```

```
        NOT(SIZEOF(QUERY(ws<*QUERY(sb<*
        sbwm\shell_based_wireframe_model.sbwm_boundary|
        'CONFIG_CONTROL_DESIGN.WIRE_SHELL'IN TYPEOF(sb))|
        NOT(SIZEOF(QUERY(vloop<*QUERY(wsb<*
        ws\wire_shell.wire_shell_extent|
        'CONFIG_CONTROL_DESIGN.VERTEX_LOOP'IN TYPEOF(wsb))|
        NOT(valid_wireframe_vertex_point
        (vloop\vertex_loop.loop_vertex\vertex_point.vertex_geometry,
        'CONFIG_CONTROL_DESIGN'))))=0))))=0))))=0;
WR11:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(vs<*QUERY(sb<*
        sbwm\shell_based_wireframe_model.sbwm_boundary|
        'CONFIG_CONTROL_DESIGN.VERTEX_SHELL'IN TYPEOF(sb))|
        NOT('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN
        TYPEOF(vs\vertex_shell.vertex_shell_extent.loop_vertex))))
        =0)))=0;
WR12:SIZEOF(QUERY(sbwm<*QUERY(it<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(vs<*QUERY(sb<*
        sbwm_based_wireframe_model.sbwm_boundary|
        'CONFIG_CONTROL_DESIGN.VERTEX_SHELL'IN TYPEOF(sb))|
        NOT(valid_wireframe_vertex_point
        (vs\vertex_shell.vertex_shell_extemt.loop_vertex\
        vertex_point.vertex_geometry,'CONFIG_CONTROL_DESIGN'))))
        =0)))=0;
WR13:SIZEOF(QUERY(mi<*QUERY(item<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'IN TYPEOF(item))|
        NOT('CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_
        REPRESENTATION'
        IN TYPEOF(mi\mapped_item.mapping_source.mapped_representation))))
        =0;
END_ENTITY;
(*
```

形式限制：

WR1：在 shell_based_wireframe_shape_representation 中的 items 应当是一个 shell_based_wireframe_model，mapped_item 或 axis2_placement_3d。

WR2：在 shell_based_wireframe_shape_representation 中的 items 至少有一个应当是 shell_based wireframe_model 或 mapped_item。

WR3：为 shell_based_wireframe_model 中的 edge_loop 定义的每条 edge 都应当是一个 edge_curve。

WR4：构成 shell_based_wireframe_model 中 edge 的每个 polyline 应当包含两个以上的不同 points.

WR5：构成 shell_based_wireframe_model 中 edge 的 edge_geometry 应当是一个 line，circle，ellipse，

parabola,hyperbola,b_spline_curve,offset_curve_3d,polyline 或 curve_replica,并且这些曲线有一个由其他曲线定义的始终如一的基准。

WR6:作为 shell_based_wireframe_model 中 edge 的初始点和结束点的每个 vertex 应当是一个 vertex_point。

WR7:各顶点定义了在 shell_based_wireframe_model 的一个 edge_loop 中的 edge 的边界,构成这些顶点的 vertex_geimetry 应当是一个 cartesian_point 或 point_replica,并且 point_replica 应当重复另一个 point_replica 或 cartesian_point。

WR8:构成 shell_based_wireframe_model 的 edge_loop 中一条 edge 的 conic 的 conic.position 只能是一个 axis2_placement_3d。

WR9:定义用作 shell_based_wireframe_model 边界的 vertex_loop 的顶点应当是一个 vertex_point。

WR10:定义用作 shell_based_wireframe-model 边界的 vertex_loop 的顶点应作为 cartesian_point 或 point_replica 的基础,并且 point_replica 应重复另一个 point_replica 或 cartesian_point。

WR11:定义 vertex_loop 的顶点应当是一个 vertex_point,它应当作为 shell_based_wireframe_model 中 vertex_shell 的 vertex_shell_extent。

WR12:定义 vertex_loop 的 vertex 应当作为 cartesian_point 或 point_replica 的基础,它用作 shell_based_wireframe_model 中 vertex_shell 的 vertex_shell_extent,并且 point_replica 应重复另一个 point_replica 或 cartesian_point。

WR13:如果在 shell_based_wireframe_shape_representation 中有一个 mapped_item,则 mapped_item 的源应当是一个 shell_based_wireframe_shape_representation。

5.2.3.1.3 几何有界的曲面形状表示

geometrically_bounded_surface_shape_representation(几何有界的曲面形状表示)是一个 shape_representation,它通过无拓扑曲面几何表示一个 product 的形状或部分形状。geometrically_bounded_surface_shape_representation 只包含 point,curve 和 surface。所有无界的曲线和曲面都被显式地裁剪。曲线的边界由该曲线上的点以及这些点和由这些点所限制的曲线间的显式缔合来定义,或由参数值来定义。曲面的边界由曲面上的曲线以及这些曲线和由这些曲线所限制的曲面间的显示缔合来定义,或由参数值来定义。

geometrically_bounded_surface_shape_representation 由一个或几个 geometric_set(几何集合)来表示。每个 geometric_set 只包含定义物理对象的那些 point,curve 和 surface,物理对象又由 geometrically_bounded_surface_shape_representation 的一个特定实例来表示。用来支持另一个几何实体定义的几何实体本身不是 geometric_set 的元素。

例子 21:在一个用 geometrically_bounded_surface_shape_representation 表示的零件上,用一圆弧定义该零件角半径曲面的边界。圆弧用 circle 来表示,并被 trimmed_curve 引用为 basic_curve。包含零件形状的 geometric_set 在它的 elements 集合中有 trimmed_curve, 可是不包含用作 basic_curve 的那个 circle。

一个 geometrically_bounded_surface_shape_representation 的 items 也可以是 mapped_item 或 axis2_placement_3d 类型。这是为了实现 geometrically_bounded_surface_shape_representation 的装配件表示。

下列约束限制了 geometrically_bounded_surface_shape_representation 实体数据类型的使用。为了能使这些限制适用于每个给定模型的实例上,对请求某一实体数据类型的所有实例,都需跟踪该模型。geometrically_bounded_surface_shape_representation 的结构不是层次的,而是网状的。curve 可以引用其他 curve,在 pcurve 的情况下,也可以引用 surface。point 可以引用 curve。对不确定的网状结构导向方法基于三个函数,它们使用递归方法:

——gbsf_check_point;

— gbsf_check_curve;

— gbfs_check_surface;

这些函数中的每一个从其目录分支内进入 geometrically_bounded_surface_shape_representation 范围内的所有实体类型。对这些类型的任何引用将通过调用上述三个函数之一进行进一步调查，这个过程可能是递归的。在每个分支中，检查适用于一个实体类型的所有限制，并返回一个布尔值。对 geometrically_bounded_surface_shape_representation 中的所有 geometric_set 的所有元素调用这些函数。

EXPRESS 描述：

```
*)
ENTITY geometrically_bounded_surface_shape_representation
    SUBTYPE OF(shape_representation);
WHERE
    WR1:SIZEOF(QUERY(it<*items|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.GEOMETRIC_SET',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
        'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D']*TYPEOF(it))=1)))=0;
    WR2:SIZEOF(QUERY(it<*items|
        SIZEOF(['CONFIG_CONTROL_DESIGN.GEOMETRIC_SET',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM']*TYPEOF(it))=1))>0;
    WR3:SIZEOF(QUERY(mi<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'IN TYPEOF(it))|
        NOT('CONFIG_CONTROL_DESIGN.'+
        'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION'
        IN TYPEOF(mi\mapped_item.mapping_source.mapped_representation))))
        =0;
    WR4:SIZEOF(QUERY(gs<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.GEOMETRIC_SET'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(pnt<*QUERY(gesl<*
        gs\geometric_set.elements|
        'CONFIG_CONTROL_DESIGN.POINT'IN TYPEOF(gsel))|
        NOT(gbsf_check_point(pnt,'CONFIG_CONTROL_DESIGN'))))=0)))=0;
    WR5:SIZEOF(QUERY(gs<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.GEOMETRIC_SET'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cv<*QUERY(gsel<*
        gs\geometric_set.elements|
        'CONFIG_CONTROL_DESIGN.CURVE'IN TYPEOF(gsel))|
        NOT(gbsf_check_curve(cv,'CONFIG_CONTROL_DESIGN'))))=0)))=0;
    WR6:SIZEOF(QUERY(gs<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.GEOMETRIC_SET'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(sf<*QUERY(gsel<*
        gs\geometric_set.elements|
        'CONFIG_CONTROL_DESIGN.SURFACE'IN TYPEOF(gsel))|
        NOT(gbsf_check_surface(sf,'CONFIG_CONTROL_DESIGN'))))=0)))=0;
```

END_ENTITY;

(*

形式限制：

WR1：geometrically_bounded_surface_shape_representation 的项应当是一个 geometric_set，mapped_item 或 axis2_placement_3d。

WR2：geometrically_bounded_surface_shape_representation 的项至少有一个应当是 geometric_set 或 mapped_item。

WR3：如果在 geometrically_bounded_surface_shape_representation 有一个 mapped_item 项，那么它的 mapping_source 的 mapped_representation 应当是一个 geometrically_bounded_surface_shape_representation。

WR4：作为一个 geometrically_bounded_surface_shape_representation 的 items 中之一的 geometric_set，其 elements 中的每个点都应当与这种表示的定义相一致。

WR5：作为一个 geometrically_bounded_surface_shape_representation 的 items 中之一的 geometric_set，其 element 中的每个曲线都应当与这种表示的定义相一致。

WR6：作为一个 geometrically_bounded_surface_shape_representation 的 items 中之一的 geometric_set，其 element 中的每个曲面都应当与这种表示的定义相一致。

非形式限制：

IP1：在几何定义网状结构中引用的 b_spline_curve 不应自交。

IP2：在几何定义网状结构中引用的 composite_curve 不应自交。

IP3：在几何定义网状结构中引用的 offset_curve_2d 不应自交。

IP4：在几何定义网状结构中引用的 offset_curve_3d 不应自交。

IP5：在几何定义网状结构中引用的 b_spline_surface 不应自交。

IP6：在几何定义网状结构中引用的 offset_surface 不应自交。

IP7：专用来支持其他几何实体定义的几何实体自身不存在于任何 geometric_set 的各元素的各集合中。

5.2.3.1.4 几何有界的线框形状表示

geometrically_bounded_wireframe_shape_representation（几何有界的线框形状表示）是一个 shape_representation，它通过无拓扑线框几何表示一个 product 的形状或部分形状。这些表示只能使用点和线来构成。所有无界的曲线都被显式的裁剪，除非它们是封闭的。曲线的边界由曲线上的点、以及这些点和由这些点所限制的曲线间的显式缔合来定义，或由参数值来定义。geometrically_bounded_surface_shape_representation 中的每个几何集 geometric_set 只包含那些定义物理对象的实体，该物理对象由 geometrically_bounded_wireframe_shape_representation 的一个特定实例来表示。用来支持另一个几何实体定义的几何实体自身不存在于 geometric_set 的 elements 集中。

例子 22：在一个用 geometrically_bounded_wireframe_shape_representation 表示的零件上，用圆弧定义零件的角半径。该圆弧用被 trimmed_curve 引用为 basic_curve 的 circle 来表示。包含零件形状的 geometric_set 在它的 element 集合中有 trimmed_curve，但不包含用作 basic_curve 的那个 circle。

EXPRESS 描述：

*)

```
ENTITY geometrically_bounded_wireframe_shape_representation
    SUBTYPE OF(shape_representation);
WHERE
    WR1:SIZEOF(QUERY(it<*SELF\representation.items|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET',
```

```
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
        'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D']*TYPEOF(it))=1)))=0;
    WR2:SIZEOF(QUERY(it<*SELF\representation.items|
        SIZEOF(['CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM']*TYPEOF(it))=1
        ))>=1;
    WR3:SIZEOF(QUERY(gcs<*QUERY(it<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(elem<*gcs\geometric_set.elements|
        NOT(valid_geometrically_bounded_wf_curve
        (elem,'CONFIG_CONTROL_DESIGN'))))=0)))=0;
    WR4:SIZEOF(QUERY(gcs<*QUERY(it<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(pnts<*QUERY(elem<*
        gcs\geometric_set.elements|
        'CONFIG_CONTROL_DESIGN.POINT'IN TYPEOF(elem))|
        NOT(valid_geometrically_bounded_wf_point
        (pnts,'CONFIG_CONTROL_DESIGN'))))=0)))=0;
    WR5:SIZEOF(QUERY(gcs<*QUERY(it<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cnc<*QUERY(elem<*
        gcs\geometric_set.elements|
        'CONFIG_CONTROL_DESIGN.CONIC'IN TYPEOF(elem))|
        NOT('CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'IN TYPEOF
        (cnc.position))))=0)))=0;
    WR6:SIZEOF(QUERY(gcs<*QUERY(it<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(pline<*QUERY(elem<*
        gcs\geometric_set.elements|
        'CONFIG_CONTROL_DESIGN.POLYLINE'IN TYPEOF(elem))|
        NOT(SIZEOF(pline.points)>2)))=0)))=0;
    WR7:SIZEOF(QUERY(mi<*QUERY(item<*SELF\representation.items|
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'IN TYPEOF(item))|
        NOT('CONFIG_CONTROL_DESIGN.'+
        'GEOMETRICALLY_BOUNDED_WIREFRAME_REPRESENTATION'
        IN TYPEOF(mi\mapped_item.mapping_source.mapped_representation))))
        =0;
END_ENTITY;
(*
```

形式限制：

WR1：geometrically_bounded_wireframe_shape_representat 中的项应当是一个 geometric_set，mapped_item 或 axis2_placement_3d。

WR2：在 geometrically_bounded_wireframe_shape_representat 中至少其 items 之一应当是 geometric_curve_set 或 mapped_item。

WR3：在一 geometrically_bounded_wireframe_shape_representat 的一 geometric_curve_set 的 elements 中的所有曲线应当通过几何明确地界定。

WR4：在一 geometrically_bounded_wireframe_shape_representation 的 geometric_curve_set 的 elements 中的所有点应当是一个 cartesian_point，point_on_curve，或 point_replica。该 point_replica 将重复另一个 point_replica 或 cartesian_point。point_on_curve 应位于一个 geometrically_bounded_wireframe_shape_representation 的一个合法曲线类型上。

WR5：在一 geometrically_bounded_wireframe_shape_representation 的 geometric_set 的 elements 中的 conic 的 position 只应该是一个 axis2_placement_3d。

WR6：在一 geometrically_bounded_wireframe_shape_representation 的 geometric_set 的 elements 中的每条 polyline 应当包含两个以上的不同点。

WR7：如果在一 geometrically_bounded_wireframe_shape_representation 中有一个 mapped_item 项，则该 mapped_item 的源应当是一个 geometrically_bounded_wireframe_shape_representation。

5.2.3.1.5 小平面边界表示形状表示

faceted_brep_shape_representation(小平面边界表示形状表示)是一个 shape_representation 类型，在其中，representation_item 是 faced_brep 实体的特例。这些边界表示只有平面和隐式边几何，这就是它们与比较通用的边界表示的区别。

EXPRESS 描述：

```
*)
ENTITY faceted_brep_shape_representation
SUBTYPE OF(shape_representation);
WHERE
    WR1:SIZEOF(QUERY(it<*items|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.FACETED_BREP',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
        'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D']*TYPEOF(it))=1)))=0;
    WR2:SIZEOF(QUERY(it<*items|
        SIZEOF(['CONFIG_CONTROL_DESIGN.FACETED_BREP',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM']*TYPEOF(it))=1))>0;
    WR3:SIZEOF(QUERY(fbrep<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.FACETED_BREP'IN TYPEOF(it))|
        NOT (SIZEOF(QUERY(csh<*
        msb_shells(fbrep,'CONFIG_CONTROL_DESIGN')|
        NOT(SIZEOF(QUERY(fcs<*csh.cfs_faces|
        NOT(('CONFIG_CONTROL_DESIGN.FACE_SURFACE'IN TYPEOF(fcs))
        AND
        (('CONFIG_CONTROL_DESIGN.PLANE'IN TYPEOF
        (fcs\face_surface.face_geometry))
        AND
        ('CONFIG_CONTROL_DESIGN.CARTESIAN_POINT'IN
        TYPEOF(fcs\face_surface.face_geometry\plane.position.location)))))))
        =0)))=0)))=0;
```

```
    WR4:SIZEOF(QUERY(fbrep<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.FACETED_BREP'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(csh<*
        msb_shells(fbrep,'CONFIG_CONTROL_DESIGN')|
        NOT(SIZEOF(QUERY(fcs<*csh.cfs_faces|
        NOT(SIZEOF(QUERY(bnds<*fcs.bounds|
        'CONFIG_CONTROL_DESIGN.FACE_OUTER_BOUND'IN TYPEOF(bnds)))=1)))
        =0)))=0)))=0;
    WR5:SIZEOF(QUERY(msb<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP'IN TYPEOF(it))|
        'CONFIG_CONTROL_DESIGN.ORIENTED_CLOSED_SHELL'IN
        TYPEOF(msb\manifold_solid-brep.outer)))=0;
    WR6:SIZEOF(QUERY(brv<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.BREP_WITH_VOIDS'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(csh<*brv\brep_with_voids.voids|
        csh\oriented_closed_shell.orientation))=0)))=0;
    WR7:SIZEOF(QUERY(mi<* QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'IN TYPEOF(it))|
        NOT ('CONFIG_CONTROL_DESIGN.FACETED_BAEP_SHAPE_
        REPRESENTATION'IN
        TYPEOF(mi\mapped_item.mapping_source.mapped_representation))))
        =0;
END_ENTITY;
(*
```

形式限制：

WR1：表示超类的 items 属性应只包含 faceted_brep，mapped_item 或 axis2_placement_3d 的实例。

WR2：在 items 集合中至少有一个 representation_item 应当是 faceted_brep 实体或 mapped_item。

WR3：每个 faceted_brep 其 items 集合应该由具有 face_surface 类型的各面来界定，它仅通过 planes 完成几何定义，该 planes 为其各自的 location 使用一个 cartesian_point。

WR4：对于 faceted_brep 的每个面要规定明确的外边界。

WR5：对于每个 manifold_solid_brep，其 items 集合中的 outer_shell 不应当是有向的子类型。

WR6：如果一个 brep_with_voids 被包含在该 items 集合中，那么 voids 集合中的每个壳应当是一个具有 orientation 值为 FALSE(假)的 oriented_closed_shell。

WR7：如果一个 mapped_item 被包含在项集合中，那么被映射的表示类型应当是一个 faceted_brep shape_representation。

5.2.3.1.6 流形曲面形状表示

manifold_surface_shape_representation(流形曲面形状表示)是一个 shape_representation，它使用具有边界的二维流形来表示一个产品的形状或部分形状。以某种方式拓扑地限制二维流形，以使其适于包含在实体表示中。

manifold_surface_shape_representation 由壳的集合构成，每个壳又由面集合构成。这些 faces 使用了 edges，以及按顺序引用几何实体的 vertexs。应使用 open_shell 和 closed_shell，二者都是 connected_face_set 的子类型。不能单独实例化 connected_face_set。拓扑实体不能脱离其对应的几何域而存在。

manifold_surface_shape_representation 的 items 也可以是 mapped_item 或 axis2_placement_3d 类型。用这些实体来实现 manifold_surface_shape_representation 的各装配表示。

所有无界几何应使用拓扑结构进行裁剪。

EXPRESS 描述：

```
*)
ENTITY manifold_surface_shape_representation
    SUBTYPE OF (shape_representation);
WHERE
    WR1:SIZEOF(QUERY(it<*items|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
        'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D']*TYPEOF(it))=1)))=0;
    WR2:SIZEOF(QUERY(it<*items|
        SIZEOF(['CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL',
        'CONFIG_CONTROL_DESIGN.MAPPED ITEM']*TYPEOF(it))=1))>0;
    WR3:SIZEOF(QUERY(mi<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.MAPPED ITEM'IN TYPEOF(it))|
        NOT ('CONFIG_CONTROL_DESIGN.MANIFOLD_SURFACE_SHAPE_
        REPRESENTATION'
        IN TYPEOF(mi\mapped_item.mapping_source.mapped_representation))))
        =0;
    WR4:SIZEOF(QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(sh<*
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.OPEN_SHELL',
        'CONFIG_CONTROL_DESIGN.CLOSED_SHELL']
        *TYPEOF(sh))=1)))=0)))=0;
    WR5:SIZEOF (QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cfs<*
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(QUERY(fa<*cfs.cfs_faces|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.FACE_SURFACE',
        'CONFIG_CONTROL_DESIGN.ORIENTED_FACE']*TYPEOF(fa))=1)))=0)))
        =0)))=0;
    WR6:SIZEOF(QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cfs<*
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(QUERY(f_sf<*QUERY(fa<*cfs.cfs_faces|
        'CONFIG_CONTROL_DESIGN.FACE_SURFACE'IN TYPEOF(fa))|
```

```
        NOT(('CONFIG_CONTROL_DESIGN.ADVANCED_FACE'IN TYPEOF(f_sf))
        OR
        (SIZEOF(['CONFIG_CONTROL_DESIGN.OFFSET_SURFACE',
        'CONFIG_CONTROL_DESIGN.SURFACE_REPLICA']*TYPEOF
        (f_sf\face_surface.face_geometry))=1))))=0)))=0)))=0;
    WR7:SIZEOF(QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cfs<*
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(QUERY(fa<*cfs.cfs_faces|
        NOT(('CONFIG_CONTROL_DESIGN.ADVANCED_FACE'IN TYPEOF(fa))
        OR
        (basis_surface_check(fa.surface.face_geometry,
        'CONFIG_CONTROL_DESIGN')))))=0)))=0)))=0;
    WR8:SIZEOF(QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZE OF (QUERY(cfs<*
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(QUERY(fa<*cfs.cfs_faces|
        NOT(('CONFIG_CONTROL_DESIGN.ADVANCED_FACE'IN TYPEOF(fa))
        OR
        (SIZEOF(QUERY(bnds<*fa.bounds|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.EDGE_LOOP',
        'CONFIG_CONTROL_DESIGN.VERTEX_LOOP']
        *TYPEOF(bnds.bound))=1)))=0))))=0)))=0)))=0;
    WR9:SIZEOF(QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cfs<*
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(QUERY(fa<*cfs.cfs_faces|
        NOT(('CONFIG_CONTROL_DESIGN.ADVANCED_FACE'IN TYPEOF(fa))
        OR
        (SIZEOF(QUERY(elp_fbnds<*QUERY(bnds<*fa.bounds|
        'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF(bnds))|
        NOT(SIZEOF(QUERY(oe<*elp_fbnds.bound\path.edge_list|
        NOT('CONFIG_CONTROL_DESIGN.EDGE_CURVE'IN TYPEOF(oe.edge_element))))
        =0))))=0)))))=0))))=0)))=0;
    WR10:SIZEOF(QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cfs<*
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(QUERY(fa<*cfs.cfs_faces|
```

```
        NOT(('CONFIG_CONTROL_DESIGN.ADVANCED_FACE'IN TYPEOF(fa))
        OR
        (SIZEOF(QUERY(elp_fbnds<*QUERY(bnds<*fa.bounds|
        'CONFIG_CONTROL_DESIGN.EDGELOOP'IN TYPEOF(bnds.bound))|
        NOT(SIZEOF(QUERY(oe_cv<*QUERY(oe<*
        elp_fbnds.bound\path.edge_list|
        'CONFIG_CONTROL_DESIGN.EDGE_CURVE'IN TYPEOF(oe.edge_element))|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.CURVE_REPLICA',
        'CONFIG_CONTROL_DESIGN.OFFSET_CURVE_3D',
        'CONFIG_CONTROL_DESIGN.SURFACE_CURVE']*
        TYPEOF(oe_cv.edge_element\edge_curve.edge_geometry))
        =1)))=0)))=0))))=0)))=0)))=0;
WR11:SIZEOF(QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cfs<*
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(QUERY(fa<*cfs.cfs_faces|
        NOT(('CONFIG_CONTROL_DESIGN.ADVANCED_FACE'IN TYPEOF(fa))
        OR
        (SIZEOF(QUERY(elp_fbnds<*QUERY(bnds<*fa.bounds|
        'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF(bnds.bound))|
        NOT(SIZEOF(QUERY(oe<*elp_fbnds.bound\path.edge_list|
        NOT(basis_curve_check(oe.edge_element\edge_curve.edge_geometry,
        'CONFIG_CONTROL_DESIGN'))))=0)))=0))))=0)))=0)))=0;
WR12:SIZEOF(QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cfs<*
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(QUERY(fa<*cfs.cfs_faces|
        NOT(('CONFIG_CONTROL_DESIGN.ADVANCED_FACE'IN TYPEOF(fa))
        OR
        (SIZEOF(QUERY(elp_fbnds<*QUERY(bnds<*fa.bounds|
        'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF(bnds))|
        NOT(SIZEOF(QUERY(oe<*elp_fbnds.bound\path.edge_list|
        NOT(('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN TYPEOF
        (oe.edge_element.edge_start))
        AND
        ('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN
        TYPEOF(oe.edge_element.edge_end)))))=0)))=0))))=0)))=0)))=0;
WR13:SIZEOF(QUERY(sbsm<*QUERY(it<*items|
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(cfs<*
```

```
        sbsm\shell_based_surface_model.sbsm_boundary|
        NOT(SIZEOF(QUERY(fa<*cfs.cfs_faces|
        NOT(('CONFIG_CONTROL_DESIGN.ADVANCED_FACE'IN TYPEOF(fa))
        OR
        (SIZEOF(QUERY(elp_fbnds<*QUERY(bnds<*fa.bounds|
        'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF(bnds.bound))|
        NOT(SIZEOF(QUERY(oe<*elp_fbnds.bound\path.edge_list|
        NOT((SIZEOF(['CONFIG_CONTROL_DESIGN.CARTESIAN_POINT',
        'CONFIG_CONTROL_DESIGN.DEGENERATE_PCURVE',
        'CONFIG_CONTROLPCURVE',
        'CONFIG_CONTROLPCURVE',
        'CONFIG_CONTROL_DESIGN.POINT_ON_CURVE',
        'CONFIG_CONTROL_DESIGN.POINT_ON_SURFACE']*TYPEOF
        (oe.edge_element.edge_start\vertex_point.vertex_geometry))=1)
        AND
        (SIZEOF(['CONFIG_CONTROL_DESIGN.CARTESIAN_POINT',
        'CONFIG_CONTROL_DESIGN.DEGENERATE_PCURVE',
        'CONFIG_CONTROL_DESIGN.POINT_ON_CURVE',
        'CONFIG_CONTROL_DESIGN.POINT_ON_SURFACE']*TYPEOF
        (oe.edge_element.edge_end\vertex_point,vertex_geometry))=1
        ))))=0)))=0))))=0)))=0)))=0;
END_ENTITY;
(*
```

形式限制：

WR1：manifold_surface_shape_representation 中的 items 应当是一个 shell_based_shape_model，mapped_item 或 axis2_placement_3d。

WR2：manifold_surface_shape_representation 中的 items 至少应有一个是 shell_based_surface_model 或 mapped_item。

WR3：如果 manifold_surface_shape_representation 中有一个 mapped_item，则其 mapping_source 的 mapped_representation 应当是一个 manifold_surface_shape_representation。

WR4：作为 manifold_surface_shape_representation 的项之一的 shell_based_surface_model，它像它的 sbsm_boundary 一样只引用 open_shell 或 closed_shell。

WR5：作为 manifold_surface_shape_representation 中 connected_face_set 的 cfs_faces 之一的 face 应当是一个 face_surface 或一 oriented_face。

WR6：一 face_surface 的 face_geometry 是一 manifold_surface_shape_representation 的一 connected_face_set 中的一个 face，该 face_geometry 应该是一个 advanced_face 引用树中的 surface，或者是一个 offset_surface，或者是一个 surface_replica。

WR7：face_surface 是 manifold_surface_shape_representation 的引用树中 connected_face_set 的一部分，它或者应在一个 advanced_face 的引用树中，或者引用合法的 surfaces 中的一个。

特别地，一个 offset_surface 的 basic_surface 应当是一个 elementary_surface，一个 b_spline_surface，一个 offset_surface，一个 swept_surface，或一个 surface_replica 之一。

一个 surface_replica 的 parent_surface 应当是一个 elementary_surface，一个 b_spline_surface，一个

offset_surface，一个 swept_surface，或一个 surface_replica。

WR8：一个 face_bound 的 bound 是 manifold_surface_shape_representation 中 shell_based_surface_model 的 connected_face_set 中的一个 face 的 bounds 之一，它或者应在一个 advanced_face 的引用树中，或者应是一个 edge_loop 或一个 vertex_loop。

WR9：作为一 manifold_surface_shape_representation 的引用树中一个 connected_face_set 的一部分的一个 oriented_edge 的 edge_element，它或者在 advanced_face 的引用树中，或者应是一个 edge_curve。

WR10：edge_curve 是 edge_loop 中的一个 edge，它的 edge_geometry 应当是一个 advanced_face 的引用树中的一个 curve，或一个 surface_curve，或一个 offset_curve_3d，或 curve_replica。而其中的 edge_loop 又是 manifold_surface_shape_representation 中 connected_face_set 中一个 face 的各 bounds 之一。

WR11：edge_curve 是 manifold_surface_shape_representation 中引用树中 connected_face_set 的一部分，它应引用一个合法的 curve。

特别地，一个 curve_replica 的 parent_curve 应当或者是一个 line，一个 conic，一个 pcurve，一个 surface_curve，一个 offset_curve_3d，一个 b_spline_curve，一个 polyline，或者应是一个 curve_replica。

offset_curve_3d 的 basic_curve 或者应是一个 line，一个 conic，一个 pcurve，surface_curve，一个 offset_curve_3d，一个 b_spline_curve，一个 polyline，或者应是一个 curve_replica。

一个 definitional_representation 是一个 pcurve 的 reference_to_curve，它的 items 集合中的一个元素也应当是一个合法的 curve。

surface_curve 的 curve_3d 应当是一个合法的 curve。

WR12：一个 edge 是一个 manifold_surface_shape_representation 的引用树中 connected_face_set 的一部分，它的 edge_start 和 edge_end 或者应在 advanced_face 的引用树中，或者是一个 vertex_point。

WR13：一个 vertex 是一个 manifold_surface_shape_representation 的引用树中 connected_face_set 的一部分，它的 vertex_geometry 或者应在一个 advanced_face 的引用树中，或者应是一个 cartesian_point，一个 point_on_curve，一个 point_on_surface，或者 degenerate_pcurve。

非形式限制：

IP1：一个 b_spline_curve 不应当自交。

IP2：一个 b_spline_surface 不应当自交。

IP3：一个 offset_curve 不应当自交。

IP4：一个 offset_surface 不应当自交。

IP5：在一个 manifold_surface_shape_representation 的一个 connected_face_set 中的一个 face 的各 bounds 应当表示为 face_outer_bound，该 manifold_surface_shape_representation 可以清楚地标识为那个 face 的各个外界限。

5.2.3.1.7 高级的表面

advanced_face(高级的表面)是 face_surface 的一种类型，限制 face_surface 以保证直接和完整地定义该几何。该高级的表面是顶层级的实体，它用来阐明一个拓扑地有界曲面的精确含义。

EXPRESS 描述：

```
*)
ENTITY advanced_face
    SUBTYPE OF(face_surface);
WHERE
    WR1:SIZEOF(['CONFIG_CONTROL_DESIGH.ELEMENTARY_SURFACE',
        'CONFIG_CONTROL_DESIGN.B_SPLINE_SURFACE',
        'CONFIG_CONTROL_DESIGN.SWEPT_SURFACE']*
        TYPEOF(face_geometry))=1;
```

```
WR2:SIZEOF(QUERY(bnd<*bounds|
    NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.EDGE_LOOP',
    'CONFIG_CONTROL_DESIGN.VERTEX_LOOP']*TYPEOF(bnd))=1)))=0;
WR3:SIZEOF(QUERY(elp_fbnds<*QUERY(bnds<*bounds|
    'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF(bnds.bound))|
    NOT(SIZEOF(QUERY(oe<*elp_fbnds.bound\path.edge_list|
    NOT('CONFIG_CONTROL_DESIGN.EDGE_CURVE'IN
    TYPEOF(oe.edge_element))))=0)))=0;
WE4:SIZEOF(QUERY(elp_fbnds<*QUERY(bnds<*bounds|
    'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF(bnds.bound))|
    NOT(SIZEOF(QUERY(oe<*elp_fbnds.bound\path.edge_list|
    NOT(('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN TYPEOF(oe.edge_start))
    AND
    ('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN TYPEOF(oe.edge_end)))))
    =0)))=0;
WR5:SIZEOF(QUERY(elp_fbnds<*QUERY(bnds<*bounds|
    'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF(bnds.bound))|
    NOT(SIZEOF(QUERY(oe<*elp_fbnds.bound\path.edge_list|
    NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.LINE',
    'CONFIG_CONTROL_DESIGN.CONIC',
    'CONFIG_CONTROL_DESIGN.POLYLINE',
    'CONFIG_CONTROL_DESIGN.PCURVE',
    'CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE']*
    TYPEOF(oe.edge_element_curve.edge_geometry))=1)))=0)))=0;
WR6:SIZEOF(QUERY(elp_fbnds<*QUERY(bnds<*bounds|
    'CONFIG_CONTROL_DESIGN.EDGE_LOOP'IN TYPEOF(bnds.bound))|
    NOT('CONFIG_CONTROL_DESIGN.ORIENTED_PATH'IN
    TYPEOF(elp_fbnds.bound))))=0;
WR7:(NOT('CONFIG_CONTROL_DESIGN.SWEPT_SURFACE'IN
    TYPEOF(face_geometry)))
    OR
    (SIZEOF(['CONFIG_CONTROL_DESIGN.LINE',
    'CONFIG_CONTROL_DESIGN.CONIC',
    'CONFIG_CONTROL_DESIGN.POLYLINE',
    'CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE']*
    TYPEOF(face_geometry\swept_surface.swept_curve))=1);
WR8:SIZEOF(QUERY(vlp_fbnds<*QUERY(bnds<*bounds|
    'CONFIG_CONTROL_DESIGN.VERTEX_LOOP'IN TYPEOF(bnds.bound))|
    NOT(('CONFIG_CONTROL_DESIGN.VERTEX_POINT'IN
    TYPEOF(vlp_fbnds.bound\vertex_loop.loop_vertex))AND
    ('CONFIG_CONTROL_DESIGN.CARTESIAN_POINT'IN
    TYPEOF
```

```
        (vlp_fbnds.bound\vertex_loop.loop_vertex\vertex_point.vertex_geometry)
        ))))=0;
END_ENTITY;
(*
```

形式限制：

WR1：在该面壳的定义中使用的几何应当受限制。该面几何应当是一个 elementary_surface，一个 swept_surface 或一个 b_spline_surface。

WR2：一个 advanced_face 的界限应当由各 vertex_loop 或者由 edge_loop 来定义。

WR3：该面的所有的界定边的几何应当完整地定义为各 edge_curve。

WR4：在该面定义中使用的所有顶点应当是 vertex_point 类型的，该面定义有一个由 edge_loop 定义的边界。

WR5：用于定义各边几何的曲线各类型应被限制为 line，conic，polyline，pcurve 或 b_spline_curve。

WR6：在该 advanced_face 的各 edge_loop 的定义中，禁止使用 oriented_path。

WR7：如果面几何是 swept_surface 类型的，那么在该定义中使用的 swept_curve 应当是 line 类型的、conic 类型的、polyline 类型的，或者是 b_spline_curve 类型的。

WR8：位于一个 vertex_loop 下列所有顶点应当是 vertex_point 类型的，应通过一个 cartesian_point 来支承 vertex_point，vertex_loop 作为该 advanced_face 的一个界限被定义。

5.2.3.1.8 高级的边界形状表示

advanced_brep_shape_representation（高级的边界形状表示）是 shape_representation 一种类型，在该 shape_representation 中，各 representation_items 是各 monifold_solid_brep 的特例。对于它们的各面和各边只有显式的几何形状，advanced_brep_shape_representation 不同与比较通用的边界表示。该面几何被限制为各元素曲面、各扫成曲面或各 B 样条曲面。

EXPRESS 描述：

```
*)
ENTITY advanced_brep_shape_representation
SUBTYPE OF(shape_representation);
WHERE
    WR1:SIZEOF(QUERY(it<*SELP.items|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP',
        'CONFIG_CONTROL_DESIGN.FACETED_BREP',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
        'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D']*TYPEOF(it))=1)))
        =0;
    WR2:SIZEOF(QUERY(it<*SELF.items|
        SIZEOF(['CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM']*TYPEOF(it))=1))>0;
    WR3:SIZEOF(QUERY(msb<*QUERY(it<*SELF.items|
        'CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP'IN TYPEOF(it))|
        NOT(SIZEOF(QUERY(csh<*
        msb_shells(msb,'CONFIG_CONTROL_DESIGN')|
        NOT(SIZEOF(QUERY(csf<*csh.cfs_faces|
        NOT('CONFIG_CONTROL_DESIGN.ADVANCED_FACE'IN TYPEOF(csf))))=0
```

```
    )))=0)))=0;
  WR4:SIZEOF(QUERY(msb<*QUERY(it<*items|
      'CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP'IN TYPEOF(it))|
      'CONFIG_CONTROL_DESIGN.ORIENTED_CLOSED_SHELL'IN
      TYPEOF(msb\manifold_solid_brep.outer)))=0;
  WR5:SIZEOF(QUERY(brv<*QUERY(it<*items|
      'CONFIG_CONTROL_DESIGN.BREP_WITH_VOIDS'IN TYPEOF(it))|
      NOT(SIZEOF(QUERY(csh<*brv\brep_with_voids.voids|
      csh\oriented_closed_shell.orientation))=0)))=0;
  WR6:SIZEOF(QUERY(mi<*QUERY(it<*items|
      'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'IN TYPEOF(it))|
      NOT('CONFIG_CONTROL_DESIGN.ADVANCED_BREP_SHAPE_
      REPRESENTATION'IN
      TYPEOF(mi\mapped_item.mapping_source.mapped_representation))))=0;
END_ENTITY;
(*
```

形式限制：

WR1：表示超类的属性可以只包含 manifold_solid_brep，mapped_item（映射项）和 axis2_placement_3d 的各实例。

注：一个 faceted_brep 不包含在该各项 SET 中，因为它应当是 faceted_brep 类型的和 manifold_solid_brep 类型的，这使得该交集的大小为 2。

WR2：在该各 items 集合中，至少一个 representation_item 应当是一个 manifold_solid_brep 实体或是一个 mapped_item。

WR3：在构造一个 manifold_solid_brep 中使用的所有面应当是 advanced_face 类型的。

WR4：对于每个 manifold_solid_brep，该各 item 集合，其 outer_shell 不应当是有向的子类型。

WR5：如果在该各 items 集合中包含一个 brep_with_voids，那么该 voids 集合中的每个壳应当是与其 closed_shell_element 方向相反的一个 oriented_closed_shell。

WR6：如果在该各 items 集合中包含一个 mapped_item，那么正被映射的表示类型应当是一个 advanced_brep_shape_representation。

5.2.3.1.9 机械相关环境

mechanical_context（机械相关环境）是一个适用于那些产品是机械产品的 product_context。

注：该实体的使用定义了一个产品的相关环境的一个视点。它不企图作为一个产品类型的分类或目录。被机械使用的这个 product_context 实体的定义规定了该产品在其数据交换中的作用方式。

例 23：对于一块印刷电路板，如果感兴趣的只是它的物理特性如形状，也可以在机械相关环境内定义，并可规定其在电路板中的安装。该印刷电路板的其他特性如连接需求和功能需求不在机械相关环境内规定。

EXPRESS 描述：

```
*)
ENTITY mechanical_context
  SUBTYPE OF(product_context);
WHERE
  WR1:SELF.discipline_type='mechanical';
END_ENTITY;
```

(*

形式限制：

WR1：mechanical_context 实体的 discipline_type 应当包含值'机械的(mechanical)'。

5.2.3.1.10 设计相关环境

design_context(设计相关环境)是一个 product_definition_context，它定义了设计的一个生命周期阶段，作为各 product_definition 的引用框架。

注：设计相关环境内的进一步改进可以用 application_context_element(应用相关环境元素)实体来做。

EXPRESS 描述：

*)

```
ENTITY design_context
    SUBTYPE OF(product_definition_context);
WHERE
    WR1:SELF.life_cycle_stage='design';
END_ENTITY;
```

(*

形式限制：

WR1：design_contex 实体的 life_cycle_stage 应包含值'设计(design)'。

5.2.3.1.11 设计参照关系

一个 design_make_from_relationship(设计参照关系)规定一个 product 的设计是从另一个 product 的设计派生出来的。该 design_make_from_relationship 的使用还意味着由 relating_product_definition 的制造产生的有形产品应用作该 related_product_definition 的制造基础。

EXPRESS 描述：

*)

```
ENTITY design_make_from_relationship
    SUBTYPE OF(product_definition_relationship);
END_ENTITY;
```

(*

属性定义：

SELF_definition_relation.relating_product_definition(自定义关系。关联产品定义)：

该 product_definition 是参照关系的源。

SELF_definition_relation.related_product_definition(自定义关系。有关的产品定义)：

该 product_definition 实体是从 product_definition 源创建的。

相关全局规则：

本标准定义的下列全局规则适用于 design_make_from_relationship 实体：

——no_shape_for_make_from(见 5.2.4.76)。

5.2.3.1.12 供应零件关系

一个 supplied_part_relationship(供应零件关系)叙述了两个 product 的标识，并规定一个标识由设计组织内部使用，另一个标识由供应商使用。

EXPRESS 描述：

*)

```
ENTITY supplied_part_relationship
    SUBTYPE OF(product_definition_relationship);
END_ENTITY;
```

(*

属性定义：

SELF_definition_relation.relating_product_definition：

该 product_definition 是该 product 的设计组织的定义和标识。

SELF_definition_relation.related_product_definition：

该 product_definition 是某组织的产品的定义和标识，该组织给设计组织提供该零件或设计。

非形式限制：

IP1：设计组织和提供组织定义的形状应当在形式、配合和功能上等价。

相关全局规则：

本标准定义的下列全局规则适用于 supplied_part_relationship 实体：

—— no_shape_for_supplied_part(见 5.2.4.77)。

5.2.3.1.13 更改请求

change_request(更改请求)是对某一条产品数据修改愿望的正式通知。

EXPRESS 描述：

*)

```
ENTITY change_request
    SUBTYPE OF(action_request_assignment);
    items:SET[1:?]OF change_request_item;
END_ENTITY;
```

(*

属性定义：

items：标识要被更改的特定的 product 版本的各 change_request_items 的一个集合。

相关全局规则：

在本标准中定义的下列全局规则适用于 change_request 实体：

—— change_request_requires_approval(见 5.2.4.6)；

—— change_request_requires_person_organization(见 5.2.4.7)；

—— change_request_requires_date_time(见 5.2.4.8)。

5.2.3.1.14 开始请求

一个 start_request(开始请求)是要开始新设计一个产品或一组产品的愿望正式通知。

EXPRESS 描述：

*)

```
ENTITY start_request
    SUBTYPE OF(action_request_assignment);
    items:SET[1:?]OF start_request_item;
END_ENTITY;
```

(*

属性定义：

items：标识特定 product 版本的 start_request_item 的一个集合。

相关全局规则：

在本标准中定义的下列全局规则适用于 start_request 实体：

—— start_request_requires_approval(见 5.2.4.11)；

—— start_request_requires_person_organization(见 5.2.4.12)；

——start_request_requires_date_time(见 5.2.4.13)。

5.2.3.1.15 更改

一个 change(更改)标识已被结合到设计中的 change_request,其结果是建立 product 的一个新版本。

EXPRESS 描述:

```
*)
ENTITY change
    SUBTYPE OF(action_assignment);
    items:SET[1:?]OF work_item;
END_ENTITY;
(*
```

属性定义:

items:标识特定各 product 的各版本的各 work_item 的一个集合,其作为综合该 change 创建的结果。

相关全局规则:

在本标准中定义的下列全局规则适用于 change 实体:

——change_requires_approval(见 5.2.4.9);

——change_requires_date_time(见 5.2.4.10);

——unique_version_change_order_rule(见 5.2.4.19)。

5.2.3.1.16 开始工作

一个 start_work(开始工作)标识已被完成的 start_request,其结果是建立一个 product 的新版本。

EXPRESS 描述:

```
*)
ENTITY start_work
    SUBTYPE OF(action_assignment);
    items:SET[1:?]OF work_item;
END_ENTITY;
(*
```

属性定义:

items:标识各 product 的各版本的各 work_item 的一个集合,其作为该 work 开始创建的结果。

相关全局规则:

在本标准中定义的下列全局规则适用于 start_work 实体:

——start_work_requires_approval(见 5.2.4.14);

——start_work_requires_date_time(见 5.2.4.15)。

5.2.3.1.17 配置控制设计认证

一个 cc_design_certification(配置控制设计认证)把一个 certification 赋值给一个外部组织提供的 product。

EXPRESS 描述:

```
*)
ENTITY cc_design_certification
  SUBTYPE OF(certification_assignment);
  items:SET[1:?]OF certified_item;
END_ENTITY;
```

(*

属性定义：

item：各 certified_item(认证项)的一个集合标识了正被赋值给它 certification 的 supplied_part_relationship 实体。

5.2.3.1.18 配置控制设计批准

一个 cc_design_approval(配置控制设计批准)将一个 approval 与产品的一个特定数据建立联系。

EXPRESS 描述：

*)

```
ENTITY cc_design_approval
  SUBTYPE OF(approval_assignment);
  items:SET[1:?]OF approved_item;
END_ENTITY;
```

(*

属性定义：

items：各 approved_item 的一个集合，其标识了 approval 应用的特定 product_definition_formation，product_definition，planned_effectivity，configuration_item，security_classification，change_request，change，start_request，start_work，certification 或 contract。

相关全局规则：

在本标准中定义的下列全局规则适用于 cc_design_approval 实体：

——change_request_requires_approval(见 5.2.4.6)；

——change_requires_approval(见 5.2.4.9)；

——start_request_requires_approval(见 5.2.4.11)；

——start_work_requires_approval(见 5.2.4.14)；

——product_version_requires_approval(见 5.2.4.22)；

——product_definition_requires_approval(见 5.2.4.26)；

——certification_requires_approval(见 5.2.4.28)；

——contract_requires_approval(见 5.2.4.35)；

——security_classification_requires_approval(见 5.2.4.38)；

——effectivity_requires_approval(见 5.2.4.66)；

——configuration_item_requires_approval(见 5.2.4.67)。

5.2.3.1.19 配置控制设计合同

一个 cc_design_contract(配置控制设计合同)将一个 contract 与一个 product_definition_formation 建立联系。

EXPRESS 描述：

*)

```
ENTITY cc_design_contract
    SUBTYPE OF(contract_assignment);
    items:SET[1:?]OF contracted_item;
END_ENTITY;
```

(*

属性定义：

items：各 contracted_item(合同项)的一个集合，其标识了各特定产品的各版本。

5.2.3.1.20 配置控制设计安全分类

cc_design_security_classification(配置控制设计安全分类)将一个 security_classification 与产品的一条特定数据建立联系。

EXPRESS 描述:

```
*)
ENTITYcc_design_security_classification
    SUBTYPE OF (security_classification_assignment);
    items:SET[1:?]OF classified_item;
END_ENTITY;
(*
```

属性定义:

items:各 classified_item(分类项)的一个集合,它标识了其 security_classification 已赋值的各特定产品的各版本或各定义关系。

相关全局规则:

在本标准中定义的下列全局规则适用于 cc_design_security_classification 实体:

——product_version_requires_security_classification(见 5.2.4.24);

——acu_requires_security_classification(见 5.2.4.70)。

5.2.3.1.21 配置控制设计人员和组织赋值

一个 cc_design_person_and_organization_assignment(配置控制设计人员和组织赋值)将一个 person_and_organization 与产品的一条特定数据建立联系。

EXPRESS 描述:

```
*)
ENTITY cc_design_person_and_organization_assignment
    SUBTYPE OF (person_and_organization_assignment);
    items:SET[1:?]OF person_organization_item;
WHERE
    WR1:cc_design_person_and_organization_correlation(SELF);
END_ENTITY;
(*
```

属性定义:

items:各 person_organization_item(人员组织项)的一个集合,其标识了要与 person_and_organization 建立联系的 change_request,start_request,configuration_item,product,product_definition_formation,contract 或 security_classification.

形式限制:

WR1:应满足该 cc_design_person_and_organization_correlation 功能,此功能把人员和组织职能与产品数据的各元素相联。该 cc_design_person_and_organization_correlation 功能的完整定义见 5.2.5.3 条。

相关全局规则:

在本标准中定义的下列全局规则适用于 cc_design_person_and_organization_assignment 实体:

——change_request_requires_person_organization(见 5.2.4.7);

——start_request_requires_person_organization(见 5.2.4.12);

——product_requires_person_organization(见 5.2.4.21);

——product_version_requires_person_organization(见 5.2.4.23);

——product_definition_requires_person_organization(见 5.2.4.25);

——contract_requires_person_organization(见 5.2.4.36)

——security_classification_requires_person_organization(见 5.2.4.39);

——configuration_item_requires_person_organization(见 5.2.4.64)。

5.2.3.1.22 配置控制设计日期和时间赋值

cc_design_date_and_time_assignment(配置控制设计日期和时间赋值)将一个 date_and_time_assignment 与产品的一个特定数据建立联系。

EXPRESS 描述:

```
*)
ENTITY cc_design_date_and_time_assignment
    SUBTYPE OF (date_and_time_assignment);
    items:SET[1:?] OF date_time_item;
WHERE
    WR1:cc_design_date_time_correlation(SELF);
END_ENTITY;
(*
```

属性定义:

items:各 date_time_item(日期时间项)的一个集合,其标识了日期时间要被赋值的特定 product_definition, change_request, start_request, change, start_work, contract, security_classification, change, certification 或 approval_person_and_organization.

形式限制:

WR1:此 cc_design_date_time_correlation 功能应满足日期和时间的职能与产品数据元素相联。有关 cc_design_date_time_correlation 功能的完整定义见 5.2.5.4 条。

相关全局规则:

在本标准定义的下列全局规则适用于 cc_design_date_and_time_assignment 实体:

change_request_requires_date_time(见 5.2.4.8);

change_requires_date_time(见 5.2.4.10);

start_request_requires_date_time(见 5.2.4.13);

start_work_requires_date_time(见 5.2.4.15);

product_definition_requires_date_time(见 5.2.4.27);

certification_requires_date_time(见 5.2.4.30);

security_classification_requires_date_time(见 5.2.4.40);

security_classification_optional_date_time(见 5.2.4.41)。

5.2.3.1.23 配置控制设计规范引用

一个 cc_design_specification_reference(配置控制设计规范引用)将一个规范的一个引用与一个产品的一条完整定义或与它形状的某个外形建立联系。

EXPRESS 描述:

```
*)
ENTITY cc_design_specification_reference
    SUBTYPE OF (document_reference);
    items:SET[1:?]OF specified_item;
END_ENTITY;
(*
```

属性定义:

items：该 specification_reference(规范引用)应用的各 product_definition 和/或各 shape_aspect 的一个集合。

相关全局规则：

在本标准中定义的下列全局规则适用于 cc_design_specification_reference 实体：

——document_to_product_definition(见 5.2.4.46)。

5.2.3.2 配置控制设计输入的实体修改

5.2.3.2.1 应用相关环境

此 application_context(应用相关环境)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 application_context 实体：

——application_context_requires_ap_definition(见 5.2.4.1)。

5.2.3.2.2 应用协议定义

此 application_protocol_definition(应用协议定义)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 application_protocol_definition 实体：

——application_context_requires_ap_definition(见 5.2.4.1)。

5.2.3.2.3 产品相关环境

此 product_context(产品相关环境)实体的基本定义在 GB/T 16656.41 中给出。下面的修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 product_context 实体：

——subtype_mandatory_product_context(见 5.2.4.2)。

5.2.3.2.4 产品定义相关环境

此 product_definition_context(产品定义相关环境)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

product_definition_context 的定义修正如下：

为了规定该 product_definition 应用设计中的准确相关环境，应该使用 application_context_element(应用相关环境元素)的属性 name(名)去修改 product_definition_context 实体。

5.2.3.2.5 产品目录

此 product_category(产品目录)实体的基本定义在 GB/T 16656.41 中给出。下列修改应用到本标准。

product_category 的定义修改如下：

在本标准中，利用各 product_category 树去归类各零件。在该树中的一个中间节点可以用于分离标准件和非标准件。在这种情况下，此 product_category 的属性 name 应当是“标准零件(standard_part)”。

5.2.3.2.6 有关产品的产品目录

此 product_related_product_category(有关产品的产品目录)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 product_related_product_category 实体：

——restrict_product_category_value(见 5.2.4.4)；

——product_requires_product_category(见 5.2.4.5)。

5.2.3.2.7 产品

此 product(产品)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 product 实体：

product_requires_product_category(见 5.2.4.5)；

product_requires_version(见 5.2.4.20)；

product_requires_person_organization(见 5.2.4.21)。

5.2.3.2.8 产品定义形成

此 product_definition formation(产品定义形成)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 product_definition_formation 实体：

product_requires_version(见 5.2.4.20)；

product_version_requires_approval(见 5.2.4.22)；

product_version_requires_person_organization(见 5.2.4.23)；

product_version_requires_security_classification(见 5.2.4.24)；

subtype_mandatory_product_definition_formation(见 5.2.4.50)。

5.2.3.2.9 产品定义

此 product_definition(产品定义)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 product_definition 实体：

design_context_for_property(见 5.2.4.3)；

product_definition_requires_person_organization(见 5.2.4.25)；

product_definition_requires_approval(见 5.2.4.26)；

product_definition_requires_date_time(见 5.2.4.27)；

document_to_product_definition(见 5.2.4.46)。

5.2.3.2.10 活动请求状态

此 action_request_status(活动请求状态)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 action_request_status 实体：

restrict_action_request_status(见 5.2.4.16)；

versioned_action_request_requires_status(见 5.2.4.17)。

5.2.3.2.11 该版活动请求

此 versioned_action_request(该版活动请求)实体的基本定义在 GB/T 16656.41 中给出。下面的修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 versioned_action_request 实体：

versioned_action_request_requires_status(见 5.2.4.17)；

versioned_action_request_requires_solution(见 5.2.4.18)。

5.2.3.2.12 活动请求方案

此 action_request_solution(活动请求方案)实体的基本定义在 GB/T 16656.41 中给出。下面的修改

适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 versioned_action_request 实体：

—— versioned_action_request_requires_solution(见 5.2.4.18)。

5.2.3.2.13 认证类型

此 certification_type(认证类型)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 certification_type 实体：

—— restrict certification_type(见 5.2.4.29)；

—— dependent instantiable_certification_type(见 5.2.4.62)。

5.2.3.2.14 认证

此 certification(认证)实体的基本定义在 GB/T 16656.41 中给出。下列修改应用到本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 certification 实体：

—— certification_requires_approval(见 5.2.4.28)；

—— certification_requires_date_time(见 5.2.4.30)。

5.2.3.2.15 批准状态

此 approval_status(批准状态)实体的基本定义在 GB/T 16656.41 中给出。下列修改应用到本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 approval_status 实体：

—— restrict_approval_status(见 5.2.4.34)；

—— dependent_instantiable approval_status(见 5.2.4.59)。

5.2.3.2.16 批准

此 approval(批准)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 approval 实体：

—— approval_are assigned(见 5.2.4.31)；

—— approval_requires_approval_person_organization(见 5.2.4.32)；

—— approval_requires_approval_date_time(见 5.2.4.33)。

5.2.3.2.17 批准赋值

此 approval_assignment(批准赋值)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 approval_assignment 实体：

—— approval_are assigned(见 5.2.4.31)。

5.2.3.2.18 批准人员组织

此 approval_person_organization(批准人员组织)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 approval_person_organization 实体：

—— approval_requires_approval_person_organization(见 5.2.4.32)。

5.2.3.2.19 批准日期时间

此 approval_date_time(批准日期时间)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 approval_date_time 实体：

approval requires_approval_date_time(见 5.2.4.33)。

5.2.3.2.20 合同类型

此 contract_type(合同类型)实体的基本定义在 GB/T 16656.41 中给出。下列修改应用到本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 contract_type 实体：

restrict_contract_type(见 5.2.4.37)；

dependent_instantiable_contract_type(见 5.2.4.61)。

5.2.3.2.21 合同

此 contract(合同)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 contract 实体：

contract_requires_approval(见 5.2.4.35)；

contract_requires_person_organization(见 5.2.4.36)。

5.2.3.2.22 安全分类等级

此 security_classification_level(安全分类等级)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 security_classification_level 实体：

restrict_security_classification_level(见 5.2.4.42)；

dependent_instantiable_security_classification_level(见 5.2.4.58)。

5.2.3.2.23 安全分类

此 security_classification(安全分类)实体的基本定义在 GB/T 16656.41 中给出。下面的修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 security_classification 实体：

security_classification_requires_approval(见 5.2.4.38)；

security_classification_requires_person_organization(见 5.2.4.39)；

security_classification_requires_date_time(见 5.2.4.40)；

security_classification_optional_date_time(见 5.2.4.41)。

5.2.3.2.24 人员和组织的职能

此 person_and_organization_role(人员和组织的职能)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 person_and_organization_role 实体：

restrict_person_organization_role(见 5.2.4.43)；

dependent_instantiable_person_and_organization_role(见 5.2.4.56)。

5.2.3.2.25 人员

此 person(人员)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

属性定义：

此属性各定义修改如下：

id：标识 person 的一种手段。

注：必须在该数据的整个作用域之上维持和保证此属性的唯一性。对于单个组织，国家或集体，该限制容易满足。在全局或全世界范围内要求唯一性的情况下，读者可以引用 ISO/IEC 8824-1——有关国际应用唯一性字符串注册指南。

5.2.3.2.26 日期时间职能

此 date_time_role(日期时间职能)实体的基本定义在 GB/T 16656.41 中给出。下列修改应用到本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 date_time_role 实体：

——restrict_date_time_role(见 5.2.4.44)；

——dependent_instantiable_date_time_role(见 5.2.4.55)。

5.2.3.2.27 文档类型

此 document_type(文档类型)实体的基本定义在 GB/T 16656.41 中给出。下列修改应用到本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 document_type 实体：

——restrict_document_type(见 5.2.4.45)；

——dependent_instantiable_document_type(见 5.2.4.60)。

5.2.3.2.28 度量

此 measure_with_unit(度量)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 measure_with_unit 实体：

——as_requires_quantity(见 5.2.4.47)。

5.2.3.2.29 全局单位赋值相关环境

此 global_unit_assigned_context(全局单位赋值相关环境)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 global_unit_assigned_context 实体：

——global_unit_assignment(见 5.2.4.48)。

5.2.3.2.30 活动

此 action(活动)实体的基本定义在 GB/T16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 action 实体：

——subtype_mandatory_action(见 5.2.4.49)。

5.2.3.2.31 活动命令

此 action_directive(活动命令)实体的基本定义在 GB/T 16656.41 中给出。下列修改应用到本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 action_directive 实体：

——dependent_instantiable_action_directive(见 5.2.4.57)。

5.2.3.2.32 日期

此 date(日期)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 date 实体：

dependent_instantiable_date(见 5.2.4.51)。

5.2.3.2.33 表示

此 representation(表示)实体的基本定义在 GB/T 16656.41 中给出。下列修改应用到本标准。

相关全局规则:

在本标准中定义的下列全局规则适用于 representation 实体:

——subtype_mandatory_representation(见 5.2.4.74)。

5.2.3.2.34 表示相关环境

此 representation_context(表示相关环境)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则:

在本标准中定义的下列全局规则适用于 representation_context 实体:

——subtype_mandatory_representation_context(见 5.2.4.75)。

5.2.3.2.35 形状表示

此 shape_representation(形状表示)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则:

在本标准中定义的下列全局规则适用于 shape_representation 实体:

dependent_instantiable_shape_representation(见 5.2.4.52);

——subtype_mandatory_shape_representation(见 5.2.4.73)。

5.2.3.2.36 相关环境依赖的形状表示

此 context_dependent_shape_representation(相关环境依赖的形状表示)实体的基本定义在 GB/T 16656.41中给出。下列修改适用于本标准。

在本标准中,应使用 context_dependent_shape_representation 实体在定义一个装配件中一构件的使用的 next_assembly_usage_occurrence 与该 shape_representation_relationship 间建立联系。其 shape_representation_relationship 定义了该装配件的形状定义中的构件形状的位置和有向。

注:可以用 context_dependent_shape_representation 在各装配件和各构件的各形状之间建立联系,而不管它们使用什么样的 shape_representation。也可以用 mapped_item 结构在各构件和各装配件的形状之间建立联系,可是在本标准中使用此 mapped_item 要求该 shape_representation 的各类型相同。

5.2.3.2.37 命名单位

此 named_unit(命名单位)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则:

在本标准中定义的下列全局规则适用于 named_unit 实体:

dependent_instantiable_named_unit(见 5.2.4.53)。

5.2.3.2.38 表示项

此 representation_item(表示项)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则:

在本标准中定义的下列全局规则适用于 representation_item 实体:

dependent_instantiable_representation_item(见 5.2.4.54)。

5.2.3.2.39 产品定义使用

此 product_definition_usage(产品定义使用)实体的基本定义在 GB/T 16656.41 中给出。下面的修改适用于本标准。

相关全局规则:

在本标准中定义的下列全局规则适用于 product_definition_usage 实体：

——acu_requires_security_classification(见 5.2.4.70)；

——subtype_mandatory_product_definition_usage(见 5.2.4.69)。

5.2.3.2.40 几何表示项

此 geometric_representation_item(几何表示项)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 geometric_representation_item 实体：

—— geometric_representation_item_3d(见 5.2.4.71)。

5.2.3.2.41 参数表示相关环境

此 parametric_representation_context(参数表示相关环境)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 parametric_representation_context 实体：

——dependent_instantiable_parametric_representation_context(见 5.2.4.72)。

5.2.3.2.42 产品概念

此 product_concept(产品概念)实体的基本定义在 GB/T 16656.41 中给出。下列修改应用到本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 product_concept 实体：

—— product_concept_requires_configuration_item(见 5.2.4.63)。

5.2.3.2.43 配置项

此 configuration_item(配置项)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 configuration_item 实体：

—— product_concept_requires_configuration_item(见 5.2.4.63)；

—— configuration_item_requires_person_organization(见 5.2.4.64)；

—— configuration_item_requires_approval(见 5.2.4.67)。

5.2.3.2.44 有效性

此 effectivity(有效性)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 effectivity 实体：

—— subtype_mandatory_effectivity(见 5.2.4.65)；

—— effectivity_requires_approval(见 5.2.4.66)。

5.2.3.2.45 下一层装配活动

此 next_assembly_usage_occurrence(下一层装配活动)实体的基本定义在 GB/T 16656.41 中给出。下列修改适用于本标准。

相关全局规则：

在本标准中定义的下列全局规则适用于 next_assembly_usage_occurrence 实体：

—— coordinated_assembly_and_shape(见 5.2.4.68)。

5.2.4 配置控制设计规则

5.2.4.1 应用相关环境需要应用协议定义

此 application_context_requires_ap_definition(应用相关环境需要应用协议定义)规则规定 application_context 的每个实例应确实由规定本标准的一个 application_protocol_definition 实体所引用。

EXPRESS 描述:

```
*)
RULE application_context_requires_ap_definition FOR
    (application_context,application_protocol_definition);
WHERE
    WR1:SIZEOF(QUERY(ac<*application_context|
        NOT(SIZEOF(QUERY(apd<*application_protocol_definition|
        (ac:=:apd.application)
        AND
    (apd.application_interpreted_model_schema_name=
        'config_control_design')))=1)))=0;
END_RULE;--application_context_requires_ap_definition
(*
```

变元定义:

application_context:标识 application_context 各实体所有实例的集合。

application_protocol_definition:标识 application_protocol_definition 各实体所有实例的集合。

形式限制:

WR1:对 application_context 的每个实例,应确实有一个 application_protocol_definition 实例引用 application_context 的实例作为它的 application,该 application 具有作为它的 application_interpreted_model_schema_name 的值'配置控制设计(config_control_design)'。

5.2.4.2 子类型强制的产品相关环境

此 subtype_mandatory_product_context(子类型强制的产品相关环境)规则规定允许使用的产品相关环境应该用 mechanical_context 实体来定义 product_context 的 discipline_type。这种用法保证交流的产品数据相关环境代表了数据的一个机械视点。

EXPRESS 描述:

```
*)
RULE subtype_mandatory_product_context FOR(product_context);
WHERE
    WR1:SIZEOF(QUERY(pc<*product_context|
        NOT ('CONFIG_CONTROL_DESIGN.MECHANICAL_CONTEXT' IN TYPEOF(pc))))
        =0;
END_RULE;
(*
```

变元定义:

product_context:标识 product_context 实体所有实例的集合。

形式限制:

WR1:product_context 的每个实例应当是一个 mechanical_context 实体。

5.2.4.3 性能设计相关环境

此 design_context_for_property(性能设计相关环境)规则规定该唯一生命周期阶段是指设计阶段,其目的可在本标准中定义。各产品结构可以使用 product_definition 相关环境的任何生命周期阶段来定

义。

EXPRESS 描述：

```
*)
RULE design_context_for_property FOR (product_definition);
WHERE
    WR1:SIZEOF(QUERY(pd<*product_definition|
        (SIZEOF(USEDIN(pd,'CONFIG_CONTROL_DESIGN.'+
        'PROPERTY_DEFINITION.DEFINITION')+
        QUERY (pdr<*USEDIN(pd,'CONFIG_CONTROL_DESIGN.'+
        'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')|
        SIZEOF(USEDIN(pdr,'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION.'+
        'DEFINITION'))>=1))>=1)AND
        (NOT('CONFIG_CONTROL_DESIGN.DESIGN_CONTEXT'IN
        TYPEOF(pd.frame_of_reference)))))=0;
END_RULE;
(*
```

变元定义：

product_definition：标识 product_definition 所有实例的集合。

形式限制：

WR1：对 product_definition 的每个实例，如果该实例作为 property_definition 的该 definition 属性被引用，或者如果该实例是一个 product_definition_relationship 实例中的 related_product_definition，并且 product_definition_relationship 实例由一个 property_definition 的属性 definition 所引用，那么对于那个 product_definition 实例的 frame_of_reference 属性必须引用一个 design_context。

5.2.4.4 限制产品目录值

此 restrict_product_category_value(限制产品目录值)规则规定了与一个 product 有关的 product_category 可以包含的各值的集合。

EXPRESS 描述：

```
*)
RULE restrict_product_category_value FOR
(product_related_product_category);
WHERE
    WR1:SIZEOF(QUERY(prpc<*
product_related_product_category|
        NOT(prpc.name IN ['assembly','detail',
        'customer_furnished_equipment','inseparable_assembly','cast',
        'coined','drawn','extruded','forged','formed','machined,'
        'molded','rolled','sheared']))
        =0;
END_RULE;
(*
```

变元定义：

product_related_product_category：标识 product_related_product_category 各实体所有实例的集合。

形式限制：

WR1：一个 product_related_product_category 的 name 属性应当是“装配(assembly)”,“细目(detail)”,“不可拆装配(inseparable_assembly)”,“配备设备的用户(customer_furnished_equipment)”,“铸造(cast)”,“高压铸造(coined)”,“拉伸(drawn)”,“挤压(extruded)”,“锻造(forged)”,“锻压(formed)”,“机加工(machined)”,“模塑(molded)”,“轧制(rolled)”,或应是“剪切(sheared)”。

属性值限制定义：

assembly：标识一个零件，该零件由满足某个特定函数而组装在一起的其他零件的集合所组成。

detail：标识一个零件，该零件位于材料结构单的底层。

customer_furnished_equipment：标识一个零件，该零件已被一个客户提交给设计部门。

inseparable_assembly：标识一个零件，该零件装配后不能被拆开，否则该装配件中的至少一个构件要受到物理损伤。

cast：标识一类要通过机械加工工艺生产的零件，在该工艺中，材料以液态形式倒入模具中并硬化。

coined：标识一类要通过机械加工工艺生产的零件，在该工艺中，材料以液态形式在高压下强制进入已制定好的模具中并强制材料填充模具的形状。

drawn：标识一类要通过机械加工工艺生产的零件，在该工艺中，金属被拉伸变形，从而形成定义的形状。

extruded：标识一类要通过机械加工工艺生产的零件，在该工艺中，材料经过挤压而改变原毛坯的形状。

forged：标识一类要通过机械加工工艺生产的零件，在该工艺中，金属材料在高温及外力的作用下被加工成有用的形状。

formed：标识一类要通过机械加工工艺生产的零件，在该工艺中，金属材料经重压改变其形状。

machined：标识一类要通过机械加工工艺生产的零件，该工艺使用一个自动设备。

molded：标识一类要通过机械加工工艺生产的零件，在该工艺中，材料被加热超过其熔点并被倒入模具中形成其形状。

rolled：标识一类要通过机械加工工艺生产的零件，在该工艺中，材料在两轧辊之间通过以减少其厚度，使其等于两轧辊间的距离。

sheared：标识一类要通过机械加工工艺生产的零件，在该工艺中，材料经切割改变形状。

5.2.4.5 产品需要产品目录

此 product_requires_product_category(产品需要产品目录)规则规定了每个 product 应确实由一个 product_related_product_category 来引用的要求，其定义了该产品是否是一个“assembly”,“inseparable_assembly”,“detail”或是“customer_furnished_equipment”。

注：product_requires_product_category 规则限制了 product_related_product_category 与 product 之间的关系。该规则规定一个 product 必须由其值为“assembly”,“inseparable_assembly”,“detail”或“customer_furnished_equipment”的一个确实的 product_related_product_category 来引用。该规则不排除引入其他 product_related_product_category 实体的实例，其值不是上面所规定的 product_related_product_category 的 name 属性的那些值。该 product_related_product_category 的属性 name 的各有效值在 restrict_product_category_value 规则中定义。

EXPRESS 描述：

```
*)
RULE product_requires_product_category FOR
    (product,product_related_product_category);
WHERE
    WR1:SIZEOF(QUERY(prod<*product|
        NOT (SIZEOF(QUERY(prpc<*product_related_product_category|
        (prod IN prpc.products)AND
```

```
        (prpc.name IN ['assembly','inseparable_assembly','detail',
        'customer_furnished_equipment']))=1)))=0;
END_RULE;
(*
```

变元定义：

product：标识 product 实体所有实例的集合。

product_related_product_category：标识 product_related_product_category 各实体的所有实例的集合。

形式限制：

WR1：对于 product 的每个实例，应确实有包含 name 属性的 product_related_product_category 的一个实例。其属性 name 的值是“assembly”，“inseparable_assembly”，“detail”或是“customer_furnished_equipment”。

5.2.4.6 更改请求需要批准

此 change_request_requires_approval(更改请求需要批准)规则规定 change_request 的每个实例应由一确实的 cc_design_approval 所引用。这个规则强调每个更改请求需要有一个批准。

EXPRESS 描述：

```
*)
RULE change_request_requires_approval FOR
    (change_request,cc_design_approval);
WHERE
  WR1:SIZEOF(QUERY(cr<*change_request|
        NOT(SIZEOF(QUERY(ccda<*cc_design_approval|
        cr IN ccda.items))=1)))=0;
END_RULE;
(*
```

变元定义：

change_request：标识 change_request 各实体所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体所有实例的集合。

形式限制：

WR1：对于 change_request 的每个实例，应至少存在 cc_design_approval 的一个实例，在其 items 的集合中包括 change_request 的实例。

5.2.4.7 更改请求需要人员组织

change_request_requires_person_organization(更改请求需要人员组织)规则规定每个 change_request 应至少由一个 cc_design_person_and_organization_assignment 所引用。这规则规定每个 change_request 必须有一个接受者接受该请求。接受者的含义可在 person_and_organization_assignment 的 role 属性中找到。

注：在该 cc_design_person_and_organization_correlation 函数中规定对于不同实体该不同职能值应与 person_and_organization_assignment 的赋值相一致。cc_design_person_and_organization_assignment 局部调用这个函数。

EXPRESS 描述：

```
*)
RULE change_request_requires_person_organization FOR
    (change_request,
    cc_design_person_and_organization_assignment);
```

```
WHERE
    WR1:SIZEOF(QUERY(cr<*change_request|
        NOT(SIZEOF(QUERY(ccpoa<*
        cc_design_person_and_organization_assignment|
        cr IN ccpoa.items))>=1)))=0;
END_RULE
(*
```

变元定义：

change_request：标识 change_request 实体所有实例的集合。

cc_design_person_and_organization_assignment：标识 cc_design_person_and_organization_assignment 各实体所有实例的集合。

形式限制：

WR1：对于 change_request 的每个实例，将至少有一个 cc_design_person_and_organization_assignment 的实例，在其 items 的集合中包括 change_request 实例。

5.2.4.8 更改请求需要日期时间

此 change_request_requires_date_time(更改请求需要日期时间)规则规定 change_request 的每个实例应确实由一个 cc_design_date_time_assignment 实体所引用。该规则强调每个 change_request 要求有一个请求日期，它标识 change_request 发布的日期。

注：在该 cc_design_date_and_time_assignment 函数中规定对于不同的实体该不同职能值应与 date_time_assignment 的赋值相一致。对于 cc_design_date_and_time_correlation 局部调用这个函数。

EXPRESS 描述：

```
*)
RULE change_request_requires_date_time FOR
    (change_request,cc_design_date_and_time_assignment);
WHERE
    WR1: SIZFOF(QUERY(cr<*change_request|
        NOT(SIZEOF(QUERY(ccdta<*cc_design_date_and_time_assignment|
        cr IN ccdta.items))=1)))=0;
END_RULE
(*
```

变元定义：

change_request：标识 change_request 各实体所有实例的集合。

cc_design_date_and_time_assignment：标识 cc_design_date_and_time_assignment 各实体所有实例的集合。

形式限制：

WR1：对于 change_request 的每个实例，应确实有 cc_design_date_and_time_assignment 的一个实例，在其 items 的集合中包括 change_request 的实例。

5.2.4.9 更改需要批准

change_requires_approval(更改需要批准)规则规定每个 change 实例应确实由 cc_design_approval 的一个实例所引用。该规则强调每个更改要有一个批准。

EXPRESS 描述：

```
*)
RULE change_requires_approval FOR
```

```
    (change,cc_design_approval);
WHERE
    WR1:SIZEOF(QUERY(chg<*change|
        NOT(SIZEOF(QUERY(ccda<*cc_design_approval|
        chg IN ccda.items))=1)))=0;
END_RULE;
(*
```

变元定义：

change：标识 change 各实体所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体所有实例的集合。

形式限制：

WR1：对于 change 的每个实例，应确实有一个 cc_design_approval 实例，在其 items 的集合中包括 change 的实例。

5.2.4.10 更改需要日期时间

change_requires_date_time(更改需要日期时间)规则规定 change 的每个实例应确实由一个 cc_design_date_and_time_assignment 所引用。该规则强调每个 change 要有一个更改日期，它标识 change 被结合进设计的日期。

注：在该 cc_design_date_and_time_correlation 函数中规定对于不同的实体该不同职能值应与 date_time_assignment 的赋值相一致。对于 cc_design_date_and_time_assignment，局部调用这个函数。

EXPRESS 描述：

```
*)
RULE change_requires_date_time FOR
    (change,cc_design_date_and_time_assignment);
WHERE
    WR1:SIZEOF(QUERY(chg<*change|
        NOT(SIZEOF(QUERY(ccdta<*cc_design_date_and_time_assignment|
        (chg IN ccdta.items)
        AND (ccdta.role.name='start_date')))=1)))=0;
END_RULE;
(*
```

变元定义：

change：标识 change 各实体所有实例的集合。

cc_design_date_and_time_assignment：标识 cc_design_date_and_time_assignment 各实体所有实例的集合。

形式限制：

WR1：对于 change 的每个实例，应确实有 cc_design_date_and_time_assignment 的一个实例，在其 items 的集合中包括 change 实例，其职能名是“开始日期(start_date)”。

5.2.4.11 开始请求需要批准

此 start_request_requires_approval(开始请求需要批准)规则规定的每个 start_request 实例应确实由 cc_design_approval 的一个实例所引用。该规则强调对于每个 start_request 的要求要有一个批准来授权一个设计的开始。

EXPRESS 描述：

```
*)
```

```
RULE start_request_requires_approval FOR
    (start_request,cc_design_approval);
WHERE
    WR1:SIZEOF(QUERY(sr<*start_request|
        NOT(SIZEOF(QUERY(ccda<*cc_design_approval|
        sr IN ccda.items))=1)))=0;
END_RULE;
```

(

变元定义：

start_request：标识 start_request 各实体所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体所有实例的集合。

形式限制：

WR1：对于 start_request 的每个实例，应确实有 cc_design_approval 的一个实例，在其各 items 的集合中包括 start_request 的实例。

5.2.4.12 开始请求需要人员组织

start_request_requires_person_organization(开始请求需要人员组织)规则规定每个 start_request 应至少由一个(cc_design_person_and_organization_assignment)实体所引用。该规则指出每个 start_request 必须有一个接受者接收该请求。接受者的含义可在 person_and_organization_assignment 实体的 role 中找到。

注：在该 cc_design_person_and_organization_correlation 函数中规定对不同实体该不同职能值应与 person_and_organization_assignment 的赋值相一致。对于 cc_design_person_and_organization_assignment，局部调用这个函数。

EXPRESS 描述：

)

```
RULE start_request_requires_person_organization FOR(start_request,
    cc_design_person_and_organization_assignment);
WHERE
    WR1:SIZEOF(QUERY(sr<*start_request|
        NOT(SIZEOF(QUERY(ccdpoa<*
        cc_design_person_and_organization_assignment|
        sr IN ccdpoa.items))>=1)))=0;
END_RULE;
```

(

变元定义：

start_request：标识 start_request 各实体所有实例的集合。

cc_design_person_and_organization_assignment：标识 cc_design_person_and_organization_assignment 各实体所有实例的集合。

形式限制：

WR1：对于 start_request 的每个实例，应至少有 cc_design_person_and organization_assignment 的一个实例，在其各 items 的集合中包括 start_request 的实例。

5.2.4.13 开始请求需要日期时间

此 start_request_requires_date_time(开始请求需要日期时间)规则规定 start_request 的每个实例应确实由一个 cc_design_date_and_time_assignment 所引用。该规则强调每个 start_request 要求有一个请

求日期指出该 start_request 发布的日期。

注：在该 cc_design_date_and_time_correlation 函数中规定对于不同的实体该不同职能值应与 date_time_assignment 的赋值相一致。对于 cc_design_date_and_time_assignment，局部调用这个函数。

EXPRESS 描述：

```
*)
RULE start_request_requires_date_time FOR
    (start_request,cc_design_date_and_time_assignment);
WHERE
    WR1: SIZEOF(QUERY(sr<*start_request|
        NOT(SIZEOF(QUERY(ccdta<*cc_design_date_and_time_assignment|
        sr IN ccdta.items))=1)))=0;
END_RULE;
(*
```

变元定义：

start_request：标识 start_requestg 各实体所有实例的集合。

cc_design_date_and_time_assignment：标识 cc_design_date_and_time_assignment 各实体所有实例的集合。

形式限制：

WR1：对于 start_request 的每个实例，应确实有 cc_design_date_and_time_assignment 的一个实例，在其各 items 的集合中包括 start_request 的实例。

5.2.4.14 开始工作需要批准

此 start_work_requires_approval(开始工作需要批准)规则规定 start_work 的每个实例应确实由 cc_design_approval 的一个实例所引用。这个规则强调对每个设计起初的要求要有一个批准。

EXPRESS 描述：

```
*)
RULE start_work_requires_approval FOR
    (start_work,cc_design_approval);
WHERE
    WR1:SIZEOF(QUERY(sw<*start_work|
        NOT(SIZEOF(QUERY(ccda<*cc_design_approval|
        sw IN ccda.items))=1)))=0;
END_RULE;
(*
```

变元定义：

start_work：标识 start_work 各实体所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体所有实例的集合。

形式限制：

WR1：对于 start_work 的每个实例，应确实有 cc_design_approval 的一个实例，在其 items(各项)的集合中包括 start_work 的实例。

5.2.4.15 开始工作需要日期时间

此 start_work_requires_date_time(开始工作需要日期时间)规则规定一个 start_work 的每个实例应确实由一个 cc_design_date_and_time_assignment 所引用。该规则强调对于每个 start_work 的要求要

有一个开始日期指出该设计工作起动的日期。

注：在该 cc_design_date_and_time_correlation 函数中规定对于不同的实体该不同职能值应与 date_time_assignment 的赋值相一致。对于 cc_design_person_and_organization_assignment，局部调用这个函数。

EXPRESS 描述：

```
*)
RULE start_work_requires_date_time FOR
    (start_work,cc_design_date_and_time_assignment);
WHERE
    WR1: SIZEOF(QUERY(sw<*start_work|
      NOT(SIZEOF(QUERY(ccdta<*cc_design_date_and_time_assignment|
      (sw IN ccdta.items)
      AND(ccdta.role.name='start_date')))=1)))=0;
END_RULE;
(*
```

变元定义：

start_work：标识 start_work 各实体所有实例的集合。

cc_design_date_and_time_assignment：标识 cc_design_date_and_time_assignment 各实体所有实例的集合。

形式限制：

WR1：对于 start_work 的每个实例，应确实有 cc_design_date_and_time_assignment 的一个实例，在其各 items 的集合中包括 start_work 的实例，其职能名是“开始日期(start_date)”。

5.2.4.16 限制活动请求状态

此 restrict_action_request_status(限制活动请求状态)规则规定了一个 action_request 的各状态允许值。

EXPRESS 描述：

```
*)
RULE restrict_action_request_status FOR(action_request_status);
WHERE
    WR1:SIZEOF(QUERY(ars<*action_request_status|
        NOT(ars.status IN
        ['proposed','in_work','issued','hold'])))=0;
END_RULE;
(*
```

变元定义：

action_request_status：标识 action_request_status 各实体所有实例的集合。

形式限制：

WR1：对于该 action_request_status 实体的每个实例，该状态属性应当有一个值，其值是“提出(proposed)”，“在工作中(in_work)”，“发布(issued)”，或是“扣留(hold)”。

属性值定义：

proposed：该 versioned_action_request 已完成，正在等待审批和授权。

in_work：该 versioned_action_request 正在开发，可能包含进设计中。

issued：该 versioned_action_request 已授权包含进设计中。

hold：该 versioned_action_request 已审批，并且授权不包含在设计中。

5.2.4.17 该版活动请求需要状态

此 versioned_action_request_requires_status(该版活动请求需要状态)规则规定 versioned_action_request 的每个实例应确实有一个状态。一个 versioned_action_request_requires_status 的状态由该 action_request_status 实体定义。

EXPRESS 描述:

```
*)
RULE versioned_action_request_requires_status FOR
    (versioned_action_request,action_request_status);
WHERE
    WR1:SIZEOF(QUERY(ar<* versioned_action_request|
        NOT(SIZEOF(QUERY(ars<* action_request_status|
        ar:=:ars.assigned_request))=1)))=0;
END_RULE;
(*
    变元定义:
    versioned_action_request:标识 versioned_action_request 的所有实例的集合。
    action_request_status:标识 action_request_status 的所有实例的集合。
    形式限制:
    WR1:对于 versioned_action_request 的每个实例,应确实有 action_request_status 的一个实例,其包括一个属性值等于 versioned_action_request 的那个实例的 assigned_request。
```

5.2.4.18 该版活动请求需要方案

此 versioned_action_request_requires_solution(该版活动请求需要方案)规则规定 versioned_action_request 的每个实例应该有一个或几个对它的建议方案。对于一个 versioned_action_request 的一个方案由该 action_request_solution 实体定义。

EXPRESS 描述:

```
*)
RULE versioned_action_request_requires_solution FOR
    (versioned_action_request,action_request_solution);
WHERE
    WR1:SIZEOF(QUERY(ar<* versioned_action_request|
        NOT(SIZEOF(QUERY(ars<* action_request_solution|
        ar:=:ars.request))>=1)))=0;
END_RULE;
(*
    变元定义:
    versioned_action_request:标识 versioned_action_request 的所有实例的集合。
    action_request_solution:标识 action_request_solution 的所有实例的集合。
    形式限制:
    WR1:对于 versioned_action_request 的每个实例,应确实存在 action_request_solution 的一个实例,其包括一个属性值等于 versioned_action_request 的那个实例的 request。
```

5.2.4.19 单个版本更改指令规则

此 unique_version_change_order_rule(单个版本更改指令规则)调用一个函数,如果一个 change 修改一个以上的 product_definition_formation,并且每次修改的 product_definition_formation 是一个不同

产品的一个版本，那么它返回真值。这规则规定单个 change 不应更改单个 product 的一个以上的版本，但是如果每个 product_definition_formation 引用了一个不同的 product，则它可以更改一个以上的 product_definition_formation。

EXPRESS 描述：

```
*)
RULE unique_version_change_order_rule FOR(change);
WHERE
    WR1:SIZEOF(QUERY(c<*change|
        NOT(unique_version_change_order(c.assigned_action))))=0;
END_RULE;
(*
```

变元定义：

change：标识 change 各实体的所有实例的集合。

形式限制：

WR1：对于 change 的每个实例，该 unique_version_change_order 函数应返回一个真值。

5.2.4.20 产品需要版本

此 product_requires_version（产品需要版本）规则规定 product 的每个实例应至少由 product_definition_formation 的一个实例所引用。这规则强调每个产品要有一个或几个版本。

EXPRESS 描述：

```
*)
RULE product_requires_version FOR(product,product_definition_formation);
WHERE
    WR1:SIZEOF(QUERY(prod<*product|
        NOT(SIZEOF(QUERY(pdf<*product_definition_formation|
        prod:=:pdf.of_product))>=1)))=0;
END_RULE;
(*
```

变元定义：

product：标识 product 各实体的所有实例的集合。

product_definition_formation：标识 product_definition_formation 各实体所有实例的集合。

形式限制：

WR1：对于 product 的每个实例，应有 product_definition_formation 的一个或几个实例，其包含一个属性值等于 product 的实例的 Of_product（产品的）。

5.2.4.21 产品需要人员组织

此 product_requires_person_organization（产品需要人员组织）规则规定 product 的每个实例应确实由 cc_design_person_and_organization_assignment 的一个实例所引用。这规则强调对于每个产品的要求要有一个“设计拥有者(design_owner)”。

EXPRESS 描述：

```
*)
RULE product_requires_person_organization FOR
    (product,cc_design_person_and_organization_assignment);
WHERE
    WR1:SIZEOF(QUERY(prod<* product|
```

```
        NOT(SIZEOF(QUERY(ccdpoa<*
        cc_design_person_and_organization_assignment|
        prod IN ccdpoa.items))=1)))=0;
END_RULE;
(*
```

变元定义：

product：标识 product 各实体的所有实例的集合。

cc_design_person_and_organization_assignment：标识 cc_design_person_and_organization_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 product 实体的每个实例，应有 cc_design_person_and_organization_assignment 的一个实例，其包含一个属性值等于 product 的实例的 items。

注：对于被赋值负责 product 的 person_and_organization 规定了此职能是“设计拥有者(design_owner)”，在 5.2.5.3 条定义的函数中形式上规定了这职能的缔合作用。

5.2.4.22 产品版本需要批准

此 product_version_requires_approval(产品版本需要批准)规则规定 product_definition_formation 的每个实例应确实由 cc_design_approval 的一个实例所引用。这规则强调对于一个设计的每个版本的要求要有一个批准。

EXPRESS 描述：

```
*)
RULE product_version_requires_approval FOR (product_definition_formation,
    cc_design_approval);
WHERE
    WR1:SIZEOF(QUERY(pdf<*product_definition_formation|
        NOT(SIZEOF(QUERY(ccda<* cc_design_approval|
        pdf IN ccda.items))=1)))=0;
END_RULE;
(*
```

变元定义：

product_definition_formation：标识 product_definition_formation 各实体的所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体所有实例的集合。

形式限制：

WR1：对于 product_definition_formation 实体的每个实例，应确实有 cc_design_approval 的一个实例，在其 items 集合中包含 product_definition_formation 的实例。

5.2.4.23 产品版本需要人员组织

product_version_requires_person_organization(产品版本需要人员组织)规则规定每个 product_definition_formation 应确实由创建者职能中的一个 cc_design_person_and_organization_assignment 和零件供应者或设计供应者职能中的一个或几个 cc_design_person_and_organization_assignment 所引用。这规则规定每个 product_definition_formation 必须有一个创建者或供应者负责该设计特定版本的创建或提交。这意味着在 5.2.4.43 条出现的创建者、零件供应者和设计提供者作为 restrict_person_organization_role 规则的定义部分。

注：在该 cc_design_person_and_organization_correlation 函数中规定对于不同的实体该不同职能值应与 person_and_organization_assignment 的赋值相一致。对于 cc_design_person_and_organization_assignment，局部调用这

个函数。见 5.2.5.3 条中定义的函数。

EXPRESS 描述：

```
*)
RULE product_version_requires_person_organization FOR
    (product_definition_formation,
    cc_design_person_and_organization_assignment);
WHERE
    WR1:SIZEOF(QUERY(pdf<*product_definition_formation|
        NOT(SIZEOF(QUERY(ccdpoa<*
        cc_design_person_and_organization_assignment|
        (pdf IN ccdpoa.items) AND (ccdpoa.role.name='creator')))=1)))=0;
    WR2:SIZEOF(QUERY(pdf<*product_definition_formation|
        NOT(SIZEOF(QUERY(ccdpoa<*
        cc_design_person_and_organization_assignment|
        (pdf IN ccdpoa.items) AND
        (ccdpoa.role.name IN['design_supplier','part_supplier'])))>=1)))
        =0;
END_RULE;
(*
```

变元定义：

product_definition_formation：标识 product_definition_formation 各实体的所有实例的集合。

cc_design_person_and_organization_assignment：标识 cc_design_person_and_organization_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 product_definition_formation 实体的每个实例，应确实有 cc_design_person_and_organization_assignment 的一个实例，在其 items 集合中包含 product_definition_formation 实例，并且它的 role 属性引用一个 person_and_organization_role 实例，其 name 属性值为“创建者(creator)”。

WR2：对于 product_definition_formation 实体的每个实例，应至少存在 cc_design_person_and_organization_assignment 的一个实例，在其 items 集合中包含 product_definition_formation 的实例，并且它的 role(职能)属性引用一个其 name 属性值为“设计提供者(design_supplier)”或“零件供应商(part_supplier)”的 person_and_organization_role。

5.2.4.24 产品版本需要安全分类

此 product_version_requires_security_classification(产品版本需要安全分类)规则规定 product_definition_formation 的每个实例应确实由 cc_design_security_classification 的一个实例所引用。这规则强调对于一个设计的每个版本的要求要有一个安全分类。

EXPRESS 描述：

```
*)
RULE product_version_requires_security_classification FOR
    (product_definition_formation,cc_design_security_classification);
WHERE
    WR1:SIZEOF(QUERY(pdf<*product_definition_formation|
        NOT(SIZEOF(QUERY(ccdsc<*cc_design_security_classification|
```

```
        pdf IN ccdsc.items))=1)))=0;
END_RULE;
(*
```

变元定义：

product_definition_formation：标识 product_definition_formation 各实体的所有实例的集合。

cc_design_security_classification：标识 cc_design_security_classification 各实体的所有实例的集合。

形式限制：

WR1：对于 product_definition_formation 的每个实例，应确实有 cc_design_security_classification 的一个实例，在其 items 集合中包含 product_definition_formation 的实例

5.2.4.25 产品定义需要人员组织

此 product_definition_requires_person_organization（产品定义需要人员组织）规则规定每个 product_definition 应确实由 cc_design_person_and_organization_assignment 的一个实例所引用。这规则规定对于每个 product_definition 必须有一个创建者负责该设计的特定定义的创建。创建者的含义在 person_and_organization_assignment 实体的 role 属性中出现。

注：在该 cc_design_person_and_organization_correlation。函数中规定对于不同的实体该不同职能值应与 person_and_organization_assignment 的赋值相一致。对于 cc_design_person_and_organization_assignment，局部调用这个函数。

EXPRESS 描述：

```
*)
RULE product_definition_requires_person_organization FOR
    (product_definition,
    cc_design_person_and_organization_assignment);
WHERE
    WR1:SIZEOF(QUERY(pd<* product_definition|
        NOT(SIZEOF(QUERY(ccdpoa<*
        cc_design_person_and_organization_assignment|
        pd IN ccdpoa.items))=1)))=0;
END_RULE;
(*
```

变元定义：

product_definition：标识 product_definition 各实体的所有实例的集合。

cc_design_person_and_organization_assignment：标识 cc_design_person_and_organization_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 product_definition 的每个实例，应确实有 cc_design_person_and_organization_assignment 的一个实例，在其 items 集合中包含 product_definition 的实例。

5.2.4.26 产品定义需要批准

此 product_definition_requires_approval（产品定义需要批准）规则规定 product_definition 的每个实例应确实由 cc_design_approval 的一个实例所引用。这规则强调对于一个设计的每个定义的要求要有一个批准。

EXPRESS 描述：

```
*)
RULE product_definition_requires_approval FOR
```

```
    (product_definition,cc_design_approval);
WHERE
    WR1:SIZEOF(QUERY(pd<* product_definition|
        NOT(SIZEOF(QUERY(ccda<* cc_design_approval|
        pd IN ccda.items))=1)))=0;
END_RULE;
(*
```

变元定义：

product_definition：标识 product_definition 各实体的所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体的所有实例的集合。

形式限制：

WR1：对于 product_definition 的每个实例，应确实有 cc_design_approval 的一个实例，在其 items 集合中包含 product_definition 的实例。

5.2.4.27 产品定义需要日期时间

此 product_definition_requires_date_time（产品定义需要日期时间）规则规定一个 product_definition 的每个实例应确实由一个 cc_design_date_and_time_assignment 所引用。这规则强调对于每个 product_definition 的要求要有一个日期指出该产品的定义创建的日期。

注：在该 cc_design_date_and_time_correlation。函数中规定对于不同的实体该不同职能值应与 date_and_time_assignment 的赋值相一致。对于 cc_design_date_and_time_assignment，局部调用这个函数。

EXPRESS 描述：

```
*)
RULE product_definition_requires_date_time FOR
    (product_definition,cc_design_date_and_time_assignment);
WHERE
    WR1:SIZEOF(QUERY(pd<* product_definition|
        NOT(SIZEOF(QUERY(ccdta<* cc_design_date_and_time_assignment|
        pd IN ccdta.items))=1)))=0;
END_RULE;
(*
```

变元定义：

product_definition：标识 product_definition 各实体的所有实例的集合。

cc_design_date_and_time_assignment：标识 cc_design_date_and_time_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 product_definition 实体的每个实例，确实有 cc_design_date_and_time_assignment 的一个实例，在其 items 集合中应包含 product_definition 的实例。

5.2.4.28 认证需要批准

此 certification_requires_approval（认证需要批准）规则规定 certification 的每个实例应确实由 cc_design_approval 的一个实例所引用。这规则强调设计或零件的供应者的每个认证的要求要有一个批准。

EXPRESS 描述：

```
*)
RULE certification_requires_approval FOR (certification,
    cc_design_approval);
```

```
WHERE
    WR1:SIZEOF(QUERY(cert<* certification|
        NOT(SIZEOF(QUERY(ccda<* cc_design_approval|
        cert IN ccda.items))=1)))=0;
END_RULE;
(*
```

变元定义：

certification：标识 certification 各实体所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体所有实例的集合。

形式限制：

WR1：对于 cc_design_approval 的每个实例，应确实有 cc_design_approval 的一个实例，在其 items 集合中包含 certification 的实例。

5.2.4.29 限制认证类型

此 restrict_certification_type(限制认证类型)规则规定 certification 只能给予“零件供应商(part_supplier)”或“设计提供者(design_supplier)”。

EXPRESS 描述：

```
*)
RULE restrict_certification_type FOR (certification_type);
WHERE
    WR1:SIZEOF(QUERY(ct<* certification_type|
        NOT(ct.description IN
        ['design_supplier','part_supplier']))=0;
END_RULE;
(*
```

变元定义：

certification_type：标识 certification_type 各实体的所有实例的集合。

形式限制：

WR1：对于 certification_type 的每个实例，该 kind 属性值应当是“设计提供者(design_supplier)”或“零件供应商(part_supplier)”。

属性值定义：

design_supplier：标识一个零件设计的提供者。

part_supplier：标识一个零件供应者。

5.2.4.30 认证需要日期时间

certification_requires_date_time(认证需要日期时间)规则规定一个 certification 的每个实例应确实由一个 cc_design_date_and_time_assignment 所引用。该规则强调对于每个 certification 的要求要有一个日期标识该认证有效的日期。

注：在该 cc_design_date_and_time_correlation 函数中规定对于不同的实体该不同职能值应与 date_time_assignment 的赋值相一致。对于 cc_design_date_and_time_assignment，局部调用这个函数。该函数定义见 5.2.5.4 条。

EXPRESS 描述：

```
*)
RULE certification_requires_date_time FOR
    (certification,cc_design_date_and_time_assignment);
WHERE
```

```
    WR1:SIZEOF(QUERY(cert<* certification|
        NOT(SIZEOF(QUERY(ccdta<* cc_design_date_and_time_assignment|
        cert IN ccdta.items))=1)))=0;
END_RULE;
(*
```

变元定义：

certification：标识 certification 各实体的所有实例的集合。

cc_design_date_and_time_assignment：标识 cc_design_date_and_time_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 certification 的每个实例，确实有 cc_design_date_and_time_assignment 的一个实例，在其 items 集合中包含 certification 的实例。

5.2.4.31 批准被赋值

此 approvals_are_assigned(批准被赋值)规则规定 approval 的每个实例应至少由批准被赋值实体的一个实例赋值。

EXPRESS 描述：

```
*)
RULE approvals_are_assigned FOR
    (approval,approval_assignment);
WHERE
    WR1:SIZEOF(QUERY(app<* approval|
        NOT(SIZEOF(QUERY(aa<* approval_assignment|
        app : = : aa.assigned_approval))>=1)))=0;
END_RULE;
(*
```

变元定义：

approval：标识 approval 各实体的所有实例的集合。

approval_assignment：标识 approval_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 approval 实体的每个实例，应当有 approval_assignment 的一个或几个实例，它包含 approval 的实例作为其 assigned_approval(赋值批准)属性。

5.2.4.32 批准需要批准人员组织

此 approval_requires_approval_person_organization(批准需要批准人员组织)规则规定 approval 的每个实例应当至少有一个 approval_person_organization 引用它。这规则强调对于一批准的需求由他们组织内的一个或多个人授权。

EXPRESS 描述：

```
*)
RULE approval_requires_approval_person_organization FOR (approval,
    approval_person_organization);
WHERE
    WR1:SIZEOF(QUERY(app<* approval|
        NOT(SIZEOF(QUERY(apo<* approval_person_organization|
        app : = : apo.authorized_approval))>=1)))=0;
```

```
END_RULE;
(*
    变元定义:
    approval:标识 approval 各实体的所有实例的集合。
    approval_person_organization:标识 approval_person_organization 各实体的所有实例的集合。
    形式限制:
    WR1:对于 approval 实体的每个实例,应当有 approval_person_organization 的一个或几个实例,
它包含 approval 实例作为其 authorized_approval(授权批准)属性。
```

5.2.4.33 批准需要批准日期时间

此 approval_requires_approval_date_time(批准需要批准日期时间)规则规定 approval 的每个实例应确实由一个 approval_date_time 所引用。这规则强调每个批准的要求要有一个日期指出该批准获得其特定状态的日期。

EXPRESS 描述:

```
*)
RULE approval_requires_approval_date_time FOR (approval,
    approval_date_time);
WHERE
    WR1:SIZEOF(QUERY(app<*approval|
        NOT(SIZEOF(QUERY(adt<*spproval_date_time|
        app:=:adt.dated_approval))=1)))=0;
END_RULE;
(*
    变元定义:
    approval:标识 approval 各实体的所有实例的集合。
    approval_date_time:标识 approval_date_time 各实体的所有实例的集合。
    形式限制:
    WR1:对于 approval 实体的每个实例,应确实有 approval_date_time 的一个实例,它包含 approval
的实例作为其 dated_approval(日期批准)属性。
```

5.2.4.34 限制批准状态

此 restrict_approval_status(限制批准状态)规则规定各 approval_status 的允许值是“已批准(approved)”,“尚未批准(not_yet_approved)”,“不批准(disapproved)”或“撤销(withdrawn)”之一。

EXPRESS 描述:

```
*)
RELE restrict_approval_status FOR(approval_status);
WHERE
  WR1:SIZEOF(QUERY(ast<*approval_status|
      NOT(ast.name IN
      ['approved','not_yet_approved','disapproved','withdrawn']))=0;
END_RULE;
(*
    变元定义:
    approval:标识 approval 各实体的所有实例的集合。
    形式限制:
```

WR1：对于 approval 的每个实例，该各 status 属性的值应当是“已准准(approved)”，“尚未批准(not_yet_approved)”，“不批准(disapproved)”或“撤销(withdrawn)”。

属性值定义：

approved：规定对一条产品数据批准的特定职能所要求的授权已获得。

not_yet_approved：规定对一条产品数据批准的特定职能所要求的授权在等待中。

disapproved：规定对一条产品数据批准的特定职能所要求的授权已否定。

withdrawn：规定对一条产品数据批准的特定职能所要求的授权已撤销。

5.2.4.35 合同需要批准

此 contract_requires_approval(合同需要批准)规则规定 contract 的每个实例应确实由 cc_design_approval 的一个实例所引用。这规则强调对于开发一个设计下的每个合同都要求有一个批准。

EXPRESS 描述：

```
*)
RULE contract_requires_approval FOR(contract,cc_design_approval);
WHERE
  WR1:SIZEOF(QUERY(c<*contract|
       NOT(SIZEOF(QUERY(ccda<*cc_design_approval|
       c IN ccda.items))=1)))=0;
END_RULE;
(*
```

变元定义：

contract：标识 contract 各实体的所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体的所有实例的集合。

形式限制：

WR1：对于 contract 实体的每个实例，确实有 cc_design_approval 的一个实例应存在，它引用其 items 集合中 contract 的实例。

5.2.4.36 合同需要人员组织

此 contract_requires_person_organization(合同需要人员组织)规则规定每个 contract 应确实由一个 cc_design_person_and_organization_assignment 所引用。这规则规定对于每个 contract 必须有由一个合同员负责有关合同信息的维护。合同员的含义在 person_and_organization_assignment 实体的 role 属性中出现。

注：在该 cc_design_person_and_organization_correlation。函数中规定对于不同的实体该不同职能值应与 person_and_organization_assignment 的赋值相一致。对于 cc_design_person_and_organization_assignment，局部调用这个函数。该函数定义见 5.2.5.3 条。

EXPRESS 描述：

```
*)
RULE contract_requires_person_organization FOR
  (contract,cc_design_person_and_organization_assignment);
WHERE
  WR1:SIZEOF(QUERY(c<*contract|
       NOT(SIZEOF(QUERY(ccdpoa<*
       cc_design_person_and_organization_assignment|
       c IN ccdpoa.items))=1)))=0;
EDN_RULE;
```

```
(*
```

变元定义：

contract：标识 contract 各实体的所有实例的集合。

cc_design_person_and_organization_assignment：标识 cc_design_person_and_organization_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 contract 的每个实例，应确实由 cc_design_person_and_organization_assignment 的一个实例，在其 items 集合中包含 contract 的实例。

5.2.4.37 限制合同类型

此 restrict_contract_type(限制合同类型)规则规定各合同允许的类型。这规则强调对于合同的各类型的要求应是“固定价格(fixed_price)”或“成本加价(cost_plus)”。

EXPRESS 描述：

```
*)
RULE restrict_contract_type FOR(contract_type);
WHERE
  WR1:SIZEOF(QUERY(ct<*contract_type|
       NOT(ct.description IN['fixed_price','cost_plus']))) = 0;
END_RULE;
(*
```

变元定义：

contract_type：标识 contract_type 各实体的所有实例的集合。

形式限制：

WR1：对于 contract_type 的每个实例，该 kind 属性应当包含“固定价格(fixed_price)”或“成本加价(cost_plus)”的一个值。

属性值定义：

fixed_price：标识一个合同的酬报已设定。

cost_plus：标识一个合同在完成后签约方在成本之上进行补偿的金额。

5.2.4.38 安全分类需要批准

此 security_classification_requires_approval(安全分类需要批准)规则规定 security_classification 的每个实例应确实由 cc_design_approval 的一个实例所引用。这规则强调对于每个赋值了一个设计的安全分类的要求要有一个批准。

EXPRESS 描述：

```
*)
RULE security_classification_requires_approval FOR
  (security_classification,cc_design_approval);
WHERE
  WR1:SIZEOF(QUERY(sc<*security_classification|
       NOT (SIZEOF(QUERY(ccda<* cc_design_approval|
       sc IN ccda.items))=1))) = 0;
END_RULE;
(*
```

变元定义：

security_classification：标识 security_classification 各实体的所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体的所有实例的集合。

形式限制：

WR1：对于 security_classification 实体的每个实例，应确实有 cc_design_approval 的一个实例，在其 items 集合中包含 security_classification 的实例。

5.2.4.39 安全分类需要人员组织

此 security_classification_requires_person_organization（安全分类需要人员组织）规则规定每个 security_classification 应确实由一个 cc_design_person_and_organization_assignment 所引用。这规则规定对于每个 security_classification 必须要有一个分类员负责发布该安全分类。该分类员的含义在 person_and_organization_assignment 实体的 role 属性中出现。

注：在该 cc_design_person_and_organization_correlation 函数中规定对于不同的实体该不同职能值应与 person_and_organization_assignment 的赋值相一致。对于 cc_design_person_and_organization_assignment，局部引用这个函数。该函数定义见 5.2.5.3 条。

EXPRESS 描述：

```
*)
RULE security_classification_requires_person_organization FOR
  (security_classification,
  cc_design_person_and_organization_assignment);
WHERE
  WR1:SIZEOF(QUERY(sc<*security_classification|
       NOT(SIZEOF(QUERY(ccdpoa<*
       cc_design_person_and_organization_assignment|
       sc IN ccdpoa.items))=1)))=0;
END_RULE;
(*
```

变元定义：

security_classification：标识 security_classification 各实体的所有实例的集合。

cc_design_person_and_organization_assignment：标识 cc_design_person_and_organization_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 security_classification 实体的每个实例，应确实有 cc_design_person_and_organization_assignment 的一个实例，在其 items 集合中包含 security_classification 的实例。

5.2.4.40 安全分类需要日期时间

此 security_classification_requires_date_time（安全分类需要日期时间）规则规定一个 security_classification 的每个实例应确实由一个具有“分类日期”(classification_date)”的一个职能的 cc_design_date_and_time_assignment 所引用。这规则强调对于每个 security_classification 的要求要有一个日期指出该安全分类有效的日期。

注：在该 cc_design_date_and_time_correlation 函数中规定对于不同的实体该不同职能值应与 date_time_assignment 的赋值相一致。对于 cc_design_date_and_time_assignment，局部调用这个函数。该函数定义见 5.2.5.4 条。

EXPRESS 描述：

```
*)
RULE security_classification_requires_date_time FOR
  (security_classification,cc_design_date_and_time_assignment);
WHERE
```

```
  WR1:SIZEOF(QUERY(sc<*security_classification|
        NOT(SIZEOF(QUERY(ccdta<*cc_design_date_and_time_assignment|
        (sc IN ccdta.items)AND
        ('classification_date'=ccdta.role.name)))=1)))=0;
END_RULE;
(*
```

变元定义：

security_classification：标识 security_classification 各实体的所有实例的集合。

cc_design_date_and_time_assignment：标识 cc_design_date_and_time_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 security_classification 实体的每个实例，应确实有 cc_design_date_and_time_assignment 的一个实例，在其 items 集合中包含 security_classification 的实例，并且其 the_role(该职能)属性引用 date_and_time_role 的一个实例，其 name 属性值为“分类日期(classification_date)”。

5.2.4.41 安全分类可选的日期时间

此 security_classification_optional_date_time(安全分类可选的日期时间)规则规定一个 security_classification 的每个实例可以被零个或一个具有一职能值为“取消分类日期(declassification_date)”的 cc_design_date_and_time_assignment 所引用。这规则强调对于每个 security_classification 可以有一个日期指出该安全分类过期的日期。

注：在该 cc_design_date_and_time_correlation。函数中规定对于不同的实体该不同职能值应与 date_time_assignment 的赋值相一致。对于 cc_design_date_and_time_assignment，局部调用这个函数。该函数定义见 5.2.5.4 条。

EXPRESS 描述：

```
*)
RULE security_classification_optional_date_time FOR
  (security_classification,cc_design_date_and_time_assignment);
WHERE
  WR1:SIZEOF(QUERY(sc<*security_classification|
        NOT(SIZEOF(QUERY(ccdta<*cc_design_date_and_time_assignment|
        (sc IN ccdta.items)AND
        ('declassification_date'=ccdta.role.name)))<=1)))=0;
END_RULE;
(*
```

变元定义：

security_classification：标识 security_classification 各实体的所有实例的集合。

cc_design_date_and_time_assignment：标识 cc_design_date_and_time_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 security_classification 的每个实例，可以有 cc_design_date_and_time_assignment 的零个或一个实例，在其 items 集合中包含 security_classification 实例，并且其 the_role 属性引用 date_and_time_role(日期时间职能)的一个实例，其 name 属性值为“取消分类日期(declassification_date)”。

5.2.4.42 限制安全分类等级

此 restrict_security_classification_level(限制安全分类等级)规则规定了允许的安全各级。这规则

强调此安全各级必须是“无级别(unclassified)”、“有级别(classified)”、“专有的(proprietary)”、“机密(confidential)”、“秘密(secret)”或“绝密(top_secret)”。

EXPRESS 描述：

```
*)
RULE restrict_security_classification_level FOR
  (security_classification_level);
WHERE
  WR1:SIZEOF(QUERY(scl<*security_classification_level|
       NOT(scl.name IN['unclassified','classified','proprietary',
       'confidential','secret','top_secret'])))=0;
END_RULE;
(*
```

变元定义：

security_classification_level：标识 security_classification_level 各实体的所有实例的集合。

形式限制：

WR1：对于 security_classification_level 的每个实例，其 name 属性值应为“无级别(unclassified)”，“有级别(classified)”，“专有的(proprietary)”，“机密(confidential)”，“秘密(secret)”或“绝密(top_secret)”。

属性值定义：

unclassified：标识该分类等级，对于此等级安全不是必需的。

classified：标识该分类等级，对于此等级安全是必需的，但没有给出等级细节。

proprietary：标识该分类等级，该等级的零件或零件设计信息的泄漏将影响一个组织的市场或竞争优势。

confidential：标识该分类等级，该等级的零件或零件设计信息的泄漏将对一个国家或组织的安全造成破坏。

secret：标识该分类等级，该等级的零件或零件设计信息的泄漏将对一个国家或组织的安全造成严重的破坏。

top_secret：标识该分类等级，该等级的零件或零件设计信息的泄漏将对一个国家或组织的安全造成最严重的破坏。

5.2.4.43 限制人员组织职能

此 restrict_person_organization_role(限制人员组织职能)规则规定 person_and_organization 各实体被允许的职能。这规则强调对 person_and_organization 各实体的各职能的要求是“请求接收者(request_recipient)”、“发起人(initiator)”、“零件供应商(part_supplier)”、“设计提供者(design_supplier)”、“配置管理员(configuration_manager)”、“合同员(contractor)”、“分类员(classification_officer)”、“创建者(creator)”或是“设计拥有者(design_owner)”。

EXPRESS 描述：

```
*)
RULE restrict_person_organization_role FOR(person_and_organization_role);
WHERE
  WR1:SIZEOF(QUERY(por<* person_and_organization_role|
       NOT(por.name IN ['request_recipient','initiator','part_supplier',
       'design_supplier','configuration_manager','contractor',
       'classification_officer','creator','design_owner'])))=0;
```

END_RULE;

(*

变元定义：

person_and_organization_role：标识 person_and_organization_role 各实体的所有实例的集合。

形式限制：

WR1：对于 person_and_organization_role 的每个实例，其 name 属性值应为“请求接收者(request_recipient)”、“发起人(initiator)”、“零件供应商(part_supplier)”、“设计提供者(design_supplier)”、“配置管理员(configuration_manager)”、“合同员(contractor)”、“分类员(classification_officer)”、“创建者(creator)”或是“设计拥有者(design_owner)”。

属性值定义：

request_recipient：标识一个组织中的人，他负责接收 change_request 或 start_request 并根据请求采取行动。

initiator：标识一个组织中的人，他负责创建 change_request 或 start_request。

part_supplier：标识一个组织中的人，他负责供应零件。

design_supplier：标识一个组织中的人，他负责提供一个零件的设计。

configuration_manager：标识一个组织中的人，他负责赋值有关一个设计的配置信息。

contractor：标识一个组织中的人，他负责有关应用到一个设计的合同信息。

classification_officer：标识一个组织中的人，他负责零件分类和取消零件分类。

creator：标识一个组织中的人，他负责创建一个特定的 product_definition_formation 或 product_definition。

design_owner：标识一个组织中的人，他负责一个 product 的所有设计。

5.2.4.44 限制日期时间职能

此 restrict_date_time_role(限制日期时间职能)规则规定 date_and_time 各实体被允许的职能。这规则强调对 date_and_time 各实体的各职能的要求是“创建日期(creation_date)”、“请求日期(request_date)”、“发放日期(release_date)”、“开始日期(start_date)”、“合同日期(contract_date)”、“认证日期(certification_date)”、“签发日期(sign_off_date)”、“分类日期(classification_date)”或“取消分类日期(declassification_date)”。

EXPRESS 描述：

*)

```
RULE restrict_date_time_role FOR(date_time_role);
WHERE
    WR1:SIZEOF(QUERY(dtr<*date_time_role|
        NOT(dtr.name IN ['creation_date','request_date','release_date',
        'start_date','contract_date','certification_date','sign_off_date',
        'classification_date','declassification_date'])))=0;
END_RULE;
```

(*

变元定义：

date_time_role：标识 date_time_role 各实体的所有实例的集合。

形式限制：

WR1：对于 date_time_role 的每个实例，其 name 属性值应为“创建日期(creation_date)”、“请求日期(request_date)”、“发放日期(release_date)”、“开始日期(start_date)”、“合同日期(contract_date)”、

“认证日期(certification_date)”、“签发日期(sign_off_date)”、“分类日期(classification_date)”或“取消分类日期(declassification_date)”。

属性值定义：

creation_date：标识一个设计版本或设计定义存在的时间和日期。

request_date：标识对一个设计工作请求的时间和日期。

release_date：标识一个设计初始发放或一个更改被结合进设计的时间和日期。

start_date：标识一个新设计或对一个现有设计进行更改开始工作的时间和日期。

contract_date：标识一个合同有效的时间和日期。

certification_date：标识一个认证有效的时间和日期。

sign_off_date：标识一个授权赋予一个批准的时间和日期。

classification_date：标识一个安全分类有效的时间和日期。

declassification_date：标识一个安全分类无效的时间和日期。

5.2.4.45 限制文档类型

此 restrict_document_type(限制文档类型)规则规定文档被允许的类型，这规则强调各文档的类型应是“材料规范(material_specification)”、“工艺规范(process_specification)”、“设计规范(design_specification)”或“表面粗糙度规范(surface_finish_specification)”。

EXPRESS 描述：

```
*)
RULE restrict_document_type FOR (document_type);
WHERE
  WR1:SIZEOF(QUERY(dt<*document_type|
        NOT (dt.product_data_type IN
        ['material_specification','process_specification',
        'design_specification','surface_finish_specification',
        'cad_filename','drawing']))=0;
END_RULE;
(*
```

变元定义：

document_type：标识 document_type 各实体的所有实例的集合。

形式限制：

WR1：对于 document_type 的每个实例，其 product_data_type(产品数据类型)属性值应为“材料规范(material_specification)”、“工艺规范(process_specification)”、“设计规范(design_specification)”或“表面粗糙度规范(surface_finish_specification)”。

属性值定义：

material_specification：标识一个文档类型，它规定了可应用的原材料、混合物或用于一个零件装配的半成品材料的特性。

process_specification：标识一个文档类型，它规定了在一个产品或材料上要进行的操作。

design_specification：标识一个文档类型，它建立一个零件的设计要求。

surface_finish_specification：标识一个文档类型，它规定可应用到一个加工或完整零件的表面纹理或保护层的特性。

cad_filename：标识一个文档类型，它被表示为一个来自计算机辅助设计系统的电子文件。

drawing：标识一个文档类型，它包含零件或零件设计的绘图表示。

5.2.4.46 文档与产品定义

此 document_to_product_definition(文档与产品定义)规则规定了用 document_relationship 和 product_definition 各实体组合在一起的各文档之间允许的关系。这规则强调了该要求,即只有 product_definition 实体可以有一组与其有关的文档。

EXPRESS 描述:

```
*)
RULE document_to_product_definition FOR
  (cc_design_specification_reference,product_definition);
WHERE
  WR1:SIZEOF(QUERY(sp<*cc_design_specification_reference|  .
       NOT((((′CONFIG_CONTROL_DESIGN.DOCUMENT_RELATIONSHIP′.+
       ′RELATING_DOCUMENT′IN
       ROLESOF(sp_reference.assigned_document))AND
       (SIZEOF(QUERY(it<*sp.items|
       NOT(′CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION′IN
       TYPEOF(it))))=0)))
       OR
       (NOT(′CONFIG_CONTROL_DESIGN.DOCUMENT_RELATIONSHIP′.+
       ′RELATING_DOCUMENT′IN
       ROLESOF(sp_reference.assigned_document))))))=0;
END_RULE;
(*
```

变元定义:

cc_design_specification_reference:标识 cc_design_specification_reference 各实体的所有实例的集合。

product_definition:标识产品定义实体所有实例的集合。

形式限制:

WR1:对于 cc_design_specification_reference 的每个实例,如果引用的 document 被用作一 document_relationship 实体中的 relating_document(关联文档),那么 items 集合中的每个元素应当是一个 product_definition 实体。

5.2.4.47 按所需数量

此 as_required_quantity(按所需数量)规则规定了 descriptive_measure 在 measure_with_unit 实体中的使用。descriptive_measure STRING 类型值应永远是“按所需(as_required)”。这规则强调了按所需量的使用规范要求。

EXPRESS 描述:

```
*)
RULE as_required_quantity FOR(measure_with_unit);
WHERE
  WR1:SIZEOF(QUERY(m<*measure_with_unit|
       (′CONFIG_CONTROL_DESIGN.DESCRIPTIVE_MEASURE′IN
       TYPEOF(m.value_component))AND
       (NOT(m.value_component=′as_required′))))=0;
END_RULE;
```

```
(*
```

变元定义：

measure_with_unit：标识 measure_with_unit 各实体的所有实例的集合。

形式限制：

WR1：对于 measure_with_unit 的每个实例，如果该属性 value(值)的类型是 descriptive_measure，那么该属性的值应是“按所需(as_required)”。

5.2.4.48 全局单位赋值

此 global_unit_assignment(全局单位赋值)规则规定对于 global_unit_assigned_context 应定义该各单位。这规则指出每个 global_unit_assigned_context 在其 unit 集合中应确实有三个元素，并且每个 global_unit_assigned_context 应包含长度，平面角和实体角的单位。

EXPRESS 描述：

```
*)
RULE global_unit_assignment FOR(global_unit_assigned_context);
WHERE
  WR1:SIZEOF(QUERY(guac<*global_unit_assigned_context|
       NOT(SIZEOF (guac.units)=3)))=0;
  WR2:SIZEOF(QUERY(guac<*global_unit_assigned_context|
       NOT((SIZEOF(QUERY(u<*guac.units|
       'CONFIG_CONTROL_DESIGN.LENGTH_UNIT' IN TYPEOF(u)))=1)AND
       (SIZEOF(QUERY(u<*guac.units|
       'CONFIG_CONTROL_DESIGN.PLANE_ANGLE_UNIT' IN TYPEOF(u)))=1)AND
       (SIZEOF(QUERY(u<*guac.units|
       'CONFIG_CONTROL_DESIGN.SOLID_ANGLE_UNIT' IN TYPEOF(u)))=1
       ))))=0;
END_RULE;
(*
```

变元定义：

global_unit_assigned_context：标识 global_unit_assigned_context 各实体的所有实例的集合。

形式限制：

WR1：对于 global_unit_assigned_context 的每个实例，该 units(单位)集合应确实包含三个元素。

WR2：对于 global_unit_assigned_context 的每个实例，该 units 集合的第一个元素应当是 length_unit，该 units 集合的第二个元素应当是 plane_angle_unit，units 集合的第三个元素应当是 solid_angle_unit。

5.2.4.49 子类型强制的活动

此 subtype_mandatory_action(子类型强制的活动)规则规定所有 action 实体应当是一个 directed_action 实体。

EXPRESS 描述：

```
*)
RULE subtype_mandatory_action FOR(action);
WHERE
  WR1:SIZEOF(QUERY(act<*action|
       NOT('CONFIG_CONTROL_DESIGN.DIRECTED_ACTION' IN TYPEOF(act))))=0;
END_RULE;
```

```
(*
    变元定义：
    action：标识被限制的 action 各实体的所有实例的集合。
  形式限制：
    WR1：action 的每个实例应当是一个 directed_action。
```

5.2.4.50 子类型强制的产品定义形成

此 subtype_mandatory_product_definition_formation(子类型强制的产品定义形成)规则规定所有 product_definition_formation 各实体应当是 product_definition_formation_with_specified_source 各实体。

EXPRESS 描述：

```
*)
RULE subtype_mandatory_product_definition_formation FOR
  (product_definition_formation);
WHERE
  WR1:SIZEOF(QUERY(pdf<*product_definition_formation|
       NOT('CONFIG_CONTROL_DESIGN'.+
       'PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE' IN
       TYPEOF(pdf))))=0;
END_RULE;
(*
    变元定义：
    product_definition_formation：标识了限制 product_definition_formation 各实体的所有实例的集合。
    形式限制：
    WR1：product_definition_formation 的每个实例应当是一个 parproduct_definition_formation_with_specified_sourcen。
```

5.2.4.51 依赖可实例化日期

此 dependent_instantiable_date(依赖可实例化日期)规则规定 date 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
*)
RULE dependent_instantiable_date FOR(date);
WHERE
  WR1:SIZEOF(QUERY(dt<*date|
       NOT(SIZEOF(USEDIN(dt,''))>=1)))=0;
END_RULE;
(*
    变元定义：
    date：标识 date 的所有实例的集合。
  形式限制：
    WR1：对于 date 的每个实例，应当有一个从其他实体的属性对该 date 实例的一个引用。
```

5.2.4.52 依赖可实例化形状表示

此 dependent_instantiable_shape_representation(依赖可实例化形状表示)规则规定 shape_repre-

sentation 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
*)
RULE dependent_instantiable_shape_representation FOR
    (shape_representation);
WHERE
  WR1:SIZEOF(QUERY sr<*shape_representation|
       NOT (SIZEOF(USEDIN(sr,''))>=1)))=0;
END_RULE;
(*
```

变元定义：

shape_representation：标识 shape_representation 的所有实例的集合。

形式限制：

WR1：对于 shape_representation 的每个实例，应当有一个从其他实体的一个属性对该 shape_representation 实例的一个引用。

5.2.4.53 依赖可实例化命名单位

此 dependent_instantiable_named_unit(依赖可实例化命名单位)规则规定 named_unit 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
*)
RULE dependent_instantiable_named_unitFOR(named_unit);
WHERE
  WR1:SIZEOF(QUERY(nu<*named_unit|
       NOT(SIZEOF(USEDIN(nu,''))>=1)))=0;
END_RULE;
(*
```

变元定义：

named_unit：标识 named_unit 的所有实例的集合。

形式限制：

WR1：对于 named_unit 的每个实例，应当有一个从其他实体的属性对 named_unit 实例的一个引用。

5.2.4.54 依赖可实例化表示项

此 dependent_instantiable_representation_item(依赖可实例化表示项)规则规定 representation_item 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
*)
RULE dependent_instantiable_representation_item FOR(representation_item);
WHERE
  WR1:SIZEOF(QUERY (ri<*representation_item|
       NOT(SIZEOF(USEDIN(ri,''))>=1)))=0;
       END_RULE;
(*
```

变元定义：

representation_item：标识 representation_item 实体所有实例的集合。

形式限制：

WR1：对于 representation_item 的每个实例，应当有一个从其他实体的属性对 representation_item 实例的一个引用。

5.2.4.55 依赖可实例化日期时间职能

此 dependent_instantiable_date_time_role(依赖可实例化日期时间职能)规则规定 date_time_role 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
*)
RULE dependent_instantiable_date_time_role FOR(date_time_role);
WHERE
  WR1:SIZEOF(QUERY(dtr<*date_time_role|
      NOT(SIZEOF(USEDIN(dtr,''))>=1)))=0;
END_RULE;
(*
```

变元定义：

date_time_role：标识 date_time_role 的所有实例的集合。

形式限制：

WR1：对于 date_time_role 的每个实例，应当有一个从其他实体的属性对 date_time_role 实例的一个引用。

5.2.4.56 依赖可实例化人员和组织职能

此 dependent_instantiable_person_and_organization_role(依赖可实例化人员和组织职能)规则规定 person_and_organization_role 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
*)
RULE dependent_instantiable_person_and_organization_roleFOR(person_and_organization_role);
WHERE
  WR1:SIZEOF(QUERY(poar<*person_and_organization_role|
      NOT(SIZEOF(USEDIN(poar,''))>=1)))=0;
END_RULE;
(*
```

变元定义：

person_and_organization_role：标识 person_and_organization_role 的所有实例的集合。

形式限制：

WR1：对于 person_and_organization_role 的每个实例，应当有一个从其他实体的一个属性对 person_and_organization_role 实例的一个引用。

5.2.4.57 依赖可实例化活动命令

此 dependent_instantiable_action_directive(依赖可实例化活动命令)规则规定 action_directive_role(活动命令职能)的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
*)
RULE dependent_instantiable_action_directive FOR(action_directive);
WHERE
  WR1:SIZEOF(QUERY(ad<*action_directive|
```

```
      NOT(SIZEOF(USEDIN(ad,''))>=1)))=0;
END_RULE;
(*
```

变元定义：

action_directive：标识 action_directive 的所有实例的集合。

形式限制：

WR1：对于 action_directive 的每个实例，应当有一个从其他实体的属性对 action_directive_role 实例的一个引用。

5.2.4.58 依赖可实例化安全分类等级

此 dependent_instantiable_security_classification_level（依赖可实例化安全分类等级）规则规定 security_classification_level 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
RULE dependent_instantiable_security_classification_levelFOR
      (security_classification_level);
WHERE
  WR1:SIZEOF(QUERY(scl<*security_classification_level|
       NOT(SIZEOF(USEDIN(scl,''))>=1)))=0;
END_RULE;
(*
```

变元定义：

security_classification_level：标识 security_classification_level 的所有实例的集合。

形式限制：

WR1：对于 security_classification_level 的每个实例，应当有一个从其他实体的一个属性对 security_classification_level 实例的一个引用。

5.2.4.59 依赖可实例化批准状态

此 dependent_instantiable_approval_status（依赖可实例化批准状态）规则规定 approval_status 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
*)
RULE dependent_instantiable_approval_status FOR(approval_status);
WHERE
WR1:SIZEOF(QUERY(ast<*approval status|
     NOT(SIZEOF(USEDIN(ast,''))>=1)))=0;
END_RULE;
(*
```

变元定义：

approval_status：标识各 approval_status 的所有实例的集合。

形式限制：

WR1：对于 approval_status 的每个实例应当有一个从其他实体的属性对 approval_status 实例的一个引用。

5.2.4.60 依赖可实例化文档类型

此 dependent_instantiable_document_type(依赖可实例化文档类型)规则规定 document_type 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述:

```
*)
RULE dependent_instantiable_document_type FOR(document_type);
WHERE
  WR1:SIZEOF(QUERY(dt<*document_type|
      NOT(SIZEOF(USEDIN(dt,''))>=1)))=0;
END_RULE;
(*
```

变元定义:

document_type:标识 document_type 的所有实例的集合。

形式限制:

WR1:对于 document_type 的每个实例应当有一个从其他实体的属性对 document_type 实例的一个引用。

5.2.4.61 依赖可实例化合同类型

此 dependent_instantiable_contract_type(依赖可实例化合同类型)规则规定 contract_type 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述:

```
*)
RULE dependent_instantiable_contract_type FOR(contract_type);
WHERE
  WR1:SIZEOF(QUERY(ct<*contract_type|
      NOT(SIZEOF(USEDIN(ct,''))>=1)))=0;
END_RULE;
(*
```

变元定义:

contract_type:标识 contract_type 的所有实例的集合。

形式限制:

WR1:对于 contract_type 的每个实例,应当有一个从其他实体的属性对 contract_type 实例的一个引用。

5.2.4.62 依赖可实例化认证类型

此 dependent_instantiable_certification_type(依赖可实例化认证类型)规则规定 certification_type 的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述:

```
*)
RULE dependent_instantiable_certification_type FOR(certification_type);
WHERE
  WR1:SIZEOF(QUERY(ct<*certification_type|
      NOT(SIZEOF(USEDIN(ct,''))>=1)))=0;
END_RULE;
(*
```

变元定义:

certification_type:标识 certification_type 的所有实例的集合。

形式限制:

WR1:对于 certification_type 的每个实例,应当有一个从其他实体的属性对 certification_type 实例的一个引用。

5.2.4.63 产品概念需要配置项

此 product_concept_requires_configuration_item(产品概念需要配置项)规则规定每个 product_concept 应至少由一个 configuration_item 所引用。这规则强调对每个 product_concept 中要求至少有一个 configuration_item。

EXPRESS 描述:

```
*)
RULE product_concept_requires_configuration_item FOR
  (product_concept,configuration_item);
WHERE
  WR1:SIZEOF(QUERY(pc<*product_concept|
       NOT(SIZEOF(QUERY(ci<*configuration_item|
       pc:=:ci.item_concept))>=1)))=0;
END_RULE;
(*
```

变元定义:

product_concept:标识 product_concept 各实体的所有实例的集合。

configuration_item:标识 configuration_item 各实体的所有实例的集合。

形式限制:

WR1:对于 product_concept 的每个实例,应当至少有 configuration_item 的一个实例包含 product_concept 的实例作为其属性 item_concept(项概念)的值。

5.2.4.64 配置项需要人员组织

此 configuration_item_requires_person_organization(配置项需要人员组织)规则规定每个 configuration_item 应确实由一个 cc_design_person_and_organization_assignment 所引用。这规则规定每个 configuration_item 要求有一个配置管理员负责。配置管理员的含义在 person_and organization_assignment 实体的 role 属性中出现。

注:在该 cc_design_person_and_organization_correlation 函数中规定对于不同的实体该不同职能值应与 person_and_organization_assignment 的赋值相一致。对于 cc_design_person_and_organization_assignment,局部调用这个函数。该函数定义见 5.2.5.3 条。

EXPRESS 描述:

```
*)
RULE configuration_item_requires_person_organization FOR
  (configuration_item,cc_design_person_and_organization_assignment);
WHERE
  WR1:SIZEOF(QUERY(ci<*configuration_item|
       NOT(SIZEOF(QUERY(ccdpoa<*
       cc_design_person_and_organization_assignment|
       ci IN ccdpoa.items))=1)))=0;
END_RULE;
(*
```

变元定义：

configuration_item：标识 configuration_item 各实体的所有实例的集合。

cc_design_person_and_organization_assignment：标识 cc_design_person_and_organization_assignment 各实体的所有实例的集合。

形式限制：

WR1：对于 configuration_item 的每个实例，应确实有 cc_design_person_and_organization_assignment 的一个实例在其 items 集合中包含 configuration_item 的实例。

5.2.4.65 子类型强制的有效性

此 subtype_mandatory_effectivity（子类型强制的有效性）规则规定该 effectivity 各实体的每个实例应当是一个复合的实体实例，它由 serial_numbered_effectivity（顺序有效性）、dated_effectivity（日期有效性）或 lot_effectivity（批量有效性）实体之一的一个实例和一个 configuration_effectivity（配置有效性）实例组成。

EXPRESS 描述：

```
*)
RULE subtype_mandatory_effectivity FOR
  (effectivity);
WHERE
  WR1:SIZEOF(QUERY(eff<*effectivity|
       NOT (( SIZEOF ([ 'CONFIG_ CONTROL_ DESIGN. SERIAL_ NUMBERED_
       EFFECTIVITY',
       'CONFIG_CONTROL_DESIGN. LOT_EFFECTIVITY',
       'CONFIG_CONTROL_DESIGN. DATED_EFFECTIVITY']*
       TYPEOF(eff))=1)AND
       ('CONFIG_ CONTROL_ DESIGN. CONFIGURATION_ EFFECTIVITY'IN TYPEOF
       (eff)))))
       =0;
END_RULE;
(*
```

变元定义：

effectivity：标识 effectivity 各实体的所有实例的集合。

形式限制：

WR1：effectivity 的每个实例应当是一个 serial_numbered_effectivity，dated_effectivity 或 lot_effectivity 和一个 configuration_effectivity 实例。

5.2.4.66 有效性需要批准

此 effectivity_requires_approval（有效性需要批准）规则规定一个 effectivity 的每个实例应确实由一个 cc_design_approval 实体所引用。这规则强调对每个 effectivity 要求要有一个批准。

EXPRESS 描述：

```
*)
RULE effectivity_requires_approval FOR
  (effectivity. cc_design_approval);
WHERE
  WR1:SIZEOF(QUERY(eff<*effectivity|
```

```
        NOT(SIZEOF(QUERY(ccda<*cc_design_approval|
        eff IN ccda.items))=1)))=0;
END_RULE;
(*
```

变元定义：

effectivity：标识 effectivity 各实体的所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体的所有实例的集合。

形式限制：

WR1：对于 effectivity 的每个实例，应确实有 cc_design_approval 的一个实例，在其 items 集合中包含 effectivity 的实例。

5.2.4.67 配置项需要批准

此 configuration_item_requires_approval（配置项需要批准）规则规定一个 configuration_item 的每个实例应确实由一个 cc_design_approval 所引用。这规则强调对每个 configuration_item 的要求要有一个批准。

EXPRESS 描述：

```
*)
RULE configuration_item_requires_approval FOR
  (configuration_item,cc_design_approval);
WHERE
  WR1:SIZEOF(QUERY(ci<*configuration_item|
        NOT(SIZEOF(QUERY(ccda<*cc_design_approval|
        ci IN ccda.items))=1)))=0;
END_RULE;
(*
```

变元定义：

configuration_item：标识 configuration_item 各实体的所有实例的集合。

cc_design_approval：标识 cc_design_approval 各实体的所有实例的集合。

形式限制：

WR1：对于 configuration_item 的每个实例，应确实有 cc_design_approval 的一个实例，在其 items 集合中包含 configuration_item 的实例。

5.2.4.68 协调的装配和形状

此 coordinated_assembly_and_shape（协调的装配和形状）规则规定两个 product_definition 实体之间的关系和两个 shape_representation 实体之间的关系必须显式地通过 context_dependent_shape_representation（相关环境依赖形状表示）实体建立联系，其中两个产品定义实体分别是 next_assembly_usage_occurrence 中的装配件和构件，两个形状表示实体分别包含 shape_representation_relationship（形状表示关系）实体中的装配件和构件的形状表示。该规则调用 assembly_shape_is_defined（装配形状被定义）函数，如果装配关系和形状关系被显式地建立，它返回真值。

EXPRESS 描述：

```
*)
RULE coordinated_assembly_and_shape FOR(next_assembly_usage_occurrence);
WHERE
  WR1:SIZEOF(QUERY(nauo<*next_assembly_usage_occurrence|
```

```
        NOT assembly_shape_is_defined(nauo)))=0;
END_RULE;
(*
```

变元定义：

next_assembly_usage_occurrence：标识 next_assembly_usage_occurrence 各实体的所有实例的集合。

形式限制：

WR1：对于 next_assembly_usage_occurrence 的每个实例，该 assembly_shape_is_defined 函数将返回真值。

5.2.4.69 子类型强制的产品定义使用

此 subtype_mandatory_product_definition_usage（子类型强制的产品定义使用）规则规定所有 product_definition_usage（产品定义使用）实体都应当是 assembly_component_usage（装配构件使用）。

EXPRESS 描述：

```
*)
RULE subtype_mandatory_product_definition_usage FOR
  (product_definition_usage);
WHERE
  WR1:SIZEOF(QUERY(acu<*product_definition_usage|
        NOT('CONFIG_CONTROL_DESIGN.'+
        'ASSEMBLY_COMPONENT_USAGE'IN TYPEOF(acu))))=0;
END_RULE;
(*
```

变元定义：

product_definition_usage：标识被限制的 product_definition_usage 各实体的所有实例的集合。

形式限制：

WR1：product_definition_usage 每个实例都应当是一个 assembly_component_usage。

5.2.4.70 装配构件使用需要安全分类

此 acu_requires_security_classification（装配构件使用需要安全分类）规则规定每个 assembly_component_usage 应确实由一个 cc_design_security_classification（配置控制设计安全分类）所引用。该规则强调对于在一个 assembly_component_usage 中的 related_product_definition（关联的产品定义）的每个 product_definition 在该 assembly_component_usage 的相关环境内要有一个安全分类的要求。

EXPRESS 描述：

```
*)
RULE acu_requires_security_classification FOR
  (assembly_component_usage,
  cc_design_security_classification);
WHERE
  WR1:SIZEOF (QUERY(acu<*assembly_component_usage|
        NOT(SIZEOF(QUERY(ccdsc<*cc_design_security_classification|
        acu IN ccdsc.items))=1)))=0;
END_RULE;
(*
```

变元定义：

assembly_component_usage：标识受限的 assembly_component_usage 各实体的所有实例的集合。

cc_design_security_classification：标识受限制的 cc_design_security_classification 各实体的所有实例的集合。

形式限制：

WR1：对于 assembly_component_usage 实体的每个实例，应确实有 cc_design_security_classification 的一个实例，在其 items 集合中包含 assembly_component_usage 的实例。

5.2.4.71 三维几何表示项

此 geometric_representation_item_3d(三维几何表示项)规则规定每个 geometric_representation_item 必须出现在一个 geometric_representation_context(几何表示相关环境)中，该实体除了用来定义 pcurve 实体，还有一个维数为 3。这规则限制所有三维几何。该 pcurve 实体是一个例外，因为它必须作为一个曲面的参数域出现在二维中。

EXPRESS 描述：

```
*)
RULE geometric_representation_item_3d FOR
  (geometric_representation_item);
WHERE
  WR1:SIZEOF(QUERY(gri<*geometric_representation_item|
       NOT((dimension_of(gri)=3)OR
       (SIZEOF(bag_to_set(USEDIN(gri,'')))-bag_to_set(USEDIN(gri,
       'CONFIG_CONTROL_DESIGN.DEFINITIONAL_REPRESENTATION.ITEMS')))=0)
       )))=0;
END_RULE;
(*
```

变元定义：

geometric_representation_item：标识受限制的 geometric_representation_item 各实体的所有实例的集合。

形式限制：

WR1：对于 geometric_representation_item 的每个实例，dim 的值应当是 3 或 geometric_representation_item 应当被用作为 definitional_representation(定义表示)中的一项。

注：该 definitional_representation 的使用是由 pcurve 实体用来定义一个曲面的参数域中的点或曲线。

5.2.4.72 依赖可实例化参数表示相关环境

此 dependent_instantiable_parametric_representation_context(依赖可实例化参数表示相关环境)规则规定 parametric_representation_context(参数表示相关环境)的所有实例依赖于定义另一个实体的使用。

EXPRESS 描述：

```
*)
RULE dependent_instantiable_parametric_representation_context FOR
  (parametric_representation_context);
WHERE
  WR1:SIZEOF(QUERY(prc<*parametric_representation_context|
       NOT(SIZEOF(USEDIN(prc,''))>=1)))=0;
END_RULE;
```

```
(*
```

变元定义：

parametric_representation_context：标识 parametric_representation_context 的所有实例的集合。

形式限制：

WR1：对于 parametric_representation_context 的每个实例，应当有一个从其他实体的一个属性对该 parametric_representation_context 的一个引用。

5.2.4.73 子类型强制的形状表示

此 subtype_mandatory_shape_representation（子类型强制的形状表示）规则请求所有的 shape_representation 各实体对于或者是 geometrically_bounded_wireframe_shape_representation，geometrically_bounded_surface_shape_representation，edge_based_wireframe_shape_representation，shell_based_wireframe_shape_representation，manifold_surface_shape_representation，faced_brep_shape_representation，或者是 advanced_brep_shape_representation，或者是在它们各自的 items 集合中只包含 axis2_placement_3d 实体，或是一个 shape_aspect 的表示，或两 shape_aspect 之间的关系。该规则定义了限制，它建立了本标准中对各形状允许的不同表示类型。

EXPRESS 描述：

```
*)
RULE subtype_mandatory_shape_representation FOR
  (shape_representation);
WHERE
  WR1:SIZEOF(QUERY(sr<*shape_representation|
        NOT((SIZEOF(['CONFIG_CONTROL_DESIGN.'+
        'ADVANCED_BREP_SHAPE_REPRESENTATION',
        'CONFIG_CONTROL_DESIGN.FACETED_BREP_SHAPE_REPRESENTATION',
        'CONFIG_ CONTROL_ DESIGN. MANIFOLD_ SURFACE_ SHAPE_
        REPRESENTATION',
'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION',
'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION',
        'CONFIG_CONTROL_DESIGN'.+
        'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION',
        'CONFIG_CONTROL_DESIGN'.+
        'GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION']*
        TYPEOF(sr))=1)OR
        (SIZEOF(QUERY(it<*sr.items|
        NOT('CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'IN TYPEOF(it))))
        =0)OR
        (SIZEOF(QUERY(sdr<*QUERY(pdr<*USEDIN(sr,
        'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION_REPRESENTATION'.+
        'USED_REPRESENTATION')|
        'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION'IN
        TYPEOF(pdr))|
        NOT(SIZEOF(['CONFIG_CONTROL_DESIGN.SHAPE_ASPECT',
        'CONFIG_CONTROL_DESIGN.SHAPE_ASPECT_RELATIONSHIP']*TYPEOF
```

```
        (sdr.definition))=1)))=0))))=0;
END_RULE;
(*
    变元定义:
    shape_representation:标识被限制的 shape_representation 的所有实例的集合。

    形式限制:
    WR1:shape_representation 的每个实例应当是
geometrically_bounded_wireframe_shape_representation,
geometrically_bounded_surface_shape_representation,edge_based_wireframe_shape_representation,
shell_based_wireframe_shape_representation,
manifold_surface_shape_representation,faced_brep_shape_representation 或 advanced_brep_shape_
representation 或在它们各自的 items 集合中只包含 axis2_placement_3d 实体或是一个 shape_aspect
的表示或 shape_aspect_relationship。
```

5.2.4.74 子类型强制的表示

此 subtype_mandatory_representation(子类型强制的表示)规则要求所有 representation 各实体都是 shape_representation。

EXPRESS 描述:

```
*)
RULE subtype_mandatory_representation FOR(representation);
WHERE
  WR1:SIZEOF(QUERY(rep<*representation|
        NOT ('CONFIG_CONTROL_DESIGN.SHAPE_REPRESENTATION'IN TYPEOF
        (rep))))
        =0;
END_RULE;
(*
    变元定义:
    representation:标识受限制的 representation 各实体的所有实例的集合。

    形式限制:
    WR1:representation 的每个实例应当是一个 shape_representation 实体。
```

5.2.4.75 子类型强制的表示相关环境

此 subtype_mandatory_representation_context(子类型强制的表示相关环境)规则要求所有 representation_context(表示相关环境)各实体都是 geometric_representation_context。

EXPRESS 描述:

```
*)
RULE subtype_mandatory_representation_context FOR
  (representation_context);
WHERE
  WR1:SIZEOF(QUERY(rep_cntxt<*representation_context|
        NOT('CONFIG_CONTROL_DESIGN.GEOMETRIC_REPRESENTATION_
        CONTEXT'IN
```

```
        TYPEOF(rep_cntxt))))=0;
END_RULE;
(*
```

变元定义：

representation_context：标识受限制的 representation_context 实体的所有实例的集合。

形式限制：

WR1：representation_context 的每个实例应当是一个 geometric_representation_context。

5.2.4.76 无参照形状

此 no_shape_for_make_from(无参照形状)规则保证作为 design_make_from_relationship 实体的 product_definition_relationship 实体没有定义形状。这规则通过不允许 design_make_from_relationship 实体在一个 product_definition_shape 实体中使用而起作用，该 product_definition_shape 实体建立一个零件的形状或一个装配件中替代件的相关形状。

EXPRESS 描述：

```
*)
RULE no_shape_for_make_from FOR
  (design_make_from_relationship);
WHERE
  WR1:SIZEOF(QUERY(dmfr<*design_make_from_relationship|
        NOT(SIZEOF(QUERY(pd<*USEDIN(dmfr,'CONFIG_CONTROL_DESIGN.'+
        'PROPERTY_DEFINITION.DEFINITION')|
        'CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION_SHAPE'INTYPEOF(pd)))=
        0)))=0;
END_RULE;
(*
```

变元定义：

design_make_from_relationship：标识受限制的 design_make_from_relationship 实体的所有实例的集合。

形式限制：

WR1：没有 design_make_from_relationship 实体应由一个 property_definition 实体的 definition 属性所引用，该 property_definition 实体是一个 product_definition_shape。

5.2.4.77 无供应零件形状

此 no_shape_for_supplied_part(无供应零件形状)规则保证作为 supplied_part_relationship(供应零件关系)各实体 product_definition_relationship 各实体没有定义形状。这规则通过不允许 supplied_part_relationship 实体在一个 product_definition_shape 实体中使用而起作用。该 product_definition_shape 实体建立一个零件的形状或一个装配件中替代件的相关形状。

EXPRESS 描述：

```
*)
RULE no_shape_for_supplied_part FOR
  (supplied_part_relationship);
WHERE
  WR1:SIZEOF(QUERY(spr<*supplied_part_relationship|
        NOT(SIZEOF(QUERY(pd<*USEDIN(spr,'CONFIG_CONTROL_DESIGN.'+
```

```
        'PROPERTY_DEFINITION.DEFINITION')|
        'CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION_SHAPE'IN TYPEOF(pd)))=
        0)))=0;
END_RULE;
(*
```

变元定义：

supplied_part_relationship：标识受限制的 supplied_part_relationship 实体所有实例的集合。

形式限制：

WR1：没有 supplied_part_relationship 实体应由一个 property_definition 实体的 definition 属性所引用，该 property_definition 实体是一个 product_definition_shape。

5.2.5 配置控制设计函数

5.2.5.1 单个版本更改指令

如果由 action_execution（活动执行）调用的 ordered_action（指令活动）使 requested_action（请求活动）各实体具体化，则 unique_version_change_order（单个版本更改指令）布尔函数接受一个 action_execution 作为输入，并返回真值。该 requested_action 各实体引用不同的 product_definition_formation 各实体，那些 product_definition_formation 实体引用不同的 product 各实体。如果 requested_action 各实体引用了由一个 action_execution 调用的 product_definition_formation 各实体，并且那些 product_definition_formation 实体引用相同的 product，则该函数返回假值。如果一个更改改变一个零件的多个版本，并且那些版本是不同零件的各版本，该函数将返回真值。单个更改不可能改变单个零件的不同版本。

EXPRESS 描述：

```
*)
FUNCTION unique_version_change_order(c:action):BOOLEAN;
  LOCAL
    ords     :action_directive:=c_action.directive;
    assign   :SET OF change_request:=[];
    versions :SET OF product_definition_formation:=[];
END_LOCAL;
    建立各 change_request 的集合，它是已知的 versioned_action_request，由 action_directive 调用。
REPEAT i:=1TO SIZEOF(ords.requests);
assign:=assign+QUERY(ara<*bag_to_set(USEDIN(ords.requests[i],
          'CONFIG_CONTROL_DESIGN.ACTION_REQUEST_ASSIGNMENT.'+
          'ASSIGNED_ACTION_REQUEST'))|
          'CONFIG_CONTROL_DESIGN.CHANGE_REQUEST'IN TYPEOF(ara));
END_REPEAT;
    收集各 product_definition_formations，它由各 change_requests 引用。
REPEATk:=1 TO SIZEOF(assign);
  versions:=versions+assign[k].items;
END_REPEAT;
    检查没有 product_definition_formation 引用同样的产品实例

RETURN (SIZEOF(QUERY(vers<*versions|
```

```
        NOT(SIZEOF(QUERY(other_vers<*versions-vers|
        vers.of_product:=:other_vers.of_product))=0)))=0);
END_FUNCTION;
(*
变元定义:
c:(输入)标识要检查的 directed_action(指定活动)。
```

5.2.5.2 袋—集合—转换

bag_to_set(袋—集合—转换)函数接受一个袋数据类型作为输入,把袋中的每个元素赋予集合中的每个元素并返回一个集合。bag_to_set 函数从袋中清除重复元素作为集合赋值操作的结果。

EXPRESS 描述:

```
*)
FUNCTION bag_to_set
  (the_bag:BAG OF GENERIC:intype):SET OF GENERIC:intype;
  LOCAL
    the_set :SET OF GENERIC:intype:=[];
    i       :INTEGER;
  END_LOCAL;
  IF SIZEOF(the_bag)>0THEN
    REPEAT i:=1 to HIINDEX(the_bag);
      the_set:=the_set+the_bag[i];
    END_REPEAT;
  END_IF;
  RETURN(the_set);
END_FUNCTION;
(*
变元定义:
the_bag:(输入)要转换成集合的袋。
```

5.2.5.3 配置控制设计人员和组织相互关系

如果 person_organization_role 实体的 name 属性值与 cc_design_person_and_organization_assignment 实体的 items 中所选的实体类型相一致,cc_design_person_and_organization_correlation(配置控制设计人员和组织相互关系)布尔函数返回真值。

该函数说明如下:

——赋予一个 change_request 或 start_request 的人员和组织可以具有"request_recipient"的职能。

——赋予一个 change_request,start_request,change 或 start_work 的人员和组织可以具有"initiator"的职能。

——赋予一个 product_definition_formation 或 product_definition 的人员和组织可以具有"creator"的职能。

——赋予一个 product_definition_formation 的人员和组织可以具有"part_supplier"的职能。

——赋予一个 product_definition_formation 的人员和组织可以具有"design_supplier"的职能。

——赋予一个 product 的人员和组织可以具有"design_owner"的职能。

——赋予一个 configuration_item 的人员和组织可以具有"configuration_manager"的职能。

——赋予一个 contract 的人员和组织可以具有“contractor”的职能。

——赋予一个 security_classification 的人员和组织可以具有“classification_officer”的职能。

EXPRESS 描述：

```
*)
FUNCTION cc_design_person_and_organization_correlationcorrelation
  (e:cc_design_person_and_organization_assignment):BOOLEAN;
LOCAL
    po_role:STRING;
  END_LOCAL;
    po_role:=e_and_organization_assignment.role.name;
  CASE po_role OF
    'request_recipient'       :IF SIZEOF(e.items)<>
                                 SIZEOF(QUERY(x<*e.items|
                                 SIZEOF(['CONFIG_CONTROL_DESIGN.'+
                                 'CHANGE_REQUEST',
                                 'CONFIG_CONTROL_DESIGN.'+
                                 'START_REQUEST']*
                                 TYPEOF(x))=1))
                                 THEN RETURN(FALSE);
                               END_IF;
    'initiator'               :IF SIZEOF(e.items)<>
                                 SIZEOF(QUERY(x<*e.items|
                                 SIZEOF(['CONFIG_CONTROL_DESIGN.'+
                                 'CHANGE_REQUEST',
                                 'CONFIG_CONTROL_DESIGN.'+
                                 'START_REQUEST',
                                 'CONFIG_CONTROL_DESIGN.'+
                                 'START_WORK',
                                 'CONFIG_CONTROL_DESIGN.'+
                                 CHANGE']*
                                 TYPEOF(x))=1))
                                 THEN RETURN(FALSE);
                               END_IF;
    'creator'                 :IF SIZEOF(e.items)<>
                                 SIZEOF(QUERY(x<*e.items|
                                 SIZEOF(['CONFIG_CONTROL_DESIGN.'+
                                 'PRODUCT_DEFINITION_FORMATION',
                                 'CONFIG_CONTROL_DESIGN.'+
                                 'PRODUCT_DEFINITION']*
                                 TYPEOF(x))=1))
                                 THEN RETURN(FALSE);
                               END_IF;
    'part_supplier'           :IF SIZEOF(e.items)<>
```

```
                                SIZEOF(QUERY(x<*e.items|
                                'CONFIG_CONTROL_DESIGN'.+
                                'PRODUCT_DEFINITION_FORMATION'
                                IN TYPEOF(x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'design_supplier'       :IF SIZEOF(e.items)<>
                                SIZEOF(QUERY(x<*e.items|
                                'CONFIG_CONTROL_DESIGN.'+
                                'PRODUCT_DEFINITION_FORMATION'
                                IN TYPEOF(x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'design_owner'          :IF SIZEOF(e.items)<>
                                SIZEOF(QUERY(x<*e.items|
                                'CONFIG_CONTROL_DESIGN.PRODUCT'
                                IN TYPEOF(x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'configuration_manager' :IF SIZEOF(e.items)<>
                                SIZEOF(QUERY(x<*e.items|
                                'CONFIG_CONTROL_DESIGN.'+
                                'CONFIGURATION_ITEM'
                                IN TYPEOF(x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'contractor'            :IF SIZEOF(e.items)<>
                                SIZEOF(QUERY(x<*e.items|
                                'CONFIG_CONTROL_DESIGN.CONTRACT'
                                IN TYPEOF(x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'classification_officer' :IF SIZEOF(e.items)<>
                                SIZEOF(QUERY(x<*e.items|
                                'CONFIG_CONTROL_DESIGN.'+
                                'SECURITY_CLASSIFICATION'
                                IN TYPEOF(x)))THEN
                                RETURN(FALSE);
                              END_IF;
    OTHERWISE:RETURN(TRUE);
  END_CASE;
  RETURN(TRUE);
END_FUNCTION;
```

(*

变元定义：

e：(输入)标识要检查的 cc_design_person_and_organization_assignment 实体。

5.2.5.4 配置控制设计日期和时间相互关系

如果 date_time_role 实体的 name 属性值与 cc_design_date_and_time_assignment 实体的 items 中所选的实体类型相一致，则 cc_design_date_time_correlation(配置控制设计日期和时间相互关系)布尔函数返回真值。

该函数说明如下：

赋予一个 product_definition 的日期和时间可以具有“creation_date”的职能。

赋予一个 change_request 或 start_request 的日期和时间可以具有“request_date”的职能。

赋予一个 change 或 start_work 的日期和时间可以具有“release_date”的职能。

赋予一个 change 或 start_work 的日期和时间可以具有“start_date”的职能。

赋予一个 approval_date_time 的日期和时间可以具有“sign_off_date”的职能。

赋予一个 contract 的日期和时间可以具有“contract_date”的职能。

赋予一个 certification 的日期和时间可以具有“certification_date”的职能。

赋予一个 security_classification 的日期和时间可以具有“classification_date”的职能。

赋予一个 security_classification 的日期和时间可以具有“declassification_date”的职能。

EXPRESS 描述：

*)

```
FUNCTION cc_design_date_time_correlation
  (e:cc_design_date_and_time_assignment):BOOLEAN;
  LOCAL
    dt_role:STRING;
  END_LOCAL;
    dt_role:=e\time_assignment.role.name;
  CASE dt_role OF
    'creation_date'          :IF SIZEOF(e.items)<>
                                 SIZEOF(QUERY(x<* e.items|
                                 'CONFIG_CONTROL_DESIGN.'+
                                 'PRODUCT_DEFINITION'
                                 IN TYPEOF(x)))
                                 THEN RETURN(FALSE);
                             END_IF;
'request_date'               :IF SIZEOF(e.items)<>
                                 SIZEOF(QUERY(x<* e.items|
                                 SIZEOF(
                                 ['CONFIG_CONTROL_DESIGN.CHANGE_REQUEST'+
                                 'CONFIG_CONTROL_DESIGN.START_REQUEST']*
                                 TYPEOF(x))=1))
                                 THEN RETURN(FALSE);
                             END_IF;
'release_date'               :IF SIZEOF(e.items)<>
                                 SIZEOF(QUERY(x<* e.items|
```

```
                    SIZEOF(
                ['CONFIG_CONTROL_DESIGN.CHANGE'+
                    'CONFIG_CONTROL_DESIGN.START_WORK']*
                    TYPEOF(x))=1))
                    THEN RETURN(FALSE);
                END_IF;
'start_date'    :IF SIZEOF(e.items)<>
                    SIZEOF(QUERY(x<* e.items|
                    SIZEOF(
                    ['CONFIG_CONTROL_DESIGN.CHANGE'+
                    'CONFIG_CONTROL_DESIGN.START_WORK']*
                    TYPEOF(x))=1))
                    THEN RETURN(FALSE);
                END_IF;
'sign_off_date' :IF SIZEOF(e.items)<>
                    SIZEOF(QUERY(x<* e.items|
                    'CONFIG_CONTROL_DESIGN.'+
                    'APPROVAL_PERSON_ORGANIZATION'
                    IN TYPEOF(x)))
                    THEN RETURN(FALSE);
                END_IF;
'contract_date' :IF SIZEOF(e.items)<>
                    SIZEOF(QUERY(x<* e.items|
                    'CONFIG_CONTROL_DESIGN.CONTRACT'
                    IN TYPEOF(x)))
                    THEN RETURN(FALSE);
                END_IF;
'certification_date' :IF SIZEOF(e.items)<>
                    SIZEOF(QUERY(x<* e.items|
                    'CONFIG_CONTROL_DESIGN.CERTIFICATION'
                    IN TYPEOF(x)))
                    THEN RETURN(FALSE);
                END_IF;
'classification_date' :IF SIZEOF(e.items)<>
                    SIZEOF(QUERY(x<* e.items|
                    'CONFIG_CONTROL_DESIGN.'+
                    'SECURITY_CLASSIFICATION'
                    IN TYPEOF(x)))
                    THEN RETURN(FALSE);
                END_IF;
'declassification_date' :IF SIZEOF(e.items)<>
                    SIZEOF(QUERY(x<* e.items|
```

```
                                    'CONFIG_CONTROL_DESIGN.'+
                                    'SECURITY_CLASSIFICATION'
                                    IN TYPEOF(x)))
                                    THEN RETURN(FALSE);
                                END_IF;
    OTHERWISE:RETURN(TRUE);
  END_CASE;
  RETURN(TRUE);
END_FUNCTION;
(*
e:(输入)标识要检查的 cc_design_date_and_time_assignment 实体。
```

5.2.5.5 被定义的装配件形状

assembly_shape_is_defined(被定义的装配件形状)函数接受一个 next_assembly_usage_occurrence 实体作为输入并返回一个布尔结果。如果作为 next_assembly_usage_occurrence 实体的属性的 realted_product_definition 和 realting_product_definition 表示的 product_definition 实体定义了其形状，并且两个形状通过一个 shape_representation_relationship 实体建立联系，两个关系通过 context_dependent_shape_representation 实体建立联系，则该函数返回真值。如果 realted_product_definition 或 realting_product_definition 的形状没有定义，该函数也返回真值。

仅当形状为了 realted_product_definition 和 realting_product_definition 实体而被定义，并通过 shape_representation_relationship 实体建立联系。但 next_assembly_usage_occurrence 和 shape_representation_relationship 实体没有显式地通过 context_dependent_shape_representation 实体建立联系时，该函数才返回假值。

EXPRESS 描述：

```
*)
FUNCTION assembly_shape_is_defined
  (assy:next_assembly_usage_occurrence):BOOLEAN;
  LOCAL
        sdr_set:SET OF shape_definition_representation:=[];
        srr_set:SET OF shape_representation_relationship:=[];
END_LOCAL;
---- 收集 shape_definition_representation 的所有实例
  ---- component part 有一个为它定义的 representation。
sdr_set:=QUERY(pdr<*bag_to_set(USEDIN
            (assy.related_product_definition,
            'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION_REPRESENTATION.'+
            'DEFINITION.DEFINITION'))|
            'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION'IN
            TYPEOF(pdr))+
            QUERY(pdr<*bag_to_set(
            USEDIN(assy.related_product_definition,
            'CONFIG_CONTROL_DESIGN.'+
            'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION.'+
```

```
        'DEFINITION.RELATED_PRODUCT_DEFINITION')) |
            'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION'IN
            TYPEOF(pdr));
——如果有一个为了该构件而定义的 representation
IF SIZEOF(sdr_set)>0 THEN
  ——对该构件形状的每个表示，收集 shape_representation_relationship 的所有实例，该实例中的该构件的表示与其他表示有关。
  REPEAT i:=1 TO HIINDEX(sdr_set);
    srr_set:=QUERY(rr<* bag_to_set(
                USEDIN(sdr_set[i]\
                property_definition_representation.used_representation,
                'CONFIG_CONTROL_DESIGN.REPRESENTATION_RELATIONSHIP.REP_
              2')) |
              'CONFIG_CONTROL_DESIGN.SHAPE_REPRESENTATION_RELATIONSHIP'IN
                TYPEOF(rr));
  ——如果有一个 shape_representation relationship，其中的构件的 shape_representation 与其他的 shape_representation 有关。
  IF SIZEOF(srr_set)> 0THEN
    ——对该集合中的每个 shape_representation_relationship
    REPEAT j:=1 TO HIINDEX(srr_set);
      ——如果在 shape_representation_relationship 中的另一个 shape_representation 是装配件的 shape_representation，至少有一个 product_definition 的实例。
      IF SIZEOF(QUERY(pdr<* bag_to_set(USEDIN
        (srr_set[j]_relationship.rep_1,
        'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION_REPRESENTATION.'+
        'USED_REPRESENTATION')) |
        'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION'IN
        TYPEOF(pdr))*
        QUERY (pdr<* bag_to_set(
        USEDIN(assy.relating_product_definition,
        'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION_REPRESENTATION.'+
        'USED_REPRESENTATION.DEFINITION')) |
        'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION'
        IN TYPEOF(pdr)))>=1 THEN
        ——如果零件和装配件中的每个 shape_representation_relationship
        和 product_definition_relationship 不是通过 context_dependent_shape_representation 给出，那么返回 FALSE(假)。
        IF SIZEOF(QUERY(cdsr<*
          USEDIN(assy,'CONFIG_CONTROL_DESIGN.'+
          'CONTEXT_DEPENDENT_SHAPE_REPRESENTATION.'+
          'REPRESENTED_PRODUCT_RELATION') |
          NOT((cdsr.represented_product_relation.definition:=:assy)AND
```

```
            (cdsr.representation_relation:=:srr_set[j])))>0
            THEN RETURN(FALSE);
             END_IF;
           END_IF;
         END_REPEAT;
       END_IF;
     END_REPEAT;
  END_IF;
    -如果零件的形状没有规定或不非法,返回 TRUE(真)
  RETURN(TRUE);
END_FUNCTION;
(*
    变元定义:
    assy:(输入)标识 next_assembly_usage_occurrence 实体,要检查其关系。
```

5.2.5.6 各流形实体表示壳

该 msb_shells(各流形实体表示壳)函数建立用于一个 manifold_solid_brep 定义中所有 closed shell 实体的集合。

EXPRESS 描述:

```
*)
FUNCTION msb_shells
  (brep:manifold_solid_brep;
  schema_name:STRING):SET[1:?]OF closed_shell;
IF(schema_name+'.BREP_WITH_VOIDS')IN TYPEOF(brep)THEN
  RETURN(brep_with_voids.voids+brep.outer);
ELSE
  RETURN([brep.outer]);
END_IF;
END_FUNCTION;
(*
    变元定义:
    brep:(输入)标识一个 manifold_solid_brep 实体,该实体需要 closed_shell 构件的集合。
    schema_name:(输入)标识 brep 属于的模式名,其类型为 STRING(串)。
```

5.2.5.7 合法的线框边曲线

valid_wireframe_edge_curve(合法的线框边曲线)函数决定一个输入曲线是否合法地用来表示一个用拓扑有界线框定义的形状。

EXPRESS 描述:

```
*)
FUNCTION valid_wireframe_edge_curve
  (crv   :curve;schma:STRING):BOOLEAN;
    -检查合法的基本曲线类型
  IF SIZEOF([schma+'.LINE',
    schma +'.CIRCLE',
    schma +'.ELLIPSE,'
```

```
    schma +'.PARABOLA',
    schma +'.HYPERBOLA',
    schma +'.B_SPLINE_CURVE',
    schma +'.POLYLINE']* TYPEOF(crv))=1
    THEN RETURN(TRUE);
ELSE
  ——递归地检查 curve_replica 的合法父曲线
IF (schma +'.CURVE_REPLICA')IN TYPEOF(crv)
  THEN RETURN
  (valid_wireframe_edge_curve(crv_replica.parent_curve,schma));
  ELSE
    ——递归地检查 offset_curve 的合法基本曲线
      IF(schma +'.OFFSET_CURVE_3D')IN TYPEOF(crv)
        THEN RETURN
        (valid_wireframe_edge_curve(crv_curve_3d.basis_curve,schma));
    END_IF;
  END_IF;
 END_IF;
 RETURN(FALSE);
END_FUNCTION;
(*
    变元定义:
    crv: (输入)要检查的 curve。
    schma: (输入)定义模式的字符串,在该模式中,crv 定义用作类型检查。
```

5.2.5.8 合法的线框顶点点

valid_wireframe_vertex_point(合法的线框顶点点)函数决定一个输入点是否合法地用来表示一个用拓扑有界线框定义的形状。

EXPRESS 描述:

```
*)
FUNCTION valid_wireframe_vertex_point
  (pnt   :point;schma:STRING):BOOLEAN;

  ——检查合法的基本点类型
 IF (schma +'.CARTESIAN_POINT'IN TYPEOF(pnt))
    THEN RETURN(TRUE);
 ELSE
    ——递归地检查作为 point_replica 之父的合法基本点类型
   IF (schma +'.POINT_REPLICA')IN TYPEOF(pnt)
     THEN RETURN(valid_wireframe_vertex_point
     (pnt_replica.parent_pt,schma));
   END_IF;
 EDN_IF;
 RETURN(FALSE);
```

```
END_FUNCTION;
(*
```

变元定义：

pnt：（输入）要检查的 point。

schma：（输入）定义模式的字符串，在该模式中，用参数 pnt 定义作类型检查。

5.2.5.9 合法的几何有界线框曲线

valid_geometrically_bounded_wf_curve（合法的几何有界线框曲线）函数决定一个输入曲线是否合法地用来表示一个用几何有界线框定义的形状。

EXPRESS 描述：

```
*)
FUNCTION valid_geometrically_bounded_wf_curve
  (crv   :curve;schma:STRING):BOOLEAN;
    检查合法的基本曲线
  IF SIZEOF([schma +'.POLYLINE',
     schma +'.B_SPLINE_CURVE',
     schma +'.ELLIPSE',
     schma +'.CIRCLE']* TYPEOF(crv))=1
     THEN RETURN(TRUE);
  ELSE
     ——如果曲线是 trimmed_curve
    IF (schma +'.TRIMMED_CURVE')IN TYPEOF(crv)THEN
       ——如果 line、parabola 或 hyperbola 被裁剪，那么合法
      IF SIZEOF([schma +'.LINE',
         schma +'.PARABOLA',
         schma +'.HYPERBOLA']* TYPEOF(crv_curve.basis_curve))=1
         THEN RETURN(TRUE);
         ——否则，递归地检查 basis_curve
         ELSE RETURN(valid_geometrically_bounded_wf_curve
         (crv_curve.basis_curve,schma));
      END_IF;
  ELSE
     ——递归地检查 offset_curves 基本曲线
    IF (schma +'.OFFSET_CURVE_3D')IN TYPEOF(crv)
      THEN RETURN(valid_geometrically_bounded_wf_curve
      (crv_curve_3d.basis_curve,schma));
    ELSE
      ——递归地检查 curve_replicas 父曲线
    IF (schma +'.CURVE_REPLICA')IN TYPEOF(crv)THEN
      RETURN(valid_geometrically_bounded_wf_curve
      (crv_replica.parent_curve,schma));
    ELSE
     ——递归地检查 composite_curves 线段
        IF (schma +'.COMPOSITE_CURVE')IN TYPEOF(crv)THEN
```

```
            RETURN(SIZEOF(QUERY(ccs<* crv_curve.segments|
            NOT(valid_geometrically_bounded_wf_curve
            (ccs.parent_curve,schma))))
            =0);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;
(*
    变元定义:
    crv: (输入)要检查的 curve。
    schma: (输入)定义模式的字符串,在该模式中,用参数 crv 定义作类型检查。
```

5.2.5.10 合法的几何有界线框点

valid_geometrically_bounded_wf_point(合法的几何有界线框点)函数决定一个输入点是否合法地用来表示一个用几何有界线框定义的形状。

EXPRESS 描述:

```
*)
FUNCTION valid_geometrically_bounded_wf_point
  (pnt:point;schma:STRING):BOOLEAN;
——检查合法的基本类型
  IF (schma +'.CARTESIAN_POINT')IN TYPEOF(pnt)
    THEN RETURN(TRUE);
  ELSE
    ——如果输入类型是 point_on_curve,那么检查作为基础的合法几何边界曲线
    IF (schma +'.POINT_ON_CURVE')IN TYPEOF(pnt)
      THEN RETURN(valid_geometrically_bounded_wf_curve
      (pnt_on_curve.basis_curve,schma));
    ELSE
    ——if the input type is a point_replica check for a valid parent point
    IF(schma +'.POINT_REPLICA')IN TYPEOF(pnt)
    THEN RETURN(valid_geometrically_bounded_wf_point
        (pnt_replica.parent_pt,schma));
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;
(*
    变元定义:
    pnt: (输入)要检查的 point。
```

schma：（输入）定义模式的字符串，在该模式中，用参数 pnt 定义作类型检查。

5.2.5.11 几何有界曲面检查点

gbsf_check_point（几何有界曲面检查点）函数检查 geometrically_bounded_surface_shape_representation 范围内一个点实的合法性。如果该点是一个合法的实例，函数返回真值，否则函数返回假值。基本的合法点类型是 cartesian_point。点的其他类型包括引用在 geometrically_bounded_surface_shape_representation 范围内的几何类型的点都要检查其使用的一致性。该检查包括验证基本几何类型以及用点定义 curve 和 surface 的几何有界。

出现在由输入点开始的引用树中的 curve 和 surface 的所有实例也被检查。对树中出现的任何非法类型引用，该函数返回假值。

注

1 该函数不检查关于 GB/T 16656.42引用的正确性。只检查由于 geometrically_bounded_surface_shape_representation 范围所增加的要求。

2 该函数使用递归跟踪树型结构下层中的实体。

EXPRESS 描述：

```
*)
FUNCTION gbsf_check_point(pnt:point;schema_name:STRING):BOOLEAN;
  --检查该输入类型是否正确。
  IF NOT(schema_name +'.POINT'IN TYPEOF(pnt))THEN RETURN(TRUE);
  END_IF;
  IF schema_name +'.CARTESIAN_POINT'IN TYPEOF(pnt)THEN
    RETURN(TRUE);
  ELSE
      --point_on_curve 需要检查其曲线的合法性；
      --gbsf_check_curve 负责向下引用。
    IF schema_name +'.POINT_ON_CURVE'IN TYPEOF(pnt)THEN
      RETURN (gbsf_check_curve
      (pnt_on_curve.basis_curve,schema_name));
    ELSE
      --point_on_surface 需要检查其曲线的合法性；
        gbsf_check_surface 负责向下引用。
      IF schema_name +'.POINT_ON_SURFACE'IN TYPEOF(pnt)THEN
        RETURN(gbsf_check_surface
        (pnt_on_surface.basis_surface,schema_name));
    ELSE
      --degenerate_pcurve 需要检查其定义几何的合法性
      --gbsf_check_curve 和 gbsf_check_surface 负责向下引用。
      --当获得一个合法点时，两个函数都返回真值。
      IF schema_name +'.DEGENERATE_PCURVE' IN TYPEOF(pnt)THEN
        RETURN
        ((gbsf_check_curve
        (pnt_pcurve.reference_to_curve.items[1],
        schema_name))
        AND (gbsf_check_surface
```

```
        (pnt_pcurve.basis_surface,schema_name)));
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;
(*
```

变元定义：

pnt：（输入）point 要检查是否是 geometrically_bounded_surface_shape_representation 范围内的合法点。

schma：（输入）模式名，在该模式中，检查点。

5.2.5.12 几何有界曲面检查曲线

gbsf_check_curve（几何有界曲面检查曲线）函数检查 geometrically_bounded_surface_shape_representation 范围内一个曲线实例的合法性。如果该曲线是一个合法的实例，函数返回真值，否则函数返回假值。基本的合法曲线类型是 circle，ellipse 和 b_spline_curve。曲线的其他类型包括引用在 geometrically_bounded_surface_shape_representation 范围内的几何类型的曲线都要检查其使用的一致性。该检查包括验证基本几何类型以及用曲线定义的 curve 和 surface 的几何界定。

出现在由输入曲线开始的引用树中的 curve 和 surface 的所有实例也检查。对树中出现的任何非法类型引用，该函数返回假值。

注

1 该函数不检查关于 GB/T 16656.42引用的正确性。只检查由于 geometrically_bounded_surface_shape_representation 范围所增加的要求。

2 该函数使用递归跟踪树型结构下层中的实体。

EXPRESS 描述：

```
*)
FUNCTION gbsf_check_curve
  (cv:curve;schema_name:STRING):BOOLEAN;
  ---检查输入类型是否正确
  IF NOT(schema_name +'.CURVE'IN TYPEOF(cv))THEN RETURN(TRUE);
  END_IF;
  ----让这些类型通过，不引用任何其他规则

  IF SIZEOF([schema_name +'.CIRCLE',schema_name +'.ELLIPSE']
    *TYPEOF(cv))=1 THEN RETURN(TRUE);
  ELSE
    ---b_spline_curve 不应自交
    IF (schema_name +'.B_SPLINE_CURVE' IN TYPEOF(cv))
      THEN RETURN(NOT(cv_spline_curve.self_intersect));
    ELSE
      ---如果曲线是复合曲线，它的所有线段都应合法
      IF schema_name +'.COMPOSITE_CURVE' IN TYPEOF(cv)THEN
        RETURN(SIZEOF(QUERY(seg<* cv_curve.segments|
```

```
    NOT(gbsf_check_curve(seg.parent_curve,schema_name))))=0);
  ELSE
    curve_replica 引用其他需要检查的曲线
  IF schema_name +'.CURVE_REPLICA'IN TYPEOF(cv)THEN
    RETURN(gbsf_check_curve
    (cv_replica.parent_curve,schema_name));
  ELSE
    --offset_curve_3d 引用曲线并且不应自交
    IF schema_name +'.OFFSET_CURVE_3D'IN TYPEOF(cv)THEN
      RETURN((gbsf_check_curve
      (cv_curve_3d.basis_curve,schema_name))
      AND
      (NOT cv_curve_3d.self_intersect));
    ELSE
      --pcurve 引用一个曲线-间接的,以及一个 basis_surface
  IF schema_name +'.PCURVE'IN TYPEOF(cv)THEN
    RETURN
    ((gbsf_check_curve
    (cv.reference_to_curve.items[1],
    schema_name))AND
    (gbsf_check_surface
    (cv.basis_surface,schema_name)));
  ELSE
      --polyline 至少应有3个点,并且只能被用来表示
      --intersection_curve
    IF schema_name +'.POLYLINE'IN TYPEOF(cv)THEN
      IF(SIZEOF(cv.points)>3) AND
        (SIZEOF((bag_to_set(USEDIN(cv,''))-
        bag_to_set(USEDIN(cv,
        schema_name +
        '.INTERSECTION_CURVE.BASIS_CURVE'))))=0)
        THEN RETURN(TRUE);
      END_IF;
    ELSE
      --surface_curve 引用 curve_3d 以及一个或两个
        pcurves 或一个或两个 surface_curves 或每一个
      IF schema_name +'.SURFACE_CURVE'IN TYPEOF(cv)THEN
        --如果曲线引用正确,也检查其他的
        IF gbsf_check_curve
        (cv_curve.curve_3d,schema_name)THEN
          REPEAT i:=1TO SIZEOF
          (cv_curve.associated_geometry);
          --为一个或两个 associated_geometrys;
```

```
            IF schema_name +'.SURFACE'IN TYPEOF
              (cv_curve.associated_geometry[i]THEN
              IF NOT gbsf_check_surface
                (cv_curve.associated_geometry[i],
                schema_name)THEN
                RETURN(FALSE);
            END_IF;
          ELSE
            IF schema_name +'.PCURVE'IN TYPEOF
              (cv_curve.associated_geometry[i]THEN
            IF NOT gbsf_check_curve
              (cv_curve.associated_geometry[i],
              schema_name)THEN
              RETURN(FALSE);
            END_IF;
          END_IF;
        END_IF;
      END_REPEAT;
      RETURN(TRUE);
    END_IF;
  ELSE
          ——如果曲线是一条 trimmed_curve
          IF(schema_name +'.TRIMMED_CURVE')IN TYPEOF(cv)THEN
          ——如果 line,parabola 或 hyperbola 被裁剪,
          ——那么合法
            IF SIZEOF([schema_name +'.LINE',
            schema_name +'.PARABOLA',
            schema_name +'.HYPERBOLA']*
            TYPEOF(cv_curve.basis_curve))=1
            THEN RETURN(TRUE);
              ——否则,递归地检查基本曲线
              ELSE RETURN (gbsf_check_curve
                (cv_curve.basis_curve,schema_name));
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
END_IF;
```

```
  RETURN(FALSE);
END_FUNCTION;
(*
```

变元定义：

cv：（输入）curve 要检查是否是 geometrically_bounded_surface_shape_representation 范围内的合法曲线。

schma：（输入）模式名，在该模式中，检查曲线。

5.2.5.13 几何有界曲面检查曲面

gbsf_check_surface（几何有界曲面检查曲面）函数检查 geometrically_bounded_surface_shape_representation 范围内一个曲面实例的合法性。如果该曲面是一个合法的实例，函数返回真值，否则函数返回假值。基本的合法曲面类型是 b_spline_surface，spherical_surface 和 toroidal_surface。曲面的其他类型包括引用在 geometrically_bounded_surface_shape_representation 范围内的几何类型的曲面都要检查其使用的一致性。该检查包括验证基本几何类型以及用曲面定义的 curve 和 surface 的几何界定。

出现在由输入曲面开始的引用树中的 curve 和 surface 的所有实例也检查。对树中出现的任何非法类型引用，该函数返回假值。

注

1 该函数不检查关于 GB/T 16656.42引用的正确性。只检查由于 geometrically_bounded_surface_shape_representation 范围所增加的要求。

2 该函数使用递归跟踪树型结构下层中的实体。

EXPRESS 描述：

```
*)
FUNCTION gbsf_check_surface(sf:surface;schema_name:STRING):BOOLEAN;
      检查输入类型是否正确
  IF NOT(schema_name +'.SURFACE')IN TYPEOF(sf)THEN RETURN(TRUE);
  END_IF;
      b_spline_surface 有一个为假的 self_intersect 属性
  IF schema_name +'.B_SPLINE_SURFACE'IN TYPEOF(st)THEN
    RETURN(NOT(sf_spline_surface.self_intersect));
  ELSE
        基本曲面类型返回真值
    IF SIZEOF([schema_name +'.SPHERICAL_SURFACE',
      schema_name +'.TOROIDAL_SURFACE']* TYPEOF(sf))=3 THEN
      RETURN(TRUE);
ELSE
    IF schema_name +'.CURVE_BOUNDED_SURFACE' IN TYPEOF(sf)THEN
        如果有一个简单的基本曲面，检查曲线
      IF SIZEOF([schema_name +'.CONICAL_SURFACE',
        schema_name +'.CYLINDRICAL_SURFACE',
        schema_name +'.PLANE']* TYPEOF(sf))=1 THEN
        RETURN(SIZEOF(QUERY(bcurve<*
        sf_bounded_surface.boundaries|
        NOT(gbsf_check_curve(bcurve,schema_name))))=0);
      ELSE
```

```
      ——递归地检查基本曲面，然后检查曲线
      IF gbsf_check_surface
      (sf_bounded_surface.basis_surface,schema_name)THEN
        RETURN(SIZEOF(QUERY(bcurve<*
        sf_bounded_surface.boundaries|
        NOT(gbsf_check_curve(bcurve,schema_name))))=0);
    END_IF;
  END_IF;
ELSE
  ——offset_surface 引用一个曲面并且不自交
  IF schema_name +'.OFFSET_SURFACE'IN TYPEOF(sf)THEN
    RETURN((gbsf_check_surface
    (sf_surface.basis_surface,schema_name))
    AND
    (NOT(sf_surface.self_intersect)));
ELSE
——rectangular_composite_surface 引用一个曲面矩阵
IF schema_name +'.RECTANGULAR_COMPOSITE_SURFACE'IN TYPEOF(sf)THEN
REPEAT i:=1 TO SIZEOF
  (sf_composite_surface.segments);
  REPEAT j:=1 TO SIZEOF
    (sf_composite_surface.segments[i]);
    IF NOT(gbsf_check_surface
      (sf_composite_surface.segments[i][j].
      parent_surface,schema_name))
      THEN RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_REPEAT;
  RETURN(TRUE);
  ELSE
  ——rectangular_trimmed_surface 有一个基本曲面
  IF schema_name +'.RECTANGULAR_TRIMMED_SURFACE'IN TYPEOF(sf)THEN
    IF SIZEOF([schema_name +'.CONICAL_SURFACE',
      schema_name +'.CYLINDRICAL_SURFACE',
      schema_name +'.PLANE']*TYPEOF
                (sf_trimmed_surface.basis_surface))=1 THEN
                RETURN(TRUE);
              ELSE
                RETURN(gbsf_check_surface
                (sf_trimmed_surface.basis_surface,schema_name));
              END_IF;
            ELSE
```

```
                --parent_surfaces 需要检查其合法性
                IF schema_name+'.SURFACE_REPLICA'IN TYPEOF (sf) THEN
                  RETURN(gbsf_check_surface
                  (sf_replica.parent_surface,schema_name));
                ELSE
                --并且扫成曲面必须有一条扫描曲线
                IF schema_name+'.SWEPT_SURFACE'IN TYPEOR(sf)THEN
                   RETURN(gbsf_check_curve
                   (sf_surface.swept_curve,schema_name));
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;
(*
```

变元定义：

sf：(输入)surface 要检查是否是 geometrically_bounded_surface_shape_representation 范围内的合法曲面。

schema_name：(输入)模式名，在该模式中，检查曲面。

5.2.5.14 基本曲线检查

basis_curve_check(基本曲线检查)函数检查各种曲线是否引用 manifold_surface_shape_representation 内的合法基本曲线。

检查下列类型：

——curve_replica(复制曲线)；

——offset_curve_3d(三维偏置曲线)；

——pcurve(参数域曲线)；

——surface_curve(曲面曲线)；

在适当情况下，此四个类型之一的一个实例可以被递归检查。这意味着如果一个基本曲线引用一个实体类型，而该实体类型引用它父实例的一个实体类型，则调用该函数。

一个 curve_replica 的 parent_curve 可以是下列类型之一：

——b_spline_curve(B 样条曲线)；

——conic(圆锥曲线)；

——curve_replica(曲线复制)(递归)；

——line(直线)；

——offset_curve_3d(三维偏置曲线)(递归)；

——pcurve(参数域曲线)(递归)；

——polyline(折线)；

——surface_curve(曲面曲线)；

一个 offset_curve_3d 的 basis_curve 可以是下列类型之一：

——b_spline_curve(B 样条曲线)；

——conic(圆锥曲线)；

——curve_replica(曲线复制)(递归)；

——line(直线)；

——offset_curve_3d(三维偏置曲线)(递归)；

——pcurve(参数域曲线)(递归)；

——polyline(折线)；

——surface_curve(曲面曲线)；

被一个 pcurve 引用作为 reference_to_curve 的 definitional_representation 的 items 集合中的一个实例可以是下列类型之一：

——b_spline_curve(B 样条曲线)；

——conic(圆锥曲线)；

——curve_replica(曲线复制)(递归)；

——line(直线)；

——polyline(折线)；

一个 surface_curve 的 curve_3d 可以是下列类型之一：

——b_spline_curve(B 样条曲线)；

——conic(圆锥曲线)；

——curve_replica(曲线复制)(递归)；

——line(直线)；

——offset_curve_3d(三维偏置曲线)(递归)；

——polyline(折线)；

如果所有潜在的递归引用的基本曲线类型是允许的类型，该函数返回逻辑真值。不引用任何其他曲线的曲线也使真值返回。

EXPRESS 描述：

```
*)
FUNCTION basis_curve_check
    (cv:curve;schema_name:STRING):BOOLEAN;
    ——让那些不引用曲线的类型通过
    IF SIZEOF([schema_name+'.B_SPLINE_CURVE',
       schema_name+'.CONIC',
       schema_name+'.LINE',
       schema_name+'.POLYLINE']* TYPEOF(cv))=1THEN
       RETURN(TRUE);
    ELSE
    ——在 curve_replica 的情况下，检查输入曲线的类型
    IF schema_name+'.CURVE_REPLICA'IN TYPEOF(cv)THEN
       RETURN (basis_curve_check(cv_replica.parent_curve,schema_name));
    ELSE
    ——在 offset_curve_3d 的情况下，检查输入曲线的类型
    IF schema_name+'.OFFSET_CURVE_3D'IN TYPEOF(cv)THEN
       RETURN(basis_curve_check
```

```
        (cv_curve_3d.basis_curve,schema_name));
    ELSE
      ----在 pcurve 的情况下,检查输入曲线的类型
    IF schema_name+'.PCURVE'IN TYPEOF(cv)THEN
    IF SIZEOF([schema_name+'.B_SPLINE_CURVE',
      schema_name+'.CONIC',
      schema_name+'.LINE',
      schema_name+'.POLYLINE']*
      TYPEOF(cv.reference_to_curve.items[1]))
      =1 THEN
      RETURN(basis_surface_check(cv.basis_surface,
      schema_name));
    ELSE
      IF schema_name+'.CURVE_REPLICA'
        IN TYPEOF
        (cv.reference_to_curve.items[1])THEN
        RETURN((basis_curve_check
        (cv.reference_to_curve.items[1],
        schema_name))AND
        (basis_surface_check(cv.basis_surface,schema_name)));
       END_IF;
    END_IF;
    ELSE
      ----在 surface_curve 的情况下,检查输入曲线的类型
IF schema_name+'.SURFACE_CURVE'IN TYPEOF(cv)THEN
    IF SIZEOF ([schema_name+'.B_SPLINE_CURVE',
      schema_name+'.CONIC',
      schema_name+'.LINE',
      schema_name+'.POLYLINE']*
      TYPEOF(cv_curve.curve_3d))=1 THEN
      REPEATi:=1 TO SIZEOF(cv_curve.associated_geometry);
        --处理一个或两个 associated_geometrys
      IF schema_name+'.SURFACE'IN
        TYPEOF(cv_curve.associated_geometry[i])THEN
        IF NOT basis_surface_check
          (cv_curve.associated_geometry[i],
          schema_name)THEN
          RETURN(FALSE);
        END_IF;
      ELSE
        IF schema_name+'.PCURVE'IN
          TYPEOF(cv_curve.associated_geometry[i])THEN
          IF NOT basis_curve_check
```

```
(cv_curve.associated_geometry[i],
              schema_name)THEN RETURN(FALSE);
            END_IF;
          END_IF;
        END_IF;
      END_REPEAT;
    ELSE
      IF schema_name+'.CURVE_REPLICA'IN TYPEOF
        (cv_curve.curve_3d)THEN
        IF NOT (basis_curve_check
          (cv_replica.parent_curve,schema_name))THEN
          RETURN(FALSE);
      END_IF;
    ELSE
      IF schema_name+'.OFFSET_CURVE_?D'IN TYPEOF
        (cv_curve.curve_3d)THEN
        IF NOT (basis_curve_check
          (cv_curve_3d.basis_curve,schema_name))THEN
          RETURN(FALSE);
      END_IF;
    ELSE
      REPEATi:=1 TO SIZEOF
      (cv_curve.associated_geometry);
      ——处理一个或两个 associated-geometrys
                IF schema_name+'.SURFACE'IN
                  TYPEOF(cv_curve.associated_geometry[i])THEN
                  IF NOT basis_surface_check
                    (cv_curve.associated_geometry[i],
                    schema_name)THEN
                    RETURN(FALSE);
                  END_IF;
                ELSE
                  IF schema_name+'.PCURVE'IN
                    TYPEOF(cv_curve.associated_geometry[i])THEN
                    IF NOT basis_curve_check
                      (cv_curve.associated_geometry[i],
                      schema_name)THEN
                      RETURN(FALSE);
                    END_IF;
                  END_IF;
                END_IF;
              END_REPEAT;
              END_IF;
```

```
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  RETURN(FALSE);
END_FUNCTION;
(*
```

变元定义：

cv：(输入)curve 要检查是否是 manifold_surface_shape_representation 内的合法曲线。

schema_name：(输入)模式名，在该模式中，定义曲线。

5.2.5.15 基本曲面检查

basis_surface_check(基本曲面检查)函数检查三种类型曲面是否引用合法的基本曲面。

检查下列类型：

——offset_surface(偏置曲面)；

——swept_surface(扫成曲面)；

——surface_replica(曲面复制)；

在适当情况下，此二类型之一的一个实例可以递归检查。这意味着如果一个基本曲面引用一个实体类型，而该实体类型引用它父实例的一个实体类型，则调用该函数。

一个 surface_replica 的 parent_surface 以及一个 offset_surface 的 basis_surface 可以是下列类型之一：

——b_spline_surface(B 样条曲面)；

——elementary_surface(元素曲面)；

——offset_surface(偏置曲面)(递归)；

——surface_replica(曲面复制)(递归)；

——swept_surface(扫成曲面)；

如果所有潜在的递归引用的基本曲面类型是允许的类型，该函数返回逻辑真值。不引用任何其他曲面的曲面也使真值返回。

EXPRESS 描述：

```
*)
FUNCTION basis_surface_check
    (surf:surface;schema_name:STRING):BOOLEAN;
        ——让那些不引用曲面的曲面类型通过
    IF SIZEOF([schema_name+'.B_SPLINE_SURFACE',
      schema_name+'.ELEMENTARY_SURFACE']*TYPEOF(surf)=1 THEN
      RETURN(TRUE);
    ELSE
    IF schema_name+'.SWEPT_SURFACE'IN TYPEOF(surf)THEN
      RETURN(basis_curve_check
      (surf_surface.swept_curve,schema_name));
    ELSE
```

```
    ——在 offset_surface 的情况下检查输入曲面的类型
      IF schema_name+'.OFFSET_SURFACE'IN TYPEOF(surf)THEN
        RETURN(basis_surface_check
        (surf_surface.basis_surface,schema_name));
    ELSE
    ——在 surface_replica 的情况下检查输入曲面的类型
      IF schema_name+'.SURFACE REPLICA'IN TYPEOF(surf)THEN
        RETURN(basis_surface_check
        (surf_replica.parent_surface,schema_name));
      END_IF;
    END_IF;
  END_IF;
  PETURN(FALSE);
END_FUNCTION;
(*
    变元定义:
    surf:(输入)surface 要检查是否是 manifold_surface_shape_representation 内的合法曲面。
    schema_name:(输入)模式名,在该模式中,定义曲面。
```

5.2.5.16 维数

函数 dimension_of(维数)返回 geometric_representation_context 的整数值 dimension_count,在 geometric_representation_context 中,输入的 geometric_representation_item 是以几何形式建立的。

通过全局规则 compatible_dimension 中的虚约束,该值是输入 geometric_representation_item 的 coordinate_space_dimension。

EXPRESS 描述:

```
*)
FUNCTION dimension_of
    (item:geometric_representation_item):
    dimension_count;
    LOCAL
      x:SET OF representation;
      y:representation_context;
    END_LOCAL;
    ——找出使用项的表示集合
    x:=using_representations(item);
    ——如果有这样一个表示,那么确定相关的 geometric_representation_context 的维数—注意到规则
compatible_dimension 保证 context_of_items 是 geometric_representation_context 类型,并且对所有的 x
有同样的维数。
    IF SIZEOF(x)>0 THEN
      y:=x[1].context_of_items;
      RETURN(y_representation_context.coordinate_space_dimension);
    END_IF;
END_FUNCTION;
```

```
(*
    变元定义：
    item：通过 geometric_representation_item 决定 dimension_count 的值。它是函数的输入。
```

5.2.5.17 利用表示

using_representations(利用表示)函数返回 representations 的集合，在该表示中使用一个 representation_item。

如果它：

a) 在该 representation 的各 item 集合中引用；或

b) 由在 representation 中使用的一个 representation_item 所引用，则在一个 representation 中要使用一个 representation_item。

注：第二个条件是一个递归检查，它允许一个 representation_item 作为相关的 representation_item 树的一部分使用在一个 representation 中。通过实现第一个条件，该树的根位于一个 representation 实体中。

EXPRESS 描述：

```
*)
FUNCTION using_representations
    (item:representation_item)
    :SET OF representation;
    LOCAL
      results                    :SET OF representation;
      result_bag                 :BAG OF representation;
      intermediate_items         :SET OF representation_item;
      i                          :INTEGER;
    END_LOCAL;
    --找出使用项的表示，并把它加入到结果集合中
    result_bag:=USEDIN(item,'CONFIG_CONTROL_DESIGN.REPRESENTATION.ITEMS');
    IF SIZEOF(result_bag)>0 THEN
      REPEAT i:=1 TO HIINDEX(result_bag);
        results:=results+result_bag[i];
      END_REPEAT;
    END_IF;
    --找出使用项的表示项的集合
    intermediate_items:=QUERY(z<*bag_to_set(USEDIN(item,''))|
      'CONFIG_CONTROL_DESIGN.REPRESENTATION_ITEM'IN TYPEOF(z));
    --如果中间项的集合不空
    IF SIZEOF (intermediate_items)>0 THEN
    --对集合中的每个元素，递归地增加那个元素的 using_representations.
    REPEAT i:=1 TO HIINDEX(intermediate_items);
      results:=results+using_representations(intermediate_items[i]);
    END_REPEAT;
  END_IF;
    --返回直接或间接使用输入项(通过插入表示项)的表示集合
      RETURN(results);
    END_FUNCTION;
```

(*

变元定义：

item：通过 representation_item 决定 representation 的使用，它是函数的输入。

EXPRESS 描述：

*)

END_SCHEMA;--config_control_design

(*

6 一致性要求

本标准的一致性包括满足此部分中被表明的要求，所支持的实现方法的要求，以及标准引用的相关要求。

一个实现应至少支持下列实现方法之一：GB/T 16656.21。在附录 D 中规定了关于实现方法的要求。

协议实现一致性声明形式列举了在此实现中可能包括的任选项或任选项的组合项。在附录 C 中提供了此协议实现一致性声明形式。

本标准提供了一实现可以支持的任选项号码。这些任选项分组组成了各一致性级别。定义了六个一致性级别。本标准的一致性至少要求级别1的一致性。通过级别2到级别6定义了任选项，并且可通过某一实现选用任选项。一特定一致性级别的一致性要求支持所有的 AIM 实体、类型以及被定义为该级别一部分的相关约束。一特定一致性级别的支持要求支持此级别中所有规定的任选项。

各一致性级别如下：

——级别1：除了形状之外的配置控制设计信息；

——级别2：级别1和几何边界线框模型、曲面模型，或者两者所表示的形状；

——级别3：级别1和拓扑线框模型表示的形状；

——级别4：级别1和拓扑流形曲面模型表示的形状；

——级别5：级别1和小平面边界表示所表示的形状；

——级别6：级别1和高级的边界表示所表示的形状。

级别1是级别2到级别6的先决条件。如果一实现与级别2到级别6的任一个相一致，那么也将与级别1相一致。

一致性级别2到级别6根据表2列出的 AIM shape_representation 子类型所要求的术语进行定义。每个 shape_representation 子类型在其局部定义域规则中规定了定义相应一致性级别的所有 AIM 结构。

表 15 一致性选择

形状表示子类型	级别2	级别3	级别4	级别5	级别6
geometrically_bounded_surface_shape_representation	X				
geometrically_bounded_wireframe_shape_representation	X				
edge_based_wireframe_shape_representation		X			
shell_based_wireframe_shape_representation		X			
manifold_surface_shape_representation			X		
facetted_brep_shape_representation				X	
advanced_brep_shape_representation					X

6.1 一致性级别1的实体

本标准一致性级别1的一个实现将支持下列实体和有关结构：

--action(活动)
--action_assignment(活动赋值)
--action_directive(活动命令)
--action_method(活动方法)
--action_request_assignment(活动请求赋值)
--action_request_solution(活动请求方案)
--action_request_status(活动请求状态)
--action_status(活动状态)
--address(地址)
--alternate_product_relationship(备用产品关系)
--application_context(应用相关环境)
--application_context_element(应用相关环境元素)
--application_protocol_definition(应用协议定义)
--approval(批准)
--approval_assignment(批准赋值)
--approval_date_time(批准日期时间)
--approval_person_organization(批准人员组织)
--approval_relationship(批准关系)
--approval_role(批准职能)
--approval_status(批准状态)
--area_measure_with_unit(面积度量)
--area_unit(面积单位)
--assembly_component_usage(装配构件使用)
--assembly_component_usage_substitute(装配构件使用代用)
--calender_date(日历日期)
--cc_desing_approval(配置控制设计批准)
--cc_desing_certification(配置控制设计认证)
--cc_desing_contract(配置控制设计合同)
--cc_desing_date_and_time_assignment(配置控制设计日期和时间赋值)
--cc_desing_person_and_organization_assignment(配置控制设计人员和组织赋值)
--cc_desing_security_classification(配置控制设计安全分类)
--cc_desing_specification_reference(配置控制设计规范引用)
--certification(认证)
--certification_assignment(认证赋值)
--certification_type(认证类型)
--change(更改)
--change_request(更改请求)
--configuration_design(配置设计)
--configuration_effectivity(配置有效性)
--configuration_item(配置项)
--context_dependent_unit(相关环境依赖单位)
--contract(合同)
--contract_assignment(合同赋值)

--contract_type(合同类型)
--conversion_based_unit(换算单位)
--coordinated_universal_time_offset(协调的世界时差)
--date(日期)
--date_and_time(日期和时间)
--date_and_time_assignment(日期和时间赋值)
--date_time_role(日期时间职能)
--dated_effectivity(日期有效性)
--design_context(设计相关环境)
--design_make_from_relationship(设计参照关系)
--dimensional_exponents(量纲指数)
--directed_action(指定活动)
--document(文档)
--document_reference(文档引用)
--document_relationship(文档关系)
--document_type(文档类型)
--document_usage_constraint(文档使用约束)
--document_with_class(有级别文档)
--effectivity(有效性)
--executed_action(完成的活动)
--length_measure_with_unit(长度度量)
--length_unit(长度单位)
--local_time(当地时间)
--lot_effectivity(批量有效性)
--mass_measure_with_unit(质量度量)
--mass_unit(质量单位)
--measure_with_unit(度量)
--mechanical_context(机械相关环境)
--named_unit(命名单位)
--next_assembly_usage_occurance(下一层装配活动)
--ordinal_date(顺序日期)
--organization(组织)
--organization_relationship(组织关系)
--organizational_address(组织的地址)
--organization_project(组织项目)
--person(人员)
--person_and_organization(人员和组织)
--person_and_organization_assignment(人员和组织赋值)
--person_and_organization_role(人员和组织职能)
--personal_address(人员的地址)
--product(产品)
--product_category(产品目录)
--product_category_relationship(产品目录关系)

--product_concept(产品概念)
--product_concept_context(产品概念相关环境)
--product_context(产品相关环境)
--product_definition(产品定义)
--product_definition_context(产品定义相关环境)
--product_definition_effectivity(产品定义有效性)
--product_definition_formation(产品定义形成)
--product_definition_formation_with_specified_source(有规定源的产品定义形成)
--product_definition_relationship(产品定义关系)
--product_definition_shape(产品定义形状)
--product_definition_usage(产品定义使用)
--product_definition_with_associated_documents(有相关文档的产品定义)
--product_related_product_category(与产品目录有关的产品)
--promissory_usage_occurance(约定的活动)
--property_definition(特性定义)
--quantified_assembly_component_usage(合格的装配构件使用)
--security_classification(安全分类)
--security_classification_assignment(安全分类赋值)
--security_classification_level(安全分类等级)
--serial_numbered_effectivity(连续编号有效性)
--si_unit(公制单位);
--specified_higher_usage_occurrence(规定的更高层活动)
--start_request(开始请求)
--start_work(开始工作)
--supplied_part_relationship(外购零件关系)
--uncertainty_measure_with_unit(误差度量)
--versioned_action_request(该版活动请求)
--volume_measure_with_unit(体积度量)
--volume_unit(体积单位)
--week_of_year_and_day_date(年的星期和日期)

6.2 一致性级别2的实体

本标准一致性级别2的一个实现将支持下列实体和有关结构:

--action(活动)
--action_assignment(活动赋值)
--action_directive(活动命令)
--action_method(活动方法)
--action_request_assignment(活动请求赋值)
--action_request_solution(活动请求方案)
--action_request_status(活动请求状态)
--action_status(活动状态)
--address(地址)
--alternate_product_relationship(备用产品关系)
--application_context(应用相关环境)

--application_context_element(应用相关环境元素)
--application_protocol_definition(应用协议定义)
--approval(批准)
--approval_assignment(批准赋值)
--approval_date_time(批准日期)
--approval_person_organization(批准人员组织)
--approval_relationship(批准关系)
--approval_role(批准职能)
--approval_status(批准状态)
--area_measure_with_unit(面积度量)
--area_unit(面积单位)
--assembly_component_usage(装配构件使用)
--assembly_component_usage_substitute(装配构件使用代用)
--axis 1_placement(轴 1 方位)
--axis 2_placement_2d(二维空间轴 2 方位)
--axis 2_placement_3d(三维空间轴 2 方位)
--b_spline_curve(B 样条曲线)
--b_spline_curve_with_knots(有结点的 B 样条曲线)
--b_spline_surface(B 样条曲面)
--b_spline_surface_with_knots(有结点的 B 样条曲面)
--bezier_curve(Bezier 曲线)
--bezier_suface(Bezier 曲面)
--boundary_curve(边界曲线)
--bounded_curve(有界曲线)
--bounded_surface(有界曲面)
--calendar_date(日历日期)
--cartesian_point(笛卡尔点)
--cartesian_transformation_operator(笛卡尔变换运算器)
--cartesian_transformation_operator_2d(二维笛卡尔变换运算器)
--cartesian_transformation_operator_3d(三维笛卡尔变换运算器)
--cc_desing_approval(配置控制设计批准)
--cc_desing_certification(配置控制设计认证)
--cc_desing_contract(配置控制设计合同)
--cc_desing_date_and_time_assignment(配置控制设计日期和时间赋值)
--cc_desing_person_and_organization_assignment(配置控制设计人员和组织赋值)
--cc_desing_security_classification(配置控制设计安全分类)
--cc_desing_specification_reference(配置控制设计规范引用)
--certification(认证)
--certification_assignment(认证赋值)
--certification_type(认证类型)
--change(更改)
--change_request(更改请求)
--circle(圆)

--composite_curve(复合曲线)
--composite_curve_on_surface(曲面上的复合曲线)
--composite_curve_segment(复合曲线段)
--configuration_design(配置设计)
--configuration_effectivity(配置有效性)
--configuration_item(配置项)
--conic(圆锥曲线)
--conical_surface(圆锥面)
--context_dependent_shape_representation(相关环境依赖形状表示)
--context_dependent_unit(相关环境依赖单位)
--contract(合同)
--contract_assignment(合同赋值)
--contract_type(合同类型)
--conversion_based_unit(换算单位)
--coordinated_universal_time_offset(协调的世界时差)
--curve(曲线)
--curve_bounded_surface(曲线界定的曲面)
--curve_replica(曲线复制)
--cylindrical_surface(圆柱面)
--date(日期)
--date_and_time(日期和时间)
--date_and_time_assignment(日期和时间赋值)
--date_time_role(日期时间职能)
--dated_effectivity(日期有效性)
--definitional_representation(定义的表达)
--degenerate_pcurve(退化参数域曲线)
--design_context(设计相关环境)
--design_make_from_relationship(设计参照关系)
--dimensional_exponents(量纲指数)
--directed_action(指定活动)
--direction(方向)
--document(文档)
--document_reference(文档引用)
--document_relationship(文档关系)
--document_type(文档类型)
--document_usage_constraint(文档使用约束)
--document_with_class(有级别文档)
--effectivity(有效性)
--elementary_surface(初等曲面)
--ellipse(椭圆)
--evaluated_degenerate_pcurve(赋值的退化参数域曲线)
--executed_action(完成的活动)
--functionally_defined_transformation(功能上定义的变换)

--geometric_curve_set(几何曲线集合)
--geometric_representation_context(几何表示相关环境)
--geometric_representation_item(几何表示项)
--geometric_set(几何集合)
--geometrically_bounded_surface_shape_representation(几何上有界曲面形状表示)
--geometrically_bounded_wireframe_shape_representation(几何上有界线框形状表示)
--global_uncertainty_assigned_context(全局带不确定性的相关环境)
--global_unit_assigned_context(全局单位相关环境)
--hyperbola(双曲线)
--intersection_curve(交线)
--item_defined_transformation(项定义变换)
--length_measure_with_unit(长度　度量)
--length_unit(长度单位)
--line(直线)
--local_time(当地时间)
--lot_effectivity(批量有效性)
--mapped_item(映射项)
--mass_measure_with_unit(质量度量)
--mass_unit(质量单位)
--measure_with_unit(度量)
--mechanical_context(机械相关环境)
--named_unit(命名单位)
--next_assembly_usage_occurance(下一层装配活动)
--offset_curve_2d(二维偏置曲线)
--offset_curve_3d(三维偏置曲线)
--offset_surface(偏置曲面)
--ordinal_date(顺序日期)
--organization(组织)
--organization_relationship(组织关系)
--organizational_address(组织的地址)
--organization_project(组织项目)
--parabola(抛物线)
--parametric_representation_context(参数表示相关环境)
--pcurve(参数域曲线)
--person(人员)
--person_and_organization(人员和组织)
--person_and_organization_assignment(人员和组织赋值)
--person_and_organization_role(人员和组织职能)
--personal_address(人员的地址)
--placement(定位)
--plane(平面)
--plane_angle_measure_with_unit(平面角度度量)
--plane_angle_unit(平面角度单位)

--point(点)
--point_on_curve(曲线上的点)
--point_on_surface(曲面上的点)
--point_replica(点复制)
--polyline(折线)
--product(产品)
--product_category(产品目录)
--product_category_relationship(产品目录关系)
--product_concept(产品概念)
--product_concept_context(产品概念相关环境)
--product_context(产品相关环境)
--product_definition(产品定义)
--product_definition_context(产品定义相关环境)
--product_definition_effectivity(产品定义有效性)
--product_definition_formation(产品定义形成)
--product_definition_formation_with_specified_source(有规定源的产品定义形成)
--product_definition_relationship(产品定义关系)
--product_definition_shape(产品定义形状)
--product_definition_usage(产品定义使用)
--product_definition_with_associated_documents(有相关文档的产品定义)
--product_related_product_category(与产品目录有关的产品)
--promissory_usage_occurance(约定的活动)
--property_definition(特性定义)
--quantified_assembly_component_usage(合格的装配构件使用)
--quasi_uniform_curve(准均匀曲线)
--quasi_uniform_surface(准均匀曲面)
--rational_b_spline_curve(有理B样条曲线)
--rational_b_spline_surface(有理B样条曲面)
--rectangular_composite_surface(矩形复合曲面)
--rectangular_trimmed_surface(矩形剪裁曲面)
--reparameterized_composite_curve_segment(重新参数化复合曲线段)
--representation(表示)
--representation_context(表示相关环境)
--representation_item(表示项)
--representation_map(表示映射)
--representation_relationship(表示关系)
--representation_relationship_with_transformation(有变换的表示关系)
--seam_curve(接合面曲线)
--security_classification(安全分类)
--security_classification_assignment(安全分类赋值)
--security_classification_level(安全分类等级)
--serial_numbered_effectivity(连续编号有效性)
--shape_aspect(形状特征)

--shape_aspect_relationship(形状特征关系)
--shape_definition_representation(形状定义表示)
--shape_representation(形状表示)
--shape_representation_relationship(形状表示关系)
--si_unit(公制单位);
--solid_angle_measure_with_unit(实体角度度量)
--solid_angle_unit(实体角度单位)
--specified_higher_usage_occurrence(规定的更高层活动)
--spherical_surface(球形曲面)
--start_request(开始请求)
--start_work(开始工作)
--supplied_part_relationship(外购零件关系)
--surface(曲面)
--surface_curve(曲面曲线)
--surface_of_linear_extrusion(线性延拓曲面)
--surface_of_revolution(旋转曲面)
--surface_patch(曲面片)
--surface_replica(曲面复制)
--swept_surface(扫成曲面)
--topological_representation_item(拓扑表示项)
--toroidal_surface(圆环曲面)
--trimmed_curve(剪裁曲线)
--uncertainty_measure_with_unit(误差度量)
--uniform_curve(均匀曲线)
--uniform_surface(均匀曲面)
--vector(矢量)
--versioned_action_request(该版活动请求)
--volume_measure_with_unit(体积度量)
--volume_unit(体积单位)
--week_of_year_and_day_date(年的星期和日期)

6.3 一致性级别3的实体

本标准一致性级别3的一个实现将支持下列实体和有关结构:

--action(活动)
--action_assignment(活动赋值)
--action_directive(活动命令)
--action_method(活动方法)
--action_request_assignment(活动请求赋值)
--action_request_solution(活动请求方案)
--action_request_status(活动请求状态)
--action_status(活动状态)
--address(地址)
--alternate_product_relationship(备用产品关系)
--application_context(应用相关环境)

--application_context_element(应用相关环境元素)
--application_protocol_definition(应用协议定义)
--approval(批准)
--approval_assignment(批准赋值)
--approval_date_time(批准日期)
--approval_person_organization(批准人员组织)
--approval_relationship(批准关系)
--approval_role(批准职能)
--approval_status(批准状态)
--area_measure_with_unit(面积度量)
--area_unit(面积单位)
--assembly_component_usage(装配构件使用)
--assembly_component_usage_substitute(装配构件使用代用)
--axis2_placement_3d(三维空间轴2方位)
--b_spline_curve(B样条曲线)
--b_spline_curve_with_knots(有结点的B样条曲线)
--bezier_curve(Bezier曲线)
--bounded_curve(有界曲线)
--calendar_date(日历日期)
--cartesian_point(笛卡尔点)
--cartesian_transformation_operator(笛卡尔变换运算器)
--cartesian_transformation_operator_3d(三维笛卡尔变换运算器)
--cc_design_approval(配置控制设计批准)
--cc_design_certification(配置控制设计认证)
--cc_design_contract(配置控制设计合同)
--cc_design_date_and_time_assignment(配置控制设计日期和时间赋值)
--cc_design_person_and_organization_assignment(配置控制设计人员和组织赋值)
--cc_design_security_classification(配置控制设计安全分类)
--cc_design_specification_reference(配置控制设计规范引用)
--certification(认证)
--certification_assignment(认证赋值)
--certification_type(认证类型)
--change(更改)
--change_request(更改请求)
--circle(圆)
--configuration_design(配置设计)
--configuration_effectivity(配置有效性)
--configuration_item(配置项)
--conic(圆锥曲线)
--connected_edge_set(连接边集)
--context_dependent_unit(相关环境依赖单位)
--contract(合同)
--contract_assignment(合同赋值)

--contract_type(合同类型)
--conversion_based_unit(换算单位)
--coordinated_universal_time_offset(协调的世界时差)
--curve(曲线)
--curve_replica(曲线复制)
--date(日期)
--date_and_time(日期和时间)
--date_and_time_assignment(日期和时间赋值)
--date_time_role(日期时间职能)
--dated_effectivity(日期有效性)
--definitional_representation(定义的表达)
--design_context(设计相关环境)
--design_make_from_relationship(设计参照关系)
--dimensional_exponents(量纲指数)
--directed_action(指定活动)
--direction(方向)
--document(文档)
--document_reference(文档引用)
--document_relationship(文档关系)
--document_type(文档类型)
--document_usage_constraint(文档使用约束)
--document_with_class(有级别文档)
--edge(边)
--edge_based_wireframe_model(基于边的线框模型)
--edge_based_wireframe_representation(基于边的线框表示)
--edge_curve(边曲线)
--edge_loop(边环)
--effectivity(有效性)
--ellipse(椭圆)
--executed_action(完成的活动)
--functionally_defined_transformation(功能上定义变换)
--geometric_representation_context(几何表示相关环境)
--geometric_representation_item(几何表示项)
--hyperbola(双曲线)
--length_measure_with_unit(长度度量)
--length_unit(长度单位)
--line(直线)
--local_time(当地时间)
--loop(环)
--lot_effectivity(批量有效性)
--mapped_item(映射项)
--mass_measure_with_unit(质量度量)
--mass_unit(质量单位)

--measure_with_unit(度量)
--mechanical_context(机械相关环境)
--named_unit(命名单位)
--next_assembly_usage_occurance(下一层装配活动)
--offset_curve_3d(三维偏置曲线)
--ordinal_date(顺序日期)
--organization(组织)
--organization_relationship(组织关系)
--organizational_address(组织的地址)
--organization_project(组织项目)
--oriented_edge(有向边)
--parabola(抛物线)
--path(路径)
--person(人员)
--person_and_organization(人员和组织)
--person_and_organization_assignment(人员和组织赋值)
--person_and_organization_role(人员和组织职能)
--personal_address(人员的地址)
--placement(定位)
--plane_angle_measure_with_unit(平面角度度量)
--plane_angle_unit(平面角度单位)
--point(点)
--point_on_curve(曲线上的点)
--point_replica(点复制)
--polyline(折线)
--product(产品)
--product_category(产品目录)
--product_category_relationship(产品目录关系)
--product_concept(产品概念)
--product_concept_context(产品概念相关环境)
--product_context(产品相关环境)
--product_definition(产品定义)
--product_definition_context(产品定义相关环境)
--product_definition_effectivity(产品定义有效性)
--product_definition_formation(产品定义形成)
--product_definition_formation_with_specified_source(有规定源的产品定义形成)
--product_definition_relationship(产品定义关系)
--product_definition_shape(产品定义形状)
--product_definition_usage(产品定义使用)
--product_definition_with_associated_documents(有相关文档的产品定义)
--product_related_product_category(与产品目录有关的产品)
--promissory_usage_occurance(约定的活动)
--property_definition(特性定义)

--quantified_assembly_component_usage(合格的装配构件使用)
--quasi_uniform_curve(准均匀曲线)
--rational_b_spline_curve(有理B样条曲线)
--representation(表示)
--representation_context(表示相关环境)
--representation_item(表示项)
--representation_map(表示映射)
--security_classification(安全分类)
--security_classification_assignment(安全分类赋值)
--security_classification_level(安全分类等级)
--serial_numbered_effectivity(连续编号有效性)
--shape_definition_representation(形状定义表示)
--shape_representation(形状表示)
--shell_based_wireframe_model(基于壳的线框模型)
--shell_based_wireframe_representation(基于壳的线框表示)
--si_unit(公制单位)
--solid_angle_measure_with_unit(实体角度度量)
--solid_angle_unit(实体角度单位)
--specified_higher_usage_occurrence(规定的更高层活动)
--start_request(开始请求)
--start_work(开始工作)
--supplied_part_relationship(外购零件关系)
--topological_representation_item(拓扑表示项)
--uncertainty_measure_with_unit(误差度量)
--uniform_curve(均匀曲线)
--vector(矢量)
--versioned_action_request(该版活动请求)
--vertex(顶)
--vertex_loop(顶环)
--vertex_point(顶点)
--vertex_shell(顶壳)
--volume_measure_with_unit(体积度量)
--volume_unit(体积单位)
--week_of_year_and_day_date(年的星期和日期)
--wire_shell(线壳)

6.4 一致性级别4的实体

本标准一致性级别4的一个实现将支持下列实体和有关结构:

--action(活动)
--action_assignment(活动赋值)
--action_directive(活动命令)
--action_method(活动方法)
--action_request_assignment(活动请求赋值)
--action_request_solution(活动请求方案)

--action_request_status(活动请求状态)
--action_status(活动状态)
--address(地址)
--advanced_face(高级的表面)
--alternate_product_relationship(备用产品关系)
--application_context(应用相关环境)
--application_context_element(应用相关环境元素)
--application_protocol_definition(应用协议定义)
--approval(批准)
--approval_assignment(批准赋值)
--approval_date_time(批准日期)
--approval_person_organization(批准人员组织)
--approval_relationship(批准关系)
--approval_role(批准职能)
--approval_status(批准状态)
--area_measure_with_unit(面积度量)
--area_unit(面积单位)
--assembly_component_usage(装配构件使用)
--assembly_component_usage_substitute(装配构件使用代用)
--axis 1 _placement(轴 1 方位)
--axis 2 _placement_2d(二维空间轴 2 方位)
--axis 2 _placement_3d(三维空间轴 2 方位)
--b_spline_curve(B 样条曲线)
--b_spline_curve_with_knots(有结点的 B 样条曲线)
--b_spline_surface(B 样条曲面)
--b_spline_surface_with_knots(有结点的 B 样条曲面)
--bezier_curve(Bezier 曲线)
--bezier_suface(Bezier 曲面)
--boundary_curve(边界曲线)
--bounded_curve(有界曲线)
--bounded_surface(有界曲面)
--calendar_date(日历日期)
--cartesian_point(笛卡尔点)
--cartesian_transformation_operator(笛卡尔变换运算器)
--cartesian_transformation_operator_3d(三维笛卡尔变换运算器)
--cc_desing_approval(配置控制设计批准)
--cc_desing_certification(配置控制设计认证)
--cc_desing_contract(配置控制设计合同)
--cc_desing_date_and_time_assignment(配置控制设计日期和时间赋值)
--cc_desing_person_and_organization_assignment(配置控制设计人员和组织赋值)
--cc_desing_security_classification(配置控制设计安全分类)
--cc_desing_specification_reference(配置控制设计规范引用)
--certification(认证)

--certification_assignment(认证赋值)
--certification_type(认证类型)
--change(更改)
--change_request(更改请求)
--circle(圆)
--closed_shell(封闭壳)
--configuration_design(配置设计)
--configuration_effectivity(配置有效性)
--configuration_item(配置项)
--conic(圆锥曲线)
--connected_face_set(连接面集合)
--context_dependent_unit(相关环境依赖单位)
--contract(合同)
--contract_assignment(合同赋值)
--contract_type(合同类型)
--conversion_based_unit(换算单位)
--coordinated_universal_time_offset(协调的世界时差)
--curve(曲线)
--curve_replica(曲线复制)
--date(日期)
--date_and_time(日期和时间)
--date_and_time_assignment(日期和时间赋值)
--date_time_role(日期时间职能)
--dated_effectivity(日期有效性)
--definitional_representation(定义的表达)
--degenerate_pcurve(退化参数域曲线)
--design_context(设计相关环境)
--design_make_from_relationship(设计参照关系)
--dimensional_exponents(量纲指数)
--directed_action(指定活动)
--direction(方向)
--document(文档)
--document_reference(文档引用)
--document_relationship(文档关系)
--document_type(文档类型)
--document_usage_constraint(文档使用约束)
--document_with_class(有级别文档)
--edge(边)
--edge_curve(边曲线)
--edge_loop(边环)
--effectivity(有效性)
--elementary_surface(初等曲面)
--ellipse(椭圆)

--evaluated_degenerate_pcurve(赋值的退化参数域曲线)
--executed_action(完成的活动)
--face(表面)
--face_bound(面界)
--face_outer_bound(表面外部界限)
--face_surface(表面曲面)
--functionally_defined_transformation(功能上定义的变换)
--geometric_representation_context(几何表示相关环境)
--geometric_representation_item(几何表示项)
--hyperbola(双曲线)
--intersection_curve(交线)
--length_measure_with_unit(长度度量)
--length_unit(长度单位)
--line(直线)
--local_time(当地时间)
--loop(环)
--lot_effectivity(批量有效性)
--manifold_surface_shape_representation(流形曲面形状表示)
--mapped_item(映射项)
--mass_measure_with_unit(质量度量)
--mass_unit(质量单位)
--measure_with_unit(度量)
--mechanical_context(机械相关环境)
--named_unit(命名单位)
--next_assembly_usage_occurance(下一层装配活动)
--offset_curve_3d(三维偏置曲线)
--offset_surface(偏置曲面)
--open_shell(开式壳)
--ordinal_date(顺序日期)
--organization(组织)
--organization_relationship(组织关系)
--organizational_address(组织的地址)
--organization_project(组织项目)
--oriented_closed_shell(有向的闭式壳)
--oriented_edge(有向的边)
--oriented_face(有向的表面)
--oriented_open_shell(有向的开式壳)
--oriented_path(有向的路径)
--parabola(抛物线)
--parametric_representation_context(参数表达相关环境)
--path(路径)
--pcurve(参数域曲线)
--person(人员)

--person_and_organization(人员和组织)
--person_and_organization_assignment(人员和组织赋值)
--person_and_organization_role(人员和组织职能)
--personal_address(人员的地址)
--placement(定位)
--plane(平面)
--plane_angle_measure_with_unit(平面角度度量)
--plane_angle_unit(平面角度单位)
--point(点)
--point_on_curve(曲线上的点)
--point_on_surface(曲面上的点)
--print_replica(点复制)
--polyline(折线)
--product(产品)
--product_category(产品目录)
--product_category_relationship(产品目录关系)
--product_concept(产品概念)
--product_concept_context(产品概念相关环境)
--product_context(产品相关环境)
--product_definition(产品定义)
--product_definition_context(产品定义相关环境)
--product_definition_effectivity(产品定义有效性)
--product_definition_formation(产品定义形成)
--product_definition_formation_with_specified_source(有规定源的产品定义形成)
--product_definition_relationship(产品定义关系)
--product_definition_shape(产品定义形状)
--product_definition_usage(产品定义使用)
--product_definition_with_associated_documents(有相关文档的产品定义)
--product_related_product_category(与产品目录有关的产品)
--promissory_usage_occurance(约定的活动)
--property_definition(特性定义)
--quantified_assembly_component_usage(合格的装配构件使用)
--quasi_uniform_curve(准均匀曲线)
--quasi_uniform_surface(准均匀曲面)
--rational_b_spline_curve(有理B样条曲线)
--rational_b_spline_surface(有理B样条曲面)
--representation(表示)
--representation_context(表示相关环境)
--representation_item(表示项)
--representation_map(表示映射)
--seam_curve(接合面曲线)
--security_classification(安全分类)
--security_classification_assignment(安全分类赋值)

--security_classification_level(安全分类等级)
--serial_numbered_effectivity(连续编号有效性)
--shape_definition_representation(形状定义表示)
--shape_representation(形状表示)
--shell_based_surface_model(基于壳的曲面模型)
--si_unit(公制单位)
--solid_angle_measure_with_unit(实体角度度量)
--solid_angle_unit(实体角度单位)
--specified_higher_usage_occurrence(规定的更高层活动)
--spherical_surface(球形曲面)
--start_request(开始请求)
--start_work(开始工作)
--supplied_part_relationship(外购零件关系)
--surface(曲面)
--surface_curve(曲面曲线)
--surface_of_linear_extrusion(线性延拓曲面)
--surface_of_revolution(旋转曲面)
--surface_replica(曲面复制)
--swept_surface(扫成曲面)
--topological_representation_item(拓扑表示项)
--uncertainty_measure_with_unit(误差度量)
--uniform_curve(均匀曲线)
--uniform_surface(均匀曲面)
--vector(矢量)
--versioned_action_request(该版活动请求)
--vertex(顶)
--vertex_loop(顶环)
--vertex_point(顶点)
--volume_measure_with_unit(体积度量)
--volume_unit(体积单位)
--week_of_year_and_day_date(年的星期和日期)

6.5 一致性级别5的实体

本标准一致性级别5的一个实现将支持下列实体和有关结构：

--action(活动)
--action_assignment(活动赋值)
--action_directive(活动命令)
--action_method(活动方法)
--action_request_assignment(活动请求赋值)
--action_request_solution(活动请求方案)
--action_request_status(活动请求状态)
--action_status(活动状态)
--address(地址)
--alternate_product_relationship(备用产品关系)

--application_context(应用相关环境)
--application_context_element(应用相关环境元素)
--application_protocol_definition(应用协议定义)
--approval(批准)
--approval_assignment(批准赋值)
--approval_date_time(批准日期)
--approval_person_organization(批准人员组织)
--approval_relationship(批准关系)
--approval_role(批准职能)
--approval_status(批准状态)
--area_measure_with_unit(面积度量)
--area_unit(面积单位)
--assembly_component_usage(装配构件使用)
--assembly_component_usage_substitute(装配构件使用代用)
--axis 2_placement_3d(三维空间轴 2 方位)
--brep_with_voids(有空洞的边界表示)
--calendar_date(日历日期)
--cartesian_point(笛卡尔点)
--cc_desing_approval(配置控制设计批准)
--cc_desing_certification(配置控制设计认证)
--cc_desing_contract(配置控制设计合同)
--cc_desing_date_and_time_assignment(配置控制设计日期和时间赋值)
--cc_desing_person_and_organization_assignment(配置控制设计人员和组织赋值)
--cc_desing_security_classification(配置控制设计安全分类)
--cc_desing_specification_reference(配置控制设计规范引用)
--certification(认证)
--certification_assignment(认证赋值)
--certification_type(认证类型)
--change(更改)
--change_request(更改请求)
--brep_with_voids(有空洞的边界表示)
--closed_shell(封闭壳)
--configuration_design(配置设计)
--configuration_effectivity(配置有效性)
--configuration_item(配置项)
--connected_face_set(连接面集合)
--context_dependent_unit(相关环境依赖单位)
--contract(合同)
--contract_assignment(合同赋值)
--contract_type(合同类型)
--conversion_based_unit(换算单位)
--coordinated_universal_time_offset(协调的世界时差)
--date(日期)

--date_and_time(日期和时间)
--date_and_time_assignment(日期和时间赋值)
--date_time_role(日期时间职能)
--dated_effectivity(日期有效性)
--definitional_representation(定义的表达)
--design_context(设计相关环境)
--design_make_from_relationship(设计参照关系)
--dimensional_exponents(量纲指数)
--directed_action(指定活动)
--direction(方向)
--document(文档)
--document_reference(文档引用)
--document_relationship(文档关系)
--document_type(文档类型)
--document_usage_constraint(文档使用约束)
--document_with_class(有级别文档)
--edge(边)
--effectivity(有效性)
--elementary_surface(初等曲面)
--executed_action(完成的活动)
--face(表面)
--face_bound(面界)
--face_outer_bound(表面外部界限)
--face_surface(表面曲面)
--faceted_brep(小平面边界表示)
--faceted_brep_shape_representation(小平面边界表示形状表示)
--geometric_representation_context(几何表示相关环境)
--geometric_representation_item(几何表示项)
--length_measure_with_unit(长度度量)
--length_unit(长度单位)
--local_time(当地时间)
--loop(环)
--lot_effectivity(批量有效性)
--manifold_solid_brep(流形实体边界表示)
--mapped_item(映射项)
--mass_measure_with_unit(质量度量)
--mass_unit(质量单位)
--measure_with_unit(度量)
--mechanical_context(机械相关环境)
--named_unit(命名单位)
--next_assembly_usage_occurance(下一层装配活动)
--open_shell(开式壳)
--ordinal_date(顺序日期)

--organization(组织)
--organization_relationship(组织关系)
--organizational_address(组织的地址)
--organization_project(组织项目)
--oriented_closed_shell(有向闭式壳)
--oriented_edge(有向边)
--oriented_face(有向表面)
--oriented_open_shell(有向开式壳)
--oriented_path(有向路径)
--path(路径)
--person(人员)
--person_and_organization(人员和组织)
--person_and_organization_assignment(人员和组织赋值)
--person_and_organization_role(人员和组织职能)
--personal_address(人员的地址)
--placement(定位)
--plane_angle_measure_with_unit(平面角度度量)
--plane_angle_unit(平面角度单位)
--point(点)
--poly_loop(多环)
--product(产品)
--product_category(产品目录)
--product_category_relationship(产品目录关系)
--product_concept(产品概念)
--product_concept_context(产品概念相关环境)
--product_context(产品相关环境)
--product_definition(产品定义)
--product_definition_context(产品定义相关环境)
--product_definition_effectivity(产品定义有效性)
--product_definition_formation(产品定义形成)
--product_definition_formation_with_specified_source(有规定源的产品定义形成)
--product_definition_relationship(产品定义关系)
--product_definition_shape(产品定义形状)
--product_definition_usage(产品定义使用)
--product_definition_with_associated_documents(有相关文档的产品定义)
--product_related_product_category(与产品目录有关的产品)
--promissory_usage_occurance(约定的活动)
--property_definition(特性定义)
--quantified_assembly_component_usage(合格的装配构件使用)
--representation(表示)
--representation_context(表示相关环境)
--representation_item(表示项)
--representation_map(表示映射)

--security_classification(安全分类)
--security_classification_assignment(安全分类赋值)
--security_classification_level(安全分类等级)
--serial_numbered_effectivity(连续编号有效性)
--shape_definition_representation(形状定义表示)
--shape_representation(形状表示)
--si_unit(公制单位);
--solid_angle_measure_with_unit(实体角度度量)
--solid_angle_unit(实体角度单位)
--solid_model(实体模型)
--specified_higher_usage_occurrence(规定的更高层活动)
--start_request(开始请求)
--start_work(开始工作)
--supplied_part_relationship(外购零件关系)
--surface(曲面)
--topological_representation_item(拓扑表示项)
--uncertainty_measure_with_unit(误差度量)
--vector(矢量)
--versioned_action_request(该版活动请求)
--vertex(顶)
--volume_measure_with_unit(体积度量)
--volume_unit(体积单位)
--week_of_year_and_day_date(年的星期和日期)

6.6 一致性级别6的实体

本标准一致性级别6的一个实现将支持下列实体和有关结构:

--action(活动)
--action_assignment(活动赋值)
--action_directive(活动命令)
--action_method(活动方法)
--action_request_assignment(活动请求赋值)
--action_request_solution(活动请求方案)
--action_request_status(活动请求状态)
--action_status(活动状态)
--address(地址)
--advanced_brep_shape_representation(高级的边界表示形状表示)
--advanced_face(高级的表面)
--alternate_product_relationship(备用产品关系)
--application_context(应用相关环境)
--application_context_element(应用相关环境元素)
--application_protocol_definition(应用协议定义)
--approval(批准)
--approval_assignment(批准赋值)
--approval_date_time(批准日期)

--approval_person_organization(批准人员组织)
--approval_relationship(批准关系)
--approval_role(批准职能)
--approval_status(批准状态)
--area_measure_with_unit(面积度量)
--area_unit(面积单位)
--assembly_component_usage(装配构件使用)
--assembly_component_usage_substitute(装配构件使用代用)
--axis 1_placement(轴 1 方位)
--axis 2_placement_3d(三维空间轴 2 方位)
--b_spline_curve(B 样条曲线)
--b_spline_curve_with_knots(有结点的 B 样条曲线)
--b_spline_surface(B 样条曲面)
--b_spline_surface_with_knots(有结点的 B 样条曲面)
--bezier_curve(Bezier 曲线)
--bezier_surface(Bezier 曲面)
--bounded_curve(有界曲线)
--bounded_surface(有界曲面)
--brep_with_voids(有空洞的边界表示)
--calendar_date(日历日期)
--cartesian_point(笛卡尔点)
--cc_desing_approval(配置控制设计批准)
--cc_desing_certification(配置控制设计认证)
--cc_desing_contract(配置控制设计合同)
--cc_desing_date_and_time_assignment(配置控制设计日期和时间赋值)
--cc_desing_person_and_organization_assignment(配置控制设计人员和组织赋值)
--cc_desing_security_classification(配置控制设计安全分类)
--cc_desing_specification_reference(配置控制设计规范引用)
--certification(认证)
--certification_assignment(认证赋值)
--certification_type(认证类型)
--change(更改)
--change_request(更改请求)
--circle(圆)
--closed_shell(闭式壳)
--configuration_design(配置设计)
--configuration_effectivity(配置有效性)
--configuration_item(配置项)
--conic(圆锥曲线)
--connected_face_set(连接面集合)
--context_dependent_unit(相关环境依赖单位)
--contract(合同)
--contract_assignment(合同赋值)

--contract_type(合同类型)
--conversion_based_unit(换算单位)
--coordinated_universal_time_offset(协调的世界时差)
--curve(曲线)
--date(日期)
--date_and_time(日期和时间)
--date_and_time_assignment(日期和时间赋值)
--date_time_role(日期时间职能)
--dated_effectivity(日期有效性)
--design_context(设计相关环境)
--design_make_from_relationship(设计参照关系)
--dimensional_exponents(量纲指数)
--directed_action(指定活动)
--direction(方向)
--document(文档)
--document_reference(文档引用)
--document_relationship(文档关系)
--document_type(文档类型)
--document_usage_constraint(文档使用约束)
--document_with_class(有级别文档)
--edge(边)
--edge_curve(边曲线)
--edge_loop(边环)
--effectivity(有效性)
--elementary_surface(初等曲面)
--ellipse(椭圆)
--executed_action(完成的活动)
--face(表面)
--face_bound(面界)
--face_outer_bound(表面外部界限)
--face_surface(表面曲面)
--geometric_representation_context(几何表示相关环境)
--geometric_representation_item(几何表示项)
--hyperbola(双曲线)
--intersection_curve(交线)
--item_defined_transformation(项定义的变换)
--length_measure_with_unit(长度度量)
--length_unit(长度单位)
--line(直线)
--local_time(当地时间)
--loop(环)
--lot_effectivity(批量有效性)
--manifold_solid_brep(流形实体边界表示)

--mapped_item(映射项)
--mass_measure_with_unit(质量度量)
--mass_unit(质量单位)
--measure_with_unit(度量)
--mechanical_context(机械相关环境)
--named_unit(命名单位)
--next_assembly_usage_occurance(下一层装配活动)
--open_shell(开式壳)
--ordinal_date(顺序日期)
--organization(组织)
--organization_relationship(组织关系)
--organizational_address(组织的地址)
--organization_project(组织项目)
--oriented_closed_shell(有向闭式壳)
--oriented_edge(有向边)
--oriented_face(有向表面)
--oriented_open_shell(有向开式壳)
--oriented_path(有向路径)
--parabola(抛物线)
--path(路径)
--person(人员)
--person_and_organization(人员和组织)
--person_and_organization_assignment(人员和组织赋值)
--person_and_organization_role(人员和组织职能)
--personal_address(人员的地址)
--placement(定位)
--plane(平面)
--plane_angle_measure_with_unit(平面角度度量)
--plane_angle_unit(平面角度单位)
--point(点)
--polyline(折线)
--product(产品)
--product_category(产品目录)
--product_category_relationship(产品目录关系)
--product_concept(产品概念)
--product_concept_context(产品概念相关环境)
--product_context(产品相关环境)
--product_definition(产品定义)
--product_definition_context(产品定义相关环境)
--product_definition_effectivity(产品定义有效性)
--product_definition_formation(产品定义形成)
--product_definition_formation_with_specified_source(有规定源的产品定义形成)
--product_definition_relationship(产品定义关系)

--product_definition_shape(产品定义形状)
--product_definition_usage(产品定义使用)
--product_definition_with_associated_documents(有相关文档的产品定义)
--product_related_product_category(与产品目录有关的产品)
--promissory_usage_occurance(约定的活动)
--property_definition(特性定义)
--quantified_assembly_component_usage(合格的装配构件使用)
--quasi_uniform_curve(准均匀曲线)
--quasi_uniform_surface(准均匀曲面)
--rational_b_spline_curve(有理B样条曲线)
--rational_b_spline_surface(有理B样条曲面)
--representation(表示)
--representation_context(表示相关环境)
--representation_item(表示项)
--representation_map(表示映射)
--security_classification(安全分类)
--security_classification_assignment(安全分类赋值)
--security_classification_level(安全分类等级)
--serial_numbered_effectivity(连续编号有效性)
--shape_definition_representation(形状定义表示)
--shape_representation(形状表示)
--si_unit(公制单位);
--solid_angle_measure_with_unit(实体角度度量)
--solid_angle_unit(实体角度单位)
--solid_model(实体模型)
--specified_higher_usage_occurrence(规定的更高层活动)
--spherical_surface(球形曲面)
--start_request(开始请求)
--start_work(开始工作)
--supplied_part_relationship(外购零件关系)
--surface(曲面)
--surface_of_linear_extrusion(线性延拓曲面)
--surface_of_revolution(旋转曲面)
--swept_surface(扫成曲面)
--topological_representation_item(拓扑表示项)
--uncertainty_measure_with_unit(误差度量)
--uniform_curve(均匀曲线)
--uniform_surface(均匀曲面)
--vector(矢量)
--versioned_action_request(该版活动请求)
--vertex(顶)
--vertex_loop(顶环)
--vertex_point(顶点)

--volume_measure_with_unit(体积度量)

--volume_unit(体积单位)

--week_of_year_and_day_date(年的星期和日期)

附 录 A
（标准的附录）
应用解释模型 EXPRESS 详表

```
*)
SCHEMA config_control_design;

TYPE ahead_or_behind = ENUMERATION OF
  (ahead,
   behind);
END_TYPE; -- ahead_or_behind

TYPE approved_item = SELECT
  (product_definition_formation,
   product_definition,
   configuration_effectivity,
   configuration_item,
   security_classification,
   change_request,
   change,
   start_request,
   start_work,
   certification,
   contract);
END_TYPE; -- approved_item

TYPE area_measure = REAL;
END_TYPE; -- area_measure

TYPE axis2_placement = SELECT
  (axis2_placement_2d,
   axis2_placement_3d);
END_TYPE; -- axis2_placement

TYPE b_spline_curve_form = ENUMERATION OF
  (elliptic_arc,
   polyline_form,
   parabolic_arc,
   circular_arc,
   unspecified,
   hyperbolic_arc);
END_TYPE; -- b_spline_curve_form
```

```
TYPE b_spline_surface_form = ENUMERATION OF
  (surf_of_linear_extrusion,
   plane_surf,
   generalised_cone,
   toroidal_surf,
   conical_surf,
   spherical_surf,
   unspecified,
   ruled_surf,
   surf_of_revolution,
   cylindrical_surf,
   quadric_surf);
END_TYPE; -- b_spline_surface_form

TYPE boolean_operand = SELECT
  (solid_model);
END_TYPE; -- boolean_operand

TYPE certified_item = SELECT
  (supplied_part_relationship);
END_TYPE; -- certified_item

TYPE change_request_item = SELECT
  (product_definition_formation);
END_TYPE; -- change_request_item

TYPE characterized_definition = SELECT
  (characterized_product_definition,
   shape_definition);
END_TYPE; -- characterized_definition

TYPE characterized_product_definition = SELECT
  (product_definition,
   product_definition_relationship);
END_TYPE; -- characterized_product_definition

TYPE classified_item = SELECT
  (product_definition_formation,
   assembly_component_usage);
END_TYPE; -- classified_item

TYPE context_dependent_measure = REAL;
END_TYPE; -- context_dependent_measure
```

```
TYPE contracted_item = SELECT
  (product_definition_formation);
END_TYPE; -- contracted_item

TYPE count_measure = NUMBER;
END_TYPE; -- count_measure

TYPE curve_on_surface = SELECT
  (pcurve,
   surface_curve,
   composite_curve_on_surface);
END_TYPE; -- curve_on_surface

TYPE date_time_item = SELECT
  (product_definition,
   change_request,
   start_request,
   change,
   start_work,
   approval_person_organization,
   contract,
   security_classification,
   certification);
END_TYPE; -- date_time_item

TYPE date_time_select = SELECT
  (date_and_time);
END_TYPE; -- date_time_select

TYPE day_in_month_number = INTEGER;
END_TYPE; -- day_in_month_number

TYPE day_in_week_number = INTEGER;
WHERE
  WR1: (1 <= SELF) AND (SELF <= 7);
END_TYPE; -- day_in_week_number

TYPE day_in_year_number = INTEGER;
END_TYPE; -- day_in_year_number

TYPE descriptive_measure = STRING;
END_TYPE; -- descriptive_measure

TYPE dimension_count = INTEGER;
```

```
WHERE
  WR1: SELF > 0;
END_TYPE; -- dimension_count

TYPE geometric_set_select = SELECT
  (point,
   curve,
   surface);
END_TYPE; -- geometric_set_select

TYPE hour_in_day = INTEGER;
WHERE
  WR1: (0 <= SELF) AND (SELF < 24);
END_TYPE; -- hour_in_day

TYPE identifier = STRING;
END_TYPE; -- identifier

TYPE knot_type = ENUMERATION OF
  (uniform_knots,
   quasi_uniform_knots,
   piecewise_bezier_knots,
   unspecified);
END_TYPE; -- knot_type

TYPE label = STRING;
END_TYPE; -- label

TYPE length_measure = REAL;
END_TYPE; -- length_measure

TYPE list_of_reversible_topology_item = LIST [0:?] OF
            reversible_topology_item;
END_TYPE; -- list_of_reversible_topology_item

TYPE mass_measure = REAL;
END_TYPE; -- mass_measure

TYPE measure_value = SELECT
  (area_measure,
   context_dependent_measure,
   count_measure,
   descriptive_measure,
   length_measure,
```

```
   mass_measure,
   plane_angle_measure,
   parameter_value,
   positive_length_measure,
   positive_plane_angle_measure,
   solid_angle_measure,
   volume_measure);
END_TYPE; -- measure_value

TYPE minute_in_hour = INTEGER;
WHERE
  WR1: (0 <= SELF) AND (SELF <= 59);
END_TYPE; -- minute_in_hour

TYPE month_in_year_number = INTEGER;
WHERE
  WR1: (1 <= SELF) AND (SELF <= 12);
END_TYPE; -- month_in_year_number

TYPE parameter_value = REAL;
END_TYPE; -- parameter_value

TYPE pcurve_or_surface = SELECT
  (pcurve,
   surface);
END_TYPE; -- pcurve_or_surface

TYPE person_organization_item = SELECT
  (change,
   start_work,
   change_request,
   start_request,
   configuration_item,
   product,
   product_definition_formation,
   product_definition,
   contract,
   security_classification);
END_TYPE; -- person_organization_item

TYPE person_organization_select = SELECT
  (person_and_organization);
END_TYPE; -- person_organization_select
```

```
TYPE plane_angle_measure = REAL;
END_TYPE; -- plane_angle_measure

TYPE positive_length_measure = length_measure;
WHERE
  WR1: SELF > 0;
END_TYPE; -- positive_length_measure

TYPE positive_plane_angle_measure = plane_angle_measure;
WHERE
  WR1: SELF > 0;
END_TYPE; -- positive_plane_angle_measure

TYPE preferred_surface_curve_representation = ENUMERATION OF
  (pcurve_s2,
   pcurve_s1,
   curve_3d);
END_TYPE; -- preferred_surface_curve_representation

TYPE reversible_topology = SELECT
  (reversible_topology_item,
   list_of_reversible_topology_item,
   set_of_reversible_topology_item);
END_TYPE; -- reversible_topology

TYPE reversible_topology_item = SELECT
  (edge,
   path,
   face,
   face_bound,
   closed_shell,
   open_shell);
END_TYPE; -- reversible_topology_item

TYPE second_in_minute = REAL;
WHERE
  WR1: (0 <= SELF) AND (SELF < 60);
END_TYPE; -- second_in_minute

TYPE set_of_reversible_topology_item = SET [0:?] OF
            reversible_topology_item;
END_TYPE; -- set_of_reversible_topology_item

TYPE shape_definition = SELECT
```

```
  (product_definition_shape,
   shape_aspect,
   shape_aspect_relationship);
END_TYPE; -- shape_definition

TYPE shell = SELECT
  (open_shell,
   closed_shell,
   vertex_shell,
   wire_shell);
END_TYPE; -- shell

TYPE si_prefix = ENUMERATION OF
  (exa,
   pico,
   mega,
   femto,
   atto,
   centi,
   nano,
   hecto,
   micro,
   tera,
   giga,
   milli,
   peta,
   deci,
   kilo,
   deca);
END_TYPE; -- si_prefix

TYPE si_unit_name = ENUMERATION OF
  (hertz,
   degree_celsius,
   siemens,
   sievert,
   lux,
   watt,
   ohm,
   second,
   becquerel,
   pascal,
   henry,
   tesla,
```

```
    volt,
    joule,
    kelvin,
    ampere,
    gram,
    steradian,
    mole,
    lumen,
    gray,
    candela,
    farad,
    radian,
    newton,
    metre,
    weber,
    coulomb);
END_TYPE; -- si_unit_name

TYPE solid_angle_measure = REAL;
END_TYPE; -- solid_angle_measure

TYPE source = ENUMERATION OF
  (bought,
   not_known,
   made);
END_TYPE; -- source

TYPE specified_item = SELECT
  (product_definition,
   shape_aspect);
END_TYPE; -- specified_item

TYPE start_request_item = SELECT
  (product_definition_formation);
END_TYPE; -- start_request_item

TYPE supported_item = SELECT
  (action_directive,
   action,
   action_method);
END_TYPE; -- supported_item

TYPE surface_model = SELECT
  (shell_based_surface_model);
```

```
END_TYPE; -- surface_model

TYPE text = STRING;
END_TYPE; -- text

TYPE transformation = SELECT
  (item_defined_transformation,
   functionally_defined_transformation);
END_TYPE; -- transformation

TYPE transition_code = ENUMERATION OF
  (discontinuous,
   cont_same_gradient_same_curvature,
   cont_same_gradient,
   continuous);
END_TYPE; -- transition_code

TYPE trimming_preference = ENUMERATION OF
  (parameter,
   unspecified,
   cartesian);
END_TYPE; -- trimming_preference

TYPE trimming_select = SELECT
  (cartesian_point,
   parameter_value);
END_TYPE; -- trimming_select

TYPE unit = SELECT
  (named_unit);
END_TYPE; -- unit

TYPE vector_or_direction = SELECT
  (vector,
   direction);
END_TYPE; -- vector_or_direction

TYPE volume_measure = REAL;
END_TYPE; -- volume_measure

TYPE week_in_year_number = INTEGER;
WHERE
  WR1: (1 <= SELF) AND (SELF <= 53);
END_TYPE; -- week_in_year_number
```

```
TYPE wireframe_model = SELECT
  (shell_based_wireframe_model,
   edge_based_wireframe_model);
END_TYPE; -- wireframe_model

TYPE work_item = SELECT
  (product_definition_formation);
END_TYPE; -- work_item

TYPE year_number = INTEGER;
END_TYPE; -- year_number

ENTITY action;
  name          : label;
  description   : text;
  chosen_method : action_method;
END_ENTITY; -- action

ENTITY action_assignment
  ABSTRACT SUPERTYPE;
  assigned_action : action;
END_ENTITY; -- action_assignment

ENTITY action_directive;
  name        : label;
  description : text;
  analysis    : text;
  comment     : text;
  requests    : SET [1:?] OF versioned_action_request;
END_ENTITY; -- action_directive

ENTITY action_method;
  name        : label;
  description : text;
  consequence : text;
  purpose     : text;
END_ENTITY; -- action_method

ENTITY action_request_assignment
  ABSTRACT SUPERTYPE;
  assigned_action_request : versioned_action_request;
END_ENTITY; -- action_request_assignment

ENTITY action_request_solution;
```

```
  method  : action_method;
  request : versioned_action_request;
END_ENTITY; -- action_request_solution

ENTITY action_request_status;
  status           : label;
  assigned_request : versioned_action_request;
END_ENTITY; -- action_request_status

ENTITY action_status;
  status          : label;
  assigned_action : executed_action;
END_ENTITY; -- action_status

ENTITY address;
  internal_location       : OPTIONAL label;
  street_number           : OPTIONAL label;
  street                  : OPTIONAL label;
  postal_box              : OPTIONAL label;
  town                    : OPTIONAL label;
  region                  : OPTIONAL label;
  postal_code             : OPTIONAL label;
  country                 : OPTIONAL label;
  facsimile_number        : OPTIONAL label;
  telephone_number        : OPTIONAL label;
  electronic_mail_address : OPTIONAL label;
  telex_number            : OPTIONAL label;
WHERE
  WR1: EXISTS(internal_location)       OR
       EXISTS(street_number)           OR
       EXISTS(street)                  OR
       EXISTS(postal_box)              OR
       EXISTS(town)                    OR
       EXISTS(region)                  OR
       EXISTS(postal_code)             OR
       EXISTS(country)                 OR
       EXISTS(facsimile_number)        OR
       EXISTS(telephone_number)        OR
       EXISTS(electronic_mail_address) OR
       EXISTS(telex_number);
END_ENTITY; -- address

ENTITY advanced_brep_shape_representation
  SUBTYPE OF (shape_representation);
```

```
WHERE
  WR1 : SIZEOF (QUERY (it <* SELF.items |
        NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP',
        'CONFIG_CONTROL_DESIGN.FACETED_BREP',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
        'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1)))
        = 0;
  WR2 : SIZEOF (QUERY (it <* SELF.items |
        SIZEOF (['CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) =1 )) > 0;
  WR3 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
        'CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
        NOT (SIZEOF (QUERY (csh <*
        msb_shells(msb, 'CONFIG_CONTROL_DESIGN') |
        NOT (SIZEOF (QUERY(csf <* csh.cfs_faces |
        NOT('CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF(csf)))) = 0
        ))) = 0))) = 0;
  WR4 : SIZEOF (QUERY (msb <* QUERY (it <* items |
        'CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
        'CONFIG_CONTROL_DESIGN.ORIENTED_CLOSED_SHELL' IN
        TYPEOF (msb\manifold_solid_brep.outer))) = 0;
  WR5 : SIZEOF (QUERY (brv <* QUERY (it <* items |
        'CONFIG_CONTROL_DESIGN.BREP_WITH_VOIDS' IN TYPEOF(it)) |
        NOT (SIZEOF (QUERY (csh <* brv\brep_with_voids.voids |
        csh\oriented_closed_shell.orientation)) = 0))) = 0;
  WR6 : SIZEOF (QUERY (mi <*  QUERY (it <* items |
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) |
        NOT ('CONFIG_CONTROL_DESIGN.ADVANCED_BREP_SHAPE_REPRESENTATION' IN
        TYPEOF(mi\mapped_item.mapping_source.mapped_representation)))) = 0;
END_ENTITY; -- advanced_brep_shape_representation

ENTITY advanced_face
  SUBTYPE OF (face_surface);
WHERE
  WR1: SIZEOF (['CONFIG_CONTROL_DESIGN.ELEMENTARY_SURFACE',
       'CONFIG_CONTROL_DESIGN.B_SPLINE_SURFACE',
       'CONFIG_CONTROL_DESIGN.SWEPT_SURFACE'] *
       TYPEOF (face_geometry)) = 1;
  WR2: SIZEOF (QUERY (bnd <* bounds |
       NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.EDGE_LOOP',
       'CONFIG_CONTROL_DESIGN.VERTEX_LOOP'] * TYPEOF (bnd)) = 1))) = 0;
  WR3: SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* bounds |
       'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
       NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
```

```
      NOT ('CONFIG_CONTROL_DESIGN.EDGE_CURVE' IN
      TYPEOF (oe.edge_element)))) = 0))) = 0;
  WR4: SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* bounds |
      'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT (('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN TYPEOF (oe.edge_start))
      AND
      ('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN TYPEOF (oe.edge_end)))))
      = 0))) = 0;
  WR5: SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* bounds |
      'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.LINE',
      'CONFIG_CONTROL_DESIGN.CONIC',
      'CONFIG_CONTROL_DESIGN.POLYLINE',
      'CONFIG_CONTROL_DESIGN.PCURVE',
      'CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE'] *
      TYPEOF (oe.edge_element\edge_curve.edge_geometry)) = 1))) = 0))) = 0;
  WR6: SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* bounds |
      'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
      NOT ('CONFIG_CONTROL_DESIGN.ORIENTED_PATH' IN
      TYPEOF (elp_fbnds.bound)))) = 0;
  WR7: (NOT ('CONFIG_CONTROL_DESIGN.SWEPT_SURFACE' IN
      TYPEOF(face_geometry)))
      OR
      (SIZEOF(['CONFIG_CONTROL_DESIGN.LINE',
      'CONFIG_CONTROL_DESIGN.CONIC',
      'CONFIG_CONTROL_DESIGN.POLYLINE',
      'CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE'] *
      TYPEOF (face_geometry\swept_surface.swept_curve)) = 1);
  WR8: SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* bounds |
      'CONFIG_CONTROL_DESIGN.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN
      TYPEOF (vlp_fbnds.bound\vertex_loop.loop_vertex)) AND
      ('CONFIG_CONTROL_DESIGN.CARTESIAN_POINT' IN
      TYPEOF
      (vlp_fbnds.bound\vertex_loop.loop_vertex\vertex_point.vertex_geometry)
      )))) = 0;
END_ENTITY; -- advanced_face

ENTITY alternate_product_relationship;
  name        : label;
  description : text;
  alternate   : product;
```

```
  base        : product;
  basis       : text;
UNIQUE
  UR1 : alternate, base;
WHERE
  WR1: alternate :<>: base;
END_ENTITY; -- alternate_product_relationship

ENTITY application_context;
  application : text;
INVERSE
  context_elements : SET [1:?] OF application_context_element FOR
                        frame_of_reference;
END_ENTITY; -- application_context

ENTITY application_context_element
  SUPERTYPE OF (ONEOF (product_context, product_definition_context,
  product_concept_context));
  name               : label;
  frame_of_reference : application_context;
END_ENTITY; -- application_context_element

ENTITY application_protocol_definition;
  status                                  : label;
  application_interpreted_model_schema_name : label;
  application_protocol_year               : year_number;
  application                             : application_context;
END_ENTITY; -- application_protocol_definition

ENTITY approval;
  status : approval_status;
  level  : label;
END_ENTITY; -- approval

ENTITY approval_assignment
ABSTRACT SUPERTYPE;
  assigned_approval : approval;
END_ENTITY; -- approval_assignment

ENTITY approval_date_time;
  date_time      : date_time_select;
  dated_approval : approval;
END_ENTITY; -- approval_date_time
```

```
ENTITY approval_person_organization;
  person_organization : person_organization_select;
  authorized_approval : approval;
  role/               : approval_role;
END_ENTITY; -- approval_person_organization

ENTITY approval_relationship;
  name              : label;
  description       : text;
  relating_approval : approval;
  related_approval  : approval;
END_ENTITY; -- approval_relationship

ENTITY approval_role;
  role : label;
END_ENTITY; -- approval_role

ENTITY approval_status;
  name : label;
END_ENTITY; -- approval_status

ENTITY area_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.AREA_UNIT' IN
       TYPEOF(SELF\measure_with_unit.unit_component);
END_ENTITY; -- area_measure_with_unit

ENTITY area_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 2) AND
       (SELF\named_unit.dimensions.mass_exponent = 0) AND
       (SELF\named_unit.dimensions.time_exponent = 0) AND
       (SELF\named_unit.dimensions.electric_current_exponent = 0) AND
       (SELF\named_unit.dimensions.
       thermodynamic_temperature_exponent = 0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent = 0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- area_unit

ENTITY assembly_component_usage
  SUPERTYPE OF (ONEOF (next_assembly_usage_occurrence,
  specified_higher_usage_occurrence, promissory_usage_occurrence))
```

```
  SUBTYPE OF (product_definition_usage);
  reference_designator : OPTIONAL identifier;
END_ENTITY; -- assembly_component_usage

ENTITY assembly_component_usage_substitute;
  name        : label;
  description : text;
  base        : assembly_component_usage;
  substitute  : assembly_component_usage;
UNIQUE
  UR1 : base, substitute;
WHERE
  WR1: base.relating_product_definition :=: substitute.
       relating_product_definition;
  WR2: base :<>: substitute;
END_ENTITY; -- assembly_component_usage_substitute

ENTITY axis1_placement
  SUBTYPE OF (placement);
  axis : OPTIONAL direction;
DERIVE
  z : direction := NVL(normalise(axis),direction([0,0,1]));
WHERE
  WR1: SELF\geometric_representation_item.dim = 3;
END_ENTITY; -- axis1_placement

ENTITY axis2_placement_2d
  SUBTYPE OF (placement);
  ref_direction : OPTIONAL direction;
DERIVE
  p : LIST [2:2] OF direction := build_2axes(ref_direction);
WHERE
  WR1: SELF\geometric_representation_item.dim = 2;
END_ENTITY; -- axis2_placement_2d

ENTITY axis2_placement_3d
  SUBTYPE OF (placement);
  axis          : OPTIONAL direction;
  ref_direction : OPTIONAL direction;
DERIVE
  p : LIST [3:3] OF direction := build_axes(axis,ref_direction);
WHERE
  WR1: SELF\placement.location.dim = 3;
  WR2: (NOT EXISTS(axis)) OR (axis.dim = 3);
```

```
  WR3: (NOT EXISTS(ref_direction)) OR (ref_direction.dim = 3);
  WR4: ((NOT EXISTS(axis)) OR (NOT EXISTS(ref_direction))) OR (
       cross_product(axis,ref_direction).magnitude > 0);
END_ENTITY; -- axis2_placement_3d

ENTITY b_spline_curve
  SUPERTYPE OF (ONEOF (uniform_curve, b_spline_curve_with_knots,
    quasi_uniform_curve, bezier_curve) ANDOR rational_b_spline_curve)
  SUBTYPE OF (bounded_curve);
  degree               : INTEGER;
  control_points_list  : LIST [2:?] OF cartesian_point;
  curve_form           : b_spline_curve_form;
  closed_curve         : LOGICAL;
  self_intersect       : LOGICAL;
DERIVE
  upper_index_on_control_points : INTEGER := SIZEOF(
                                  control_points_list) - 1;
  control_points                : ARRAY [0:
                                  upper_index_on_control_points] OF
                                  cartesian_point := list_to_array(
                                  control_points_list,0,
                                  upper_index_on_control_points);
WHERE
  WR1: ('CONFIG_CONTROL_DESIGN.UNIFORM_CURVE' IN TYPEOF(SELF)) OR
       ('CONFIG_CONTROL_DESIGN.QUASI_UNIFORM_CURVE' IN TYPEOF(SELF)) OR
       ('CONFIG_CONTROL_DESIGN.BEZIER_CURVE' IN TYPEOF(SELF)) OR
       ('CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE_WITH_KNOTS' IN TYPEOF(SELF));
END_ENTITY; -- b_spline_curve

ENTITY b_spline_curve_with_knots
  SUBTYPE OF (b_spline_curve);
  knot_multiplicities : LIST [2:?] OF INTEGER;
  knots               : LIST [2:?] OF parameter_value;
  knot_spec           : knot_type;
DERIVE
  upper_index_on_knots : INTEGER := SIZEOF(knots);
WHERE
  WR1: constraints_param_b_spline(degree,upper_index_on_knots,
       upper_index_on_control_points,knot_multiplicities,knots);
  WR2: SIZEOF(knot_multiplicities) = upper_index_on_knots;
END_ENTITY; -- b_spline_curve_with_knots

ENTITY b_spline_surface
  SUPERTYPE OF (ONEOF (b_spline_surface_with_knots, uniform_surface,
```

```
    quasi_uniform_surface, bezier_surface) ANDOR
    rational_b_spline_surface)
  SUBTYPE OF (bounded_surface);
  u_degree             : INTEGER;
  v_degree             : INTEGER;
  control_points_list  : LIST [2:?] OF LIST [2:?] OF cartesian_point;
  surface_form         : b_spline_surface_form;
  u_closed             : LOGICAL;
  v_closed             : LOGICAL;
  self_intersect       : LOGICAL;
DERIVE
  u_upper         : INTEGER := SIZEOF(control_points_list) - 1;
  v_upper         : INTEGER := SIZEOF(control_points_list[1]) - 1;
  control_points : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF
                       cartesian_point := make_array_of_array(
                      control_points_list,0,u_upper,0,v_upper);
WHERE
  WR1: ('CONFIG_CONTROL_DESIGN.UNIFORM_SURFACE' IN TYPEOF(SELF)) OR
       ('CONFIG_CONTROL_DESIGN.QUASI_UNIFORM_SURFACE' IN TYPEOF(SELF)) OR
       ('CONFIG_CONTROL_DESIGN.BEZIER_SURFACE' IN TYPEOF(SELF)) OR
       ('CONFIG_CONTROL_DESIGN.B_SPLINE_SURFACE_WITH_KNOTS' IN TYPEOF(SELF));
END_ENTITY; -- b_spline_surface

ENTITY b_spline_surface_with_knots
  SUBTYPE OF (b_spline_surface);
  u_multiplicities : LIST [2:?] OF INTEGER;
  v_multiplicities : LIST [2:?] OF INTEGER;
  u_knots          : LIST [2:?] OF parameter_value;
  v_knots          : LIST [2:?] OF parameter_value;
  knot_spec        : knot_type;
DERIVE
  knot_u_upper : INTEGER := SIZEOF(u_knots);
  knot_v_upper : INTEGER := SIZEOF(v_knots);
WHERE
  WR1: constraints_param_b_spline(SELF\b_spline_surface.u_degree,
       knot_u_upper,SELF\b_spline_surface.u_upper,u_multiplicities,
       u_knots);
  WR2: constraints_param_b_spline(SELF\b_spline_surface.v_degree,
       knot_v_upper,SELF\b_spline_surface.v_upper,v_multiplicities,
       v_knots);
  WR3: SIZEOF(u_multiplicities) = knot_u_upper;
  WR4: SIZEOF(v_multiplicities) = knot_v_upper;
END_ENTITY; -- b_spline_surface_with_knots
```

```
ENTITY bezier_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- bezier_curve

ENTITY bezier_surface
  SUBTYPE OF (b_spline_surface);
END_ENTITY; -- bezier_surface

ENTITY boundary_curve
  SUBTYPE OF (composite_curve_on_surface);
WHERE
  WR1: SELF\composite_curve.closed_curve;
END_ENTITY; -- boundary_curve

ENTITY bounded_curve
  SUPERTYPE OF (ONEOF (polyline, b_spline_curve, trimmed_curve,
    composite_curve))
  SUBTYPE OF (curve);
END_ENTITY; -- bounded_curve

ENTITY bounded_surface
  SUPERTYPE OF (ONEOF (b_spline_surface, rectangular_trimmed_surface,
    curve_bounded_surface, rectangular_composite_surface))
  SUBTYPE OF (surface);
END_ENTITY; -- bounded_surface

ENTITY brep_with_voids
  SUBTYPE OF (manifold_solid_brep);
  voids : SET [1:?] OF oriented_closed_shell;
END_ENTITY; -- brep_with_voids

ENTITY calendar_date
  SUBTYPE OF (date);
  day_component   : day_in_month_number;
  month_component : month_in_year_number;
WHERE
  WR1: valid_calendar_date(SELF);
END_ENTITY; -- calendar_date

ENTITY cartesian_point
  SUBTYPE OF (point);
  coordinates : LIST [1:3] OF length_measure;
END_ENTITY; -- cartesian_point
```

```
ENTITY cartesian_transformation_operator
  SUPERTYPE OF (ONEOF (cartesian_transformation_operator_2d,
    cartesian_transformation_operator_3d))
  SUBTYPE OF (geometric_representation_item,
    functionally_defined_transformation);
  axis1        : OPTIONAL direction;
  axis2        : OPTIONAL direction;
  local_origin : cartesian_point;
  scale        : OPTIONAL REAL;
DERIVE
  scl : REAL := NVL(scale,1);
WHERE
  WR1: scl > 0;
END_ENTITY; -- cartesian_transformation_operator

ENTITY cartesian_transformation_operator_2d
  SUBTYPE OF (cartesian_transformation_operator);
DERIVE
  u : LIST [2:2] OF direction := base_axis(2,SELF\
      cartesian_transformation_operator.axis1,SELF\
      cartesian_transformation_operator.axis2,?);
WHERE
  WR1: SELF\cartesian_transformation_operator.dim = 2;
END_ENTITY; -- cartesian_transformation_operator_2d

ENTITY cartesian_transformation_operator_3d
  SUBTYPE OF (cartesian_transformation_operator);
  axis3 : OPTIONAL direction;
DERIVE
  u : LIST [3:3] OF direction := base_axis(3,
      SELF\cartesian_transformation_operator.axis1,
      SELF\cartesian_transformation_operator.axis2, axis3);
WHERE
  WR1: SELF\cartesian_transformation_operator.dim = 3;
END_ENTITY; -- cartesian_transformation_operator_3d

ENTITY cc_design_approval
  SUBTYPE OF (approval_assignment);
  items : SET [1:?] OF approved_item;
END_ENTITY; -- cc_design_approval

ENTITY cc_design_certification
  SUBTYPE OF (certification_assignment);
  items : SET [1:?] OF certified_item;
```

```
END_ENTITY; -- cc_design_certification

ENTITY cc_design_contract
  SUBTYPE OF (contract_assignment);
  items : SET [1:?] OF contracted_item;
END_ENTITY; -- cc_design_contract

ENTITY cc_design_date_and_time_assignment
  SUBTYPE OF (date_and_time_assignment);
  items : SET [1:?] OF date_time_item;
WHERE
  WR1: cc_design_date_time_correlation(SELF);
END_ENTITY; -- cc_design_date_and_time_assignment

ENTITY cc_design_person_and_organization_assignment
  SUBTYPE OF (person_and_organization_assignment);
  items : SET [1:?] OF person_organization_item;
WHERE
  WR1: cc_design_person_and_organization_correlation(SELF);
END_ENTITY; -- cc_design_person_and_organization_assignment

ENTITY cc_design_security_classification
  SUBTYPE OF (security_classification_assignment);
  items : SET [1:?] OF classified_item;
END_ENTITY; -- cc_design_security_classification

ENTITY cc_design_specification_reference
  SUBTYPE OF (document_reference);
  items : SET [1:?] OF specified_item;
END_ENTITY; -- cc_design_specification_reference

ENTITY certification;
  name    : label;
  purpose : text;
  kind    : certification_type;
END_ENTITY; -- certification

ENTITY certification_assignment
  ABSTRACT SUPERTYPE;
  assigned_certification : certification;
END_ENTITY; -- certification_assignment

ENTITY certification_type;
  description : label;
```

```
END_ENTITY; -- certification_type

ENTITY change
  SUBTYPE OF (action_assignment);
  items : SET [1:?] OF work_item;
END_ENTITY; -- change

ENTITY change_request
  SUBTYPE OF (action_request_assignment);
  items : SET [1:?] OF change_request_item;
END_ENTITY; -- change_request

ENTITY circle
  SUBTYPE OF (conic);
  radius : positive_length_measure;
END_ENTITY; -- circle

ENTITY closed_shell
  SUBTYPE OF (connected_face_set);
END_ENTITY; -- closed_shell

ENTITY composite_curve
  SUBTYPE OF (bounded_curve);
  segments       : LIST [1:?] OF composite_curve_segment;
  self_intersect : LOGICAL;
DERIVE
  n_segments   : INTEGER := SIZEOF(segments);
  closed_curve : LOGICAL := segments[n_segments].transition <>
                    discontinuous;
WHERE
  WR1: ((NOT closed_curve) AND
       (SIZEOF (QUERY (temp <* segments |
       temp.transition = discontinuous)) = 1)) OR
       ((closed_curve) AND
       (SIZEOF (QUERY (temp <* segments |
       temp.transition = discontinuous)) = 0));
END_ENTITY; -- composite_curve

ENTITY composite_curve_on_surface
  SUPERTYPE OF (boundary_curve)
  SUBTYPE OF (composite_curve);
DERIVE
  basis_surface : SET [0:2] OF surface := get_basis_surface(SELF);
WHERE
```

```
  WR1: SIZEOF(basis_surface) > 0;
  WR2: constraints_composite_curve_on_surface(SELF);
END_ENTITY; -- composite_curve_on_surface

ENTITY composite_curve_segment;
  transition    : transition_code;
  same_sense    : BOOLEAN;
  parent_curve : curve;
INVERSE
  using_curves : BAG [1:?] OF composite_curve FOR segments;
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.BOUNDED_CURVE' IN TYPEOF(parent_curve);
END_ENTITY; -- composite_curve_segment

ENTITY configuration_design;
  configuration : configuration_item;
  design        : product_definition_formation;
UNIQUE
  UR1 : configuration, design;
END_ENTITY; -- configuration_design

ENTITY configuration_effectivity
  SUBTYPE OF (product_definition_effectivity);
  configuration : configuration_design;
UNIQUE
  UR1 : configuration, usage, id;
WHERE
  WR1: SELF\product_definition_effectivity.usage.
       relating_product_definition IN
       (USEDIN (SELF.configuration.design, 'CONFIG_CONTROL_DESIGN.' +
       'PRODUCT_DEFINITION.FORMATION') +
       relatives_of_product_definitions (bag_to_set (USEDIN
       (SELF.configuration.design, 'CONFIG_CONTROL_DESIGN.' +
       'PRODUCT_DEFINITION.FORMATION')), 'CONFIG_CONTROL_DESIGN.' +
       'PRODUCT_DEFINITION_USAGE'));
  WR2: 'CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION_USAGE' IN
       TYPEOF(SELF\product_definition_effectivity.usage);
END_ENTITY; -- configuration_effectivity

ENTITY configuration_item;
  id           : identifier;
  name         : label;
  description  : OPTIONAL text;
  item_concept : product_concept;
```

```
  purpose       : OPTIONAL label;
UNIQUE
  UR1 : id;
END_ENTITY; -- configuration_item

ENTITY conic
  SUPERTYPE OF (ONEOF (circle, ellipse, hyperbola, parabola))
  SUBTYPE OF (curve);
  position : axis2_placement;
END_ENTITY; -- conic

ENTITY conical_surface
  SUBTYPE OF (elementary_surface);
  radius     : length_measure;
  semi_angle : plane_angle_measure;
WHERE
  WR1: radius >= 0;
END_ENTITY; -- conical_surface

ENTITY connected_edge_set
  SUBTYPE OF (topological_representation_item);
  ces_edges : SET [1:?] OF edge;
END_ENTITY; -- connected_edge_set

ENTITY connected_face_set
  SUPERTYPE OF (ONEOF (closed_shell, open_shell))
  SUBTYPE OF (topological_representation_item);
  cfs_faces : SET [1:?] OF face;
END_ENTITY; -- connected_face_set

ENTITY context_dependent_shape_representation;
  representation_relation      : shape_representation_relationship;
  represented_product_relation : product_definition_shape;
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION_RELATIONSHIP' IN
       TYPEOF(SELF.represented_product_relation.definition);
END_ENTITY; -- context_dependent_shape_representation

ENTITY context_dependent_unit
  SUBTYPE OF (named_unit);
  name : label;
END_ENTITY; -- context_dependent_unit

ENTITY contract;
```

```
  name    : label;
  purpose : text;
  kind    : contract_type;
END_ENTITY; -- contract

ENTITY contract_assignment
  ABSTRACT SUPERTYPE;
  assigned_contract : contract;
END_ENTITY; -- contract_assignment

ENTITY contract_type;
  description : label;
END_ENTITY; -- contract_type

ENTITY conversion_based_unit
  SUBTYPE OF (named_unit);
  name              : label;
  conversion_factor : measure_with_unit;
END_ENTITY; -- conversion_based_unit

ENTITY coordinated_universal_time_offset;
  hour_offset   : hour_in_day;
  minute_offset : OPTIONAL minute_in_hour;
  sense         : ahead_or_behind;
END_ENTITY; -- coordinated_universal_time_offset

ENTITY curve
  SUPERTYPE OF (ONEOF (line, conic, pcurve, surface_curve, offset_curve_2d,
    offset_curve_3d, curve_replica))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- curve

ENTITY curve_bounded_surface
  SUBTYPE OF (bounded_surface);
  basis_surface  : surface;
  boundaries     : SET [1:?] OF boundary_curve;
  implicit_outer : BOOLEAN;
WHERE
  WR1: NOT (implicit_outer AND
       ('CONFIG_CONTROL_DESIGN.OUTER_BOUNDARY_CURVE' IN TYPEOF(boundaries)));
  WR2: (NOT implicit_outer) OR
       ('CONFIG_CONTROL_DESIGN.BOUNDED_SURFACE' IN TYPEOF(basis_surface));
  WR3: SIZEOF (QUERY (temp <* boundaries |
       'CONFIG_CONTROL_DESIGN.OUTER_BOUNDARY_CURVE' IN TYPEOF(temp))) <= 1;
```

```
  WR4: SIZEOF (QUERY (temp <* boundaries |
       temp\composite_curve_on_surface.basis_surface[1] :<>:
       SELF.basis_surface)) = 0;
END_ENTITY; -- curve_bounded_surface

ENTITY curve_replica
  SUBTYPE OF (curve);
  parent_curve    : curve;
  transformation : cartesian_transformation_operator;
WHERE
  WR1: transformation.dim = parent_curve.dim;
  WR2: acyclic_curve_replica(SELF,parent_curve);
END_ENTITY; -- curve_replica

ENTITY cylindrical_surface
  SUBTYPE OF (elementary_surface);
  radius : positive_length_measure;
END_ENTITY; -- cylindrical_surface

ENTITY date
  SUPERTYPE OF (ONEOF (calendar_date, ordinal_date,
    week_of_year_and_day_date));
  year_component : year_number;
END_ENTITY; -- date

ENTITY date_and_time;
  date_component : date;
  time_component : local_time;
END_ENTITY; -- date_and_time

ENTITY date_and_time_assignment
  ABSTRACT SUPERTYPE;
  assigned_date_and_time : date_and_time;
  role                   : date_time_role;
END_ENTITY; -- date_and_time_assignment

ENTITY date_time_role;
  name : label;
END_ENTITY; -- date_time_role

ENTITY dated_effectivity
  SUBTYPE OF (effectivity);
  effectivity_start_date : date_and_time;
  effectivity_end_date   : OPTIONAL date_and_time;
```

```
END_ENTITY; -- dated_effectivity

ENTITY definitional_representation
  SUBTYPE OF (representation);
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT' IN
       TYPEOF (SELF\representation.context_of_items);
END_ENTITY; -- definitional_representation

ENTITY degenerate_pcurve
  SUBTYPE OF (point);
  basis_surface      : surface;
  reference_to_curve : definitional_representation;
WHERE
  WR1: SIZEOF(reference_to_curve\representation.items) = 1;
  WR2: 'CONFIG_CONTROL_DESIGN.CURVE' IN
       TYPEOF (reference_to_curve\representation.items[1]);
  WR3: reference_to_curve\representation.items[1]\
       geometric_representation_item.dim = 2;
END_ENTITY; -- degenerate_pcurve

ENTITY design_context
  SUBTYPE OF (product_definition_context);
WHERE
  WR1: SELF.life_cycle_stage = 'design';
END_ENTITY; -- design_context

ENTITY design_make_from_relationship
  SUBTYPE OF (product_definition_relationship);
END_ENTITY; -- design_make_from_relationship

ENTITY dimensional_exponents;
  length_exponent                    : REAL;
  mass_exponent                      : REAL;
  time_exponent                      : REAL;
  electric_current_exponent          : REAL;
  thermodynamic_temperature_exponent : REAL;
  amount_of_substance_exponent       : REAL;
  luminous_intensity_exponent        : REAL;
END_ENTITY; -- dimensional_exponents

ENTITY directed_action
  SUBTYPE OF (executed_action);
  directive : action_directive;
```

```
END_ENTITY; -- directed_action

ENTITY direction
  SUBTYPE OF (geometric_representation_item);
  direction_ratios : LIST [2:3] OF REAL;
WHERE
  WR1: SIZEOF (QUERY (tmp <* direction_ratios | tmp <> 0)) > 0;
END_ENTITY; -- direction

ENTITY document;
  id          : identifier;
  name        : label;
  description : text;
  kind        : document_type;
UNIQUE
  UR1 : id;
END_ENTITY; -- document

ENTITY document_reference
  ABSTRACT SUPERTYPE;
  assigned_document : document;
  source            : label;
END_ENTITY; -- document_reference

ENTITY document_relationship;
  name               : label;
  description        : text;
  relating_document  : document;
  related_document   : document;
END_ENTITY; -- document_relationship

ENTITY document_type;
  product_data_type : label;
END_ENTITY; -- document_type

ENTITY document_usage_constraint;
  source                : document;
  subject_element       : label;
  subject_element_value : text;
END_ENTITY; -- document_usage_constraint

ENTITY document_with_class
  SUBTYPE OF (document);
  class : identifier;
```

```
END_ENTITY; -- document_with_class

ENTITY edge
  SUPERTYPE OF (ONEOF (edge_curve, oriented_edge))
  SUBTYPE OF (topological_representation_item);
  edge_start : vertex;
  edge_end   : vertex;
END_ENTITY; -- edge

ENTITY edge_based_wireframe_model
  SUBTYPE OF (geometric_representation_item);
  ebwm_boundary : SET [1:?] OF connected_edge_set;
END_ENTITY; -- edge_based_wireframe_model

ENTITY edge_based_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1:  SIZEOF (QUERY (it <* SELF\representation.items |
        NOT (SIZEOF(['CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
        'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF (it)) = 1
        ))) = 0;
  WR2:  SIZEOF (QUERY (it <* SELF\representation.items |
        SIZEOF(['CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL',
        'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'] * TYPEOF (it)) = 1
        )) >= 1;
  WR3:  SIZEOF (QUERY (ebwm <*
        QUERY (it <* SELF\representation.items |
        'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (eb <*
        ebwm\edge_based_wireframe_model.ebwm_boundary |
        NOT (SIZEOF (QUERY (edges <* eb.ces_edges |
        NOT ('CONFIG_CONTROL_DESIGN.EDGE_CURVE' IN TYPEOF (edges)))) = 0
        ))) = 0))) = 0;
  WR4:  SIZEOF (QUERY (ebwm <*
        QUERY (it <* SELF\representation.items |
        'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (eb <*
        ebwm\edge_based_wireframe_model.ebwm_boundary |
        NOT (SIZEOF (QUERY (pline_edges <* QUERY (edges <* eb.ces_edges |
        'CONFIG_CONTROL_DESIGN.POLYLINE' IN
        TYPEOF (edges\edge_curve.edge_geometry)) |
        NOT (SIZEOF (pline_edges\edge_curve.edge_geometry\polyline.points)
        > 2))) = 0))) = 0))) = 0;
```

```
WR5:  SIZEOF (QUERY (ebwm <*
      QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (eb <*
      ebwm\edge_based_wireframe_model.ebwm_boundary |
      NOT (SIZEOF (QUERY (edges <* eb.ces_edges |
      NOT (('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN
      TYPEOF (edges.edge_start)) AND
      ('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN
      TYPEOF (edges.edge_end))))) = 0))) = 0))) = 0;
WR6:  SIZEOF (QUERY (ebwm <*
      QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (eb <*
      ebwm\edge_based_wireframe_model.ebwm_boundary |
      NOT (SIZEOF (QUERY (edges <* eb.ces_edges |
      NOT (valid_wireframe_edge_curve
      (edges\edge_curve.edge_geometry, 'CONFIG_CONTROL_DESIGN')))) =
      0))) = 0))) = 0;
WR7:  SIZEOF (QUERY (ebwm <*
      QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (eb <*
      ebwm\edge_based_wireframe_model.ebwm_boundary |
      NOT (SIZEOF (QUERY (edges <* eb.ces_edges |
      NOT((valid_wireframe_vertex_point
      (edges.edge_start\vertex_point.vertex_geometry,
      'CONFIG_CONTROL_DESIGN')) AND
      (valid_wireframe_vertex_point
      (edges.edge_end\vertex_point.vertex_geometry,
      'CONFIG_CONTROL_DESIGN'))))) = 0))) = 0))) = 0;
WR8:  SIZEOF (QUERY (ebwm <*
      QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (eb <*
      ebwm\edge_based_wireframe_model.ebwm_boundary |
      NOT (SIZEOF (QUERY (con_edges <* QUERY (edges <* eb.ces_edges |
      'CONFIG_CONTROL_DESIGN.CONIC' IN
      TYPEOF (edges\edge_curve.edge_geometry)) |
      NOT ('CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D' IN
      TYPEOF (con_edges\edge_curve.edge_geometry\conic.position)))) = 0
      ))) = 0))) = 0;
WR9:  SIZEOF (QUERY (mi <* QUERY (item <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.MAPPED_ITEM' IN TYPEOF (item)) |
```

```
        NOT ('CONFIG_CONTROL_DESIGN.' +
        'EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION'
        IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
        = 0;
END_ENTITY; -- edge_based_wireframe_shape_representation

ENTITY edge_curve
  SUBTYPE OF (edge, geometric_representation_item);
  edge_geometry : curve;
  same_sense    : BOOLEAN;
END_ENTITY; -- edge_curve

ENTITY edge_loop
  SUBTYPE OF (loop, path);
DERIVE
  ne : INTEGER := SIZEOF(SELF\path.edge_list);
WHERE
  WR1: SELF\path.edge_list[1].edge_start :=:
       SELF\path.edge_list[ne].edge_end;
END_ENTITY; -- edge_loop

ENTITY effectivity
  SUPERTYPE OF (ONEOF (serial_numbered_effectivity, dated_effectivity,
    lot_effectivity));
  id : identifier;
END_ENTITY; -- effectivity

ENTITY elementary_surface
  SUPERTYPE OF (ONEOF (plane, cylindrical_surface, conical_surface,
    spherical_surface, toroidal_surface))
  SUBTYPE OF (surface);
  position : axis2_placement_3d;
END_ENTITY; -- elementary_surface

ENTITY ellipse
  SUBTYPE OF (conic);
  semi_axis_1 : positive_length_measure;
  semi_axis_2 : positive_length_measure;
END_ENTITY; -- ellipse

ENTITY evaluated_degenerate_pcurve
  SUBTYPE OF (degenerate_pcurve);
  equivalent_point : cartesian_point;
END_ENTITY; -- evaluated_degenerate_pcurve
```

```
ENTITY executed_action
  SUBTYPE OF (action);
END_ENTITY; -- executed_action

ENTITY face
  SUPERTYPE OF (ONEOF (face_surface, oriented_face))
  SUBTYPE OF (topological_representation_item);
  bounds : SET [1:?] OF face_bound;
WHERE
  WR1: NOT mixed_loop_type_set(list_to_set(list_face_loops(SELF)));
  WR2: SIZEOF (QUERY (temp <* bounds |
       'CONFIG_CONTROL_DESIGN.FACE_OUTER_BOUND' IN TYPEOF(temp))) <= 1;
END_ENTITY; -- face

ENTITY face_bound
  SUBTYPE OF (topological_representation_item);
  bound       : loop;
  orientation : BOOLEAN;
END_ENTITY; -- face_bound

ENTITY face_outer_bound
  SUBTYPE OF (face_bound);
END_ENTITY; -- face_outer_bound

ENTITY face_surface
  SUBTYPE OF (face, geometric_representation_item);
  face_geometry : surface;
  same_sense    : BOOLEAN;
END_ENTITY; -- face_surface

ENTITY faceted_brep
  SUBTYPE OF (manifold_solid_brep);
END_ENTITY; -- faceted_brep

ENTITY faceted_brep_shape_representation
SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* items |
       NOT (SIZEOF(['CONFIG_CONTROL_DESIGN.FACETED_BREP',
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
       'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1))) = 0;
  WR2: SIZEOF (QUERY (it <* items |
       SIZEOF(['CONFIG_CONTROL_DESIGN.FACETED_BREP',
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1)) > 0;
```

```
  WR3: SIZEOF (QUERY (fbrep <* QUERY ( it <* items |
       'CONFIG_CONTROL_DESIGN.FACETED_BREP' IN TYPEOF(it)) |
       NOT (SIZEOF (QUERY (csh <*
       msb_shells (fbrep, 'CONFIG_CONTROL_DESIGN') |
       NOT (SIZEOF (QUERY (fcs <* csh.cfs_faces |
       NOT (('CONFIG_CONTROL_DESIGN.FACE_SURFACE' IN TYPEOF (fcs))
       AND
       (('CONFIG_CONTROL_DESIGN.PLANE' IN TYPEOF
       (fcs\face_surface.face_geometry))
       AND
       ('CONFIG_CONTROL_DESIGN.CARTESIAN_POINT' IN
       TYPEOF (fcs\face_surface.face_geometry\plane.position.location))))))
       = 0))) = 0))) = 0;
  WR4: SIZEOF (QUERY (fbrep <* QUERY ( it <* items |
       'CONFIG_CONTROL_DESIGN.FACETED_BREP' IN TYPEOF(it)) |
       NOT (SIZEOF (QUERY (csh <*
       msb_shells (fbrep, 'CONFIG_CONTROL_DESIGN') |
       NOT (SIZEOF (QUERY (fcs <* csh.cfs_faces |
       NOT (SIZEOF (QUERY (bnds <* fcs.bounds |
       'CONFIG_CONTROL_DESIGN.FACE_OUTER_BOUND' IN TYPEOF(bnds))) = 1)))
       = 0))) = 0))) = 0;
  WR5: SIZEOF (QUERY (msb <* QUERY (it <* items |
       'CONFIG_CONTROL_DESIGN.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
       'CONFIG_CONTROL_DESIGN.ORIENTED_CLOSED_SHELL' IN
       TYPEOF (msb\manifold_solid_brep.outer))) = 0;
  WR6: SIZEOF (QUERY (brv <* QUERY (it <* items |
       'CONFIG_CONTROL_DESIGN.BREP_WITH_VOIDS' IN TYPEOF(it)) |
       NOT (SIZEOF (QUERY (csh <* brv\brep_with_voids.voids |
       csh\oriented_closed_shell.orientation)) = 0))) = 0;
  WR7: SIZEOF (QUERY (mi <*  QUERY (it <* items |
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) |
       NOT ('CONFIG_CONTROL_DESIGN.FACETED_BREP_SHAPE_REPRESENTATION' IN
       TYPEOF(mi\mapped_item.mapping_source.mapped_representation))))
       = 0;
END_ENTITY; -- faceted_brep_shape_representation

ENTITY functionally_defined_transformation;
  name        : label;
  description : text;
END_ENTITY; -- functionally_defined_transformation

ENTITY geometric_curve_set
  SUBTYPE OF (geometric_set);
WHERE
```

```
  WR1: SIZEOF (QUERY (temp <* SELF\geometric_set.elements |
       'CONFIG_CONTROL_DESIGN.SURFACE' IN TYPEOF(temp))) = 0;
END_ENTITY; -- geometric_curve_set

ENTITY geometric_representation_context
  SUBTYPE OF (representation_context);
  coordinate_space_dimension : dimension_count;
END_ENTITY; -- geometric_representation_context

ENTITY geometric_representation_item
  SUPERTYPE OF (ONEOF (point, direction, vector, placement,
    cartesian_transformation_operator, curve, surface, edge_curve,
    face_surface, vertex_point, poly_loop, solid_model,
    shell_based_surface_model, shell_based_wireframe_model,
    edge_based_wireframe_model, geometric_set))
  SUBTYPE OF (representation_item);
DERIVE
  dim : dimension_count := dimension_of(SELF);
WHERE
  WR1: SIZEOF (QUERY (using_rep <* using_representations (SELF) |
       NOT ('CONFIG_CONTROL_DESIGN.GEOMETRIC_REPRESENTATION_CONTEXT' IN
       TYPEOF (using_rep.context_of_items)))) = 0;
END_ENTITY; -- geometric_representation_item

ENTITY geometric_set
  SUPERTYPE OF (geometric_curve_set)
  SUBTYPE OF (geometric_representation_item);
  elements : SET [1:?] OF geometric_set_select;
END_ENTITY; -- geometric_set

ENTITY geometrically_bounded_surface_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* items |
       NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.GEOMETRIC_SET',
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
       'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF (it)) = 1))) = 0;
  WR2: SIZEOF (QUERY (it <* items |
       SIZEOF (['CONFIG_CONTROL_DESIGN.GEOMETRIC_SET',
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'] * TYPEOF (it)) = 1)) > 0;
  WR3: SIZEOF (QUERY (mi <* QUERY (it <* items |
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM' IN TYPEOF (it)) |
       NOT ('CONFIG_CONTROL_DESIGN.' +
       'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION'
```

```
      IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
      = 0;
  WR4: SIZEOF (QUERY (gs <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.GEOMETRIC_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (pnt <* QUERY (gsel <*
      gs\geometric_set.elements |
      'CONFIG_CONTROL_DESIGN.POINT' IN TYPEOF (gsel)) |
      NOT (gbsf_check_point(pnt, 'CONFIG_CONTROL_DESIGN')))) = 0))) = 0;
  WR5: SIZEOF (QUERY (gs <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.GEOMETRIC_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cv <* QUERY (gsel <*
      gs\geometric_set.elements |
      'CONFIG_CONTROL_DESIGN.CURVE' IN TYPEOF (gsel)) |
      NOT (gbsf_check_curve(cv, 'CONFIG_CONTROL_DESIGN')))) = 0))) = 0;
  WR6: SIZEOF (QUERY (gs <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.GEOMETRIC_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (sf <* QUERY (gsel <*
      gs\geometric_set.elements |
      'CONFIG_CONTROL_DESIGN.SURFACE' IN TYPEOF (gsel)) |
      NOT (gbsf_check_surface(sf, 'CONFIG_CONTROL_DESIGN')))) = 0))) = 0;
END_ENTITY; -- geometrically_bounded_surface_shape_representation

ENTITY geometrically_bounded_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* SELF\representation.items |
      NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET',
      'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
      'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1))) = 0;
  WR2: SIZEOF (QUERY (it <* SELF\representation.items |
      SIZEOF(['CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET',
      'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'] * TYPEOF (it)) = 1
      )) >= 1;
  WR3: SIZEOF (QUERY (gcs <* QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (elem <* gcs\geometric_set.elements |
      NOT (valid_geometrically_bounded_wf_curve
      (elem, 'CONFIG_CONTROL_DESIGN')))) = 0))) = 0;
  WR4: SIZEOF (QUERY (gcs <* QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (pnts <* QUERY (elem <*
      gcs\geometric_set.elements |
      'CONFIG_CONTROL_DESIGN.POINT' IN TYPEOF (elem)) |
      NOT (valid_geometrically_bounded_wf_point
```

```
      (pnts, 'CONFIG_CONTROL_DESIGN')))) = 0))) = 0;
  WR5: SIZEOF (QUERY (gcs <* QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cnc <* QUERY (elem <*
      gcs\geometric_set.elements |
      'CONFIG_CONTROL_DESIGN.CONIC' IN TYPEOF (elem)) |
      NOT ('CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D' IN TYPEOF
      (cnc\conic.position)))) = 0))) = 0;
  WR6: SIZEOF (QUERY (gcs <* QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (pline <* QUERY (elem <*
      gcs\geometric_set.elements |
      'CONFIG_CONTROL_DESIGN.POLYLINE' IN TYPEOF(elem)) |
      NOT (SIZEOF (pline\polyline.points) > 2))) = 0))) = 0;
  WR7: SIZEOF (QUERY (mi <* QUERY (item <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.MAPPED_ITEM' IN TYPEOF (item)) |
      NOT ('CONFIG_CONTROL_DESIGN.' +
      'GEOMETRICALLY_BOUNDED_WIREFRAME_REPRESENTATION'
      IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
      = 0;
END_ENTITY; -- geometrically_bounded_wireframe_shape_representation

ENTITY global_uncertainty_assigned_context
  SUBTYPE OF (representation_context);
  uncertainty : SET [1:?] OF uncertainty_measure_with_unit;
END_ENTITY; -- global_uncertainty_assigned_context

ENTITY global_unit_assigned_context
  SUBTYPE OF (representation_context);
  units : SET [1:?] OF unit;
END_ENTITY; -- global_unit_assigned_context

ENTITY hyperbola
  SUBTYPE OF (conic);
  semi_axis      : positive_length_measure;
  semi_imag_axis : positive_length_measure;
END_ENTITY; -- hyperbola

ENTITY intersection_curve
  SUBTYPE OF (surface_curve);
WHERE
  WR1: SIZEOF(SELF\surface_curve.associated_geometry) = 2;
  WR2: associated_surface(SELF\surface_curve.associated_geometry[1])
      <> associated_surface(SELF\surface_curve.associated_geometry[2]);
```

```
END_ENTITY; -- intersection_curve

ENTITY item_defined_transformation;
  name              : label;
  description       : text;
  transform_item_1 : representation_item;
  transform_item_2 : representation_item;
END_ENTITY; -- item_defined_transformation

ENTITY length_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.LENGTH_UNIT' IN
       TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY; -- length_measure_with_unit

ENTITY length_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 1) AND
       (SELF\named_unit.dimensions.mass_exponent = 0) AND
       (SELF\named_unit.dimensions.time_exponent = 0) AND
       (SELF\named_unit.dimensions.electric_current_exponent = 0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent = 0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- length_unit

ENTITY line
  SUBTYPE OF (curve);
  pnt : cartesian_point;
  dir : vector;
WHERE
  WR1: dir.dim = pnt.dim;
END_ENTITY; -- line

ENTITY local_time;
  hour_component   : hour_in_day;
  minute_component : OPTIONAL minute_in_hour;
  second_component : OPTIONAL second_in_minute;
  zone             : coordinated_universal_time_offset;
WHERE
  WR1: valid_time (SELF);
END_ENTITY; -- local_time
```

```
ENTITY loop
  SUPERTYPE OF (ONEOF (vertex_loop, edge_loop, poly_loop))
  SUBTYPE OF (topological_representation_item);
END_ENTITY; -- loop

ENTITY lot_effectivity
  SUBTYPE OF (effectivity);
  effectivity_lot_id   : identifier;
  effectivity_lot_size : measure_with_unit;
END_ENTITY; -- lot_effectivity

ENTITY manifold_solid_brep
  SUBTYPE OF (solid_model);
  outer : closed_shell;
END_ENTITY; -- manifold_solid_brep

ENTITY manifold_surface_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* items |
       NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL',
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
       'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF (it)) = 1))) = 0;
  WR2: SIZEOF (QUERY (it <* items |
       SIZEOF (['CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL',
       'CONFIG_CONTROL_DESIGN.MAPPED ITEM'] * TYPEOF (it)) = 1)) > 0;
  WR3: SIZEOF (QUERY (mi <* QUERY (it <* items |
       'CONFIG_CONTROL_DESIGN.MAPPED ITEM' IN TYPEOF (it)) |
       NOT ('CONFIG_CONTROL_DESIGN.MANIFOLD_SURFACE_SHAPE_REPRESENTATION'
       IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
       = 0;
  WR4: SIZEOF (QUERY (sbsm <* QUERY (it <* items |
       'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (sh <*
       sbsm\shell_based_surface_model.sbsm_boundary |
       NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.OPEN_SHELL',
       'CONFIG_CONTROL_DESIGN.CLOSED_SHELL']
       * TYPEOF (sh)) = 1))) = 0))) = 0;
  WR5: SIZEOF (QUERY (sbsm <* QUERY (it <* items |
       'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (cfs <*
       sbsm\shell_based_surface_model.sbsm_boundary |
       NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
       NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.FACE_SURFACE',
```

```
      'CONFIG_CONTROL_DESIGN.ORIENTED_FACE'] * TYPEOF (fa)) = 1))) = 0)))
      = 0))) = 0;
WR6: SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (f_sf <* QUERY (fa <* cfs.cfs_faces |
      'CONFIG_CONTROL_DESIGN.FACE_SURFACE' IN TYPEOF (fa)) |
      NOT (('CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (f_sf))
      OR
      (SIZEOF (['CONFIG_CONTROL_DESIGN.OFFSET_SURFACE',
      'CONFIG_CONTROL_DESIGN.SURFACE_REPLICA'] * TYPEOF
      (f_sf\face_surface.face_geometry)) = 1)))) = 0))) = 0))) = 0;
WR7: SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
      NOT (('CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (basis_surface_check(fa\face_surface.face_geometry,
      'CONFIG_CONTROL_DESIGN'))))) = 0))) = 0))) = 0;
WR8: SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
      NOT (('CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (bnds <* fa.bounds |
      NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.EDGE_LOOP',
      'CONFIG_CONTROL_DESIGN.VERTEX_LOOP']
      * TYPEOF (bnds.bound)) = 1))) = 0)))) = 0))) = 0))) = 0;
WR9: SIZEOF (QUERY (sbsm <* QUERY (it <* items|
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
      NOT (('CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
      'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (bnds)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT ('CONFIG_CONTROL_DESIGN.EDGE_CURVE' IN TYPEOF (oe.edge_element))))
```

```
      = 0))) = 0)))) = 0))) = 0))) = 0;
WR10: SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
      NOT (('CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
      'CONFIG_CONTROL_DESIGN.EDGE LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe_cv <* QUERY (oe <*
      elp_fbnds.bound\path.edge_list |
      'CONFIG_CONTROL_DESIGN.EDGE_CURVE' IN TYPEOF (oe.edge_element)) |
      NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.CURVE_REPLICA',
      'CONFIG_CONTROL_DESIGN.OFFSET_CURVE_3D',
      'CONFIG_CONTROL_DESIGN.SURFACE_CURVE'] *
      TYPEOF (oe_cv.edge_element\edge_curve.edge_geometry))
      = 1))) = 0))) = 0)))) = 0))) = 0))) = 0;
WR11: SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
      NOT (('CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
      'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT (basis_curve_check (oe.edge_element\edge_curve.edge_geometry,
      'CONFIG_CONTROL_DESIGN')))) = 0))) = 0)))) = 0))) = 0))) = 0;
WR12: SIZEOF (QUERY(sbsm <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
      NOT (('CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
      'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (bnds)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list|
      NOT (('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN TYPEOF
      (oe.edge_element.edge_start))
      AND
      ('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN
```

```
      TYPEOF (oe.edge_element.edge_end))))) = 0))) = 0)))) = 0))) = 0))) = 0;
 WR13: SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
      NOT (('CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa. bounds |
      'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT ((SIZEOF (['CONFIG_CONTROL_DESIGN.CARTESIAN_POINT',
      'CONFIG_CONTROL_DESIGN.DEGENERATE_PCURVE',
      'CONFIG_CONTROL_DESIGN.POINT_ON_CURVE',
      'CONFIG_CONTROL_DESIGN.POINT_ON_SURFACE'] * TYPEOF
      (oe.edge_element.edge_start\vertex_point.vertex_geometry)) = 1)
      AND
      (SIZEOF (['CONFIG_CONTROL_DESIGN.CARTESIAN_POINT',
      'CONFIG_CONTROL_DESIGN.DEGENERATE_PCURVE',
      'CONFIG_CONTROL_DESIGN.POINT_ON_CURVE',
      'CONFIG_CONTROL_DESIGN.POINT_ON_SURFACE'] * TYPEOF
      (oe.edge_element.edge_end\vertex_point.vertex_geometry)) = 1
      ))))) = 0))) = 0)))) = 0))) = 0))) = 0;
END_ENTITY; -- manifold_surface_shape_representation

ENTITY mapped_item
  SUBTYPE OF (representation_item);
  mapping_source : representation_map;
  mapping_target : representation_item;
WHERE
  WR1: acyclic_mapped_representation(using_representations(SELF), [SELF]);
END_ENTITY; -- mapped_item

ENTITY mass_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.MASS_UNIT' IN
       TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY; -- mass_measure_with_unit

ENTITY mass_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 0) AND
```

```
        (SELF\named_unit.dimensions.mass_exponent = 1) AND
        (SELF\named_unit.dimensions.time_exponent = 0) AND
        (SELF\named_unit.dimensions.electric_current_exponent = 0) AND
        (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0) AND
        (SELF\named_unit.dimensions.amount_of_substance_exponent = 0) AND
        (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- mass_unit

ENTITY measure_with_unit
  SUPERTYPE OF (ONEOF (length_measure_with_unit, mass_measure_with_unit,
    plane_angle_measure_with_unit, solid_angle_measure_with_unit,
    area_measure_with_unit, volume_measure_with_unit));
  value_component : measure_value;
  unit_component  : unit;
WHERE
  WR1: valid_units(SELF);
END_ENTITY; -- measure_with_unit

ENTITY mechanical_context
  SUBTYPE OF (product_context);
WHERE
  WR1: SELF.discipline_type = 'mechanical';
END_ENTITY; -- mechanical_context

ENTITY named_unit
  SUPERTYPE OF (ONEOF (si_unit, conversion_based_unit,
    context_dependent_unit) ANDOR ONEOF (length_unit, mass_unit,
    plane_angle_unit, solid_angle_unit, area_unit, volume_unit));
  dimensions : dimensional_exponents;
END_ENTITY; -- named_unit

ENTITY next_assembly_usage_occurrence
  SUBTYPE OF (assembly_component_usage);
END_ENTITY; -- next_assembly_usage_occurrence

ENTITY offset_curve_2d
  SUBTYPE OF (curve);
  basis_curve    : curve;
  distance       : length_measure;
  self_intersect : LOGICAL;
WHERE
  WR1: basis_curve.dim = 2;
END_ENTITY; -- offset_curve_2d
```

```
ENTITY offset_curve_3d
  SUBTYPE OF (curve);
  basis_curve    : curve;
  distance       : length_measure;
  self_intersect : LOGICAL;
  ref_direction  : direction;
WHERE
  WR1: (basis_curve.dim = 3) AND (ref_direction.dim = 3);
END_ENTITY; -- offset_curve_3d

ENTITY offset_surface
  SUBTYPE OF (surface);
  basis_surface  : surface;
  distance       : length_measure;
  self_intersect : LOGICAL;
END_ENTITY; -- offset_surface

ENTITY open_shell
  SUBTYPE OF (connected_face_set);
END_ENTITY; -- open_shell

ENTITY ordinal_date
SUBTYPE OF (date);
  day_component : day_in_year_number;
WHERE
  WR1: (NOT leap_year(SELF.year_component) AND {1 <= day_component <= 365})
       OR
       (leap_year(SELF.year_component) AND {1 <= day_component <= 366});
END_ENTITY; -- ordinal_date

ENTITY organization;
  id          : OPTIONAL identifier;
  name        : label;
  description : text;
END_ENTITY; -- organization

ENTITY organization_relationship;
  name                  : label;
  description           : text;
  relating_organization : organization;
  related_organization  : organization;
END_ENTITY; -- organization_relationship

ENTITY organizational_address
```

```
  SUBTYPE OF (address);
  organizations : SET [1:?] OF organization;
  description   : text;
END_ENTITY; -- organizational_address

ENTITY organizational_project;
  name                       : label;
  description                : text;
  responsible_organizations : SET [1:?] OF organization;
END_ENTITY; -- organizational_project

ENTITY oriented_closed_shell
  SUBTYPE OF (closed_shell);
  closed_shell_element : closed_shell;
  orientation          : BOOLEAN;
DERIVE
  SELF\connected_face_set.cfs_faces : SET [1:?] OF face :=
                                      conditional_reverse(SELF.orientation,
                                      SELF.closed_shell_element.cfs_faces);
WHERE
  WR1: NOT ('CONFIG_CONTROL_DESIGN.ORIENTED_CLOSED_SHELL' IN
       TYPEOF(SELF.closed_shell_element));
END_ENTITY; -- oriented_closed_shell

ENTITY oriented_edge
  SUBTYPE OF (edge);
  edge_element : edge;
  orientation  : BOOLEAN;
DERIVE
  SELF\edge.edge_start : vertex := boolean_choose(SELF.orientation,
                         SELF.edge_element.edge_start,
                         SELF.edge_element.edge_end);
  SELF\edge.edge_end   : vertex := boolean_choose(SELF.orientation,
                         SELF.edge_element.edge_end,
                         SELF.edge_element.edge_start);
WHERE
  WR1: NOT ('CONFIG_CONTROL_DESIGN.ORIENTED_EDGE' IN
       TYPEOF (SELF.edge_element));
END_ENTITY; -- oriented_edge

ENTITY oriented_face
  SUBTYPE OF (face);
  face_element : face;
  orientation  : BOOLEAN;
```

```
DERIVE
  SELF\face.bounds : SET [1:?] OF face_bound := conditional_reverse
                     (SELF.orientation, SELF.face_element.bounds);
WHERE
  WR1: NOT ('CONFIG_CONTROL_DESIGN.ORIENTED_FACE' IN
       TYPEOF(SELF.face_element));
END_ENTITY; -- oriented_face

ENTITY oriented_open_shell
  SUBTYPE OF (open_shell);
  open_shell_element : open_shell;
  orientation        : BOOLEAN;
DERIVE
  SELF\connected_face_set.cfs_faces : SET [1:?] OF face :=
                                      conditional_reverse(SELF.orientation,
                                      SELF.open_shell_element.cfs_faces);
WHERE
  WR1: NOT ('CONFIG_CONTROL_DESIGN.ORIENTED_OPEN_SHELL' IN
       TYPEOF (SELF.open_shell_element));
END_ENTITY; -- oriented_open_shell

ENTITY oriented_path
  SUBTYPE OF (path);
  path_element : path;
  orientation  : BOOLEAN;
DERIVE
  SELF\path.edge_list : LIST [1:?] OF UNIQUE oriented_edge :=
                        conditional_reverse(SELF.orientation,
                        SELF.path_element.edge_list);
WHERE
  WR1: NOT ('CONFIG_CONTROL_DESIGN.ORIENTED_PATH' IN
       TYPEOF (SELF.path_element));
END_ENTITY; -- oriented_path

ENTITY outer_boundary_curve
  SUBTYPE OF (boundary_curve);
END_ENTITY; -- outer_boundary_curve

ENTITY parabola
  SUBTYPE OF (conic);
  focal_dist : length_measure;
WHERE
  WR1: focal_dist <> 0;
END_ENTITY; -- parabola
```

```
ENTITY parametric_representation_context
  SUBTYPE OF (representation_context);
END_ENTITY; -- parametric_representation_context

ENTITY path
  SUPERTYPE OF (ONEOF (edge_loop, oriented_path))
  SUBTYPE OF (topological_representation_item);
  edge_list : LIST [1:?] OF UNIQUE oriented_edge;
WHERE
  WR1: path_head_to_tail (SELF);
END_ENTITY; -- path

ENTITY pcurve
  SUBTYPE OF (curve);
  basis_surface      : surface;
  reference_to_curve : definitional_representation;
WHERE
  WR1: SIZEOF (reference_to_curve\representation.items) = 1;
  WR2: 'CONFIG_CONTROL_DESIGN.CURVE' IN
       TYPEOF(reference_to_curve\representation.items[1]);
  WR3: reference_to_curve\representation.items[1]\
       geometric_representation_item.dim = 2;
END_ENTITY; -- pcurve

ENTITY person;
  id            : identifier;
  last_name     : OPTIONAL label;
  first_name    : OPTIONAL label;
  middle_names  : OPTIONAL LIST [1:?] OF label;
  prefix_titles : OPTIONAL LIST [1:?] OF label;
  suffix_titles : OPTIONAL LIST [1:?] OF label;
UNIQUE
  UR1 : id;
WHERE
  WR1: EXISTS (last_name) OR EXISTS (first_name);
END_ENTITY; -- person

ENTITY person_and_organization;
  the_person       : person;
  the_organization : organization;
END_ENTITY; -- person_and_organization

ENTITY person_and_organization_assignment
  ABSTRACT SUPERTYPE;
```

```
  assigned_person_and_organization : person_and_organization;
  role                             : person_and_organization_role;
END_ENTITY; -- person_and_organization_assignment

ENTITY person_and_organization_role;
  name : label;
END_ENTITY; -- person_and_organization_role

ENTITY personal_address
  SUBTYPE OF (address);
  people      : SET [1:?] OF person;
  description : text;
END_ENTITY; -- personal_address

ENTITY placement
  SUPERTYPE OF (ONEOF (axis1_placement, axis2_placement_2d,
    axis2_placement_3d))
  SUBTYPE OF (geometric_representation_item);
  location : cartesian_point;
END_ENTITY; -- placement

ENTITY plane
SUBTYPE OF (elementary_surface);
END_ENTITY; -- plane

ENTITY plane_angle_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.PLANE_ANGLE_UNIT' IN
       TYPEOF(SELF\measure_with_unit.unit_component);
END_ENTITY; -- plane_angle_measure_with_unit

ENTITY plane_angle_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 0) AND
       (SELF\named_unit.dimensions.mass_exponent = 0) AND
       (SELF\named_unit.dimensions.time_exponent = 0) AND
       (SELF\named_unit.dimensions.electric_current_exponent = 0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent = 0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- plane_angle_unit
```

```
ENTITY point
  SUPERTYPE OF (ONEOF (cartesian_point, point_on_curve, point_on_surface,
    point_replica, degenerate_pcurve))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- point

ENTITY point_on_curve
  SUBTYPE OF (point);
  basis_curve     : curve;
  point_parameter : parameter_value;
END_ENTITY; -- point_on_curve

ENTITY point_on_surface
  SUBTYPE OF (point);
  basis_surface     : surface;
  point_parameter_u : parameter_value;
  point_parameter_v : parameter_value;
END_ENTITY; -- point_on_surface

ENTITY point_replica
  SUBTYPE OF (point);
  parent_pt      : point;
  transformation : cartesian_transformation_operator;
WHERE
  WR1: transformation.dim = parent_pt.dim;
  WR2: acyclic_point_replica(SELF,parent_pt);
END_ENTITY; -- point_replica

ENTITY poly_loop
  SUBTYPE OF (loop, geometric_representation_item);
    polygon : LIST [3:?] OF UNIQUE cartesian_point;
END_ENTITY; -- poly_loop

ENTITY polyline
  SUBTYPE OF (bounded_curve);
  points : LIST [2:?] OF cartesian_point;
END_ENTITY; -- polyline

ENTITY product;
  id                  : identifier;
  name                : label;
  description         : text;
  frame_of_reference : SET [1:?] OF product_context;
UNIQUE
```

```
  UR1 : id;
END_ENTITY; -- product

ENTITY product_category;
  name        : label;
  description : OPTIONAL text;
END_ENTITY; -- product_category

ENTITY product_category_relationship;
  name         : label;
  description  : text;
  category     : product_category;
  sub_category : product_category;
WHERE
  WR1: acyclic_product_category_relationship(SELF,[SELF.sub_category]);
END_ENTITY; -- product_category_relationship

ENTITY product_concept;
  id             : identifier;
  name           : label;
  description    : text;
  market_context : product_concept_context;
UNIQUE
  UR1 : id;
END_ENTITY; -- product_concept

ENTITY product_concept_context
  SUBTYPE OF (application_context_element);
  market_segment_type : label;
END_ENTITY; -- product_concept_context

ENTITY product_context
  SUBTYPE OF (application_context_element);
  discipline_type : label;
END_ENTITY; -- product_context

ENTITY product_definition;
  id                  : identifier;
  description         : text;
  formation           : product_definition_formation;
  frame_of_reference : product_definition_context;
END_ENTITY; -- product_definition

ENTITY product_definition_context
```

```
  SUBTYPE OF (application_context_element);
  life_cycle_stage : label;
END_ENTITY; -- product_definition_context

ENTITY product_definition_effectivity
  SUBTYPE OF (effectivity);
  usage : product_definition_relationship;
UNIQUE
  UR1 : usage, id;
END_ENTITY; -- product_definition_effectivity

ENTITY product_definition_formation;
  id          : identifier;
  description : text;
  of_product  : product;
UNIQUE
  UR1 : id, of_product;
END_ENTITY; -- product_definition_formation

ENTITY product_definition_formation_with_specified_source
  SUBTYPE OF (product_definition_formation);
  make_or_buy : source;
END_ENTITY; -- product_definition_formation_with_specified_source

ENTITY product_definition_relationship;
  id                          : identifier;
  name                        : label;
  description                 : text;
  relating_product_definition : product_definition;
  related_product_definition  : product_definition;
END_ENTITY; -- product_definition_relationship

ENTITY product_definition_shape
  SUBTYPE OF (property_definition);
UNIQUE
  UR1 : definition;
WHERE
  WR1: 'PRODUCT_PROPERTY_DEFINITION.CHARACTERIZED_PRODUCT_DEFINITION'
       IN TYPEOF(SELF\property_definition.definition);
END_ENTITY; -- product_definition_shape

ENTITY product_definition_usage
  SUPERTYPE OF (assembly_component_usage)
  SUBTYPE OF (product_definition_relationship);
```

```
UNIQUE
  UR1 : id, relating_product_definition, related_product_definition;
WHERE
  WR1: acyclic_product_definition_relationship (SELF,
       [SELF\product_definition_relationship.related_product_definition],
       'CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION_USAGE.' +
       'RELATED_PRODUCT_DEFINITION');
END_ENTITY; -- product_definition_usage

ENTITY product_definition_with_associated_documents
  SUBTYPE OF (product_definition);
  documentation_ids : SET [1:?] OF document;
END_ENTITY; -- product_definition_with_associated_documents

ENTITY product_related_product_category
  SUBTYPE OF (product_category);
  products : SET [1:?] OF product;
END_ENTITY; -- product_related_product_category

ENTITY promissory_usage_occurrence
  SUBTYPE OF (assembly_component_usage);
END_ENTITY; -- promissory_usage_occurrence

ENTITY property_definition;
  name        : label;
  description : text;
  definition  : characterized_definition;
END_ENTITY; -- property_definition

ENTITY property_definition_representation;
  definition          : property_definition;
  used_representation : representation;
END_ENTITY; -- property_definition_representation

ENTITY quantified_assembly_component_usage
  SUBTYPE OF (assembly_component_usage);
  quantity : measure_with_unit;
END_ENTITY; -- quantified_assembly_component_usage

ENTITY quasi_uniform_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- quasi_uniform_curve

ENTITY quasi_uniform_surface
```

```
  SUBTYPE OF (b_spline_surface);
END_ENTITY; -- quasi_uniform_surface

ENTITY rational_b_spline_curve
  SUBTYPE OF (b_spline_curve);
  weights_data : LIST [2:?] OF REAL;
DERIVE
  weights : ARRAY [0:upper_index_on_control_points] OF REAL :=
            list_to_array(weights_data, 0, upper_index_on_control_points);
WHERE
  WR1: SIZEOF(weights_data) =
       SIZEOF (SELF\b_spline_curve.control_points_list);
  WR2: curve_weights_positive (SELF);
END_ENTITY; -- rational_b_spline_curve

ENTITY rational_b_spline_surface
SUBTYPE OF (b_spline_surface);
  weights_data : LIST [2:?] OF LIST [2:?] OF REAL;
DERIVE
  weights : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF REAL :=
            make_array_of_array(weights_data, 0, u_upper, 0, v_upper);
WHERE
  WR1: (SIZEOF(weights_data) =
       SIZEOF(SELF\b_spline_surface.control_points_list)) AND
       (SIZEOF(weights_data[1]) =
       SIZEOF(SELF\b_spline_surface.control_points_list[1]));
  WR2: surface_weights_positive (SELF);
END_ENTITY; -- rational_b_spline_surface

ENTITY rectangular_composite_surface
  SUBTYPE OF (bounded_surface);
  segments : LIST [1:?] OF LIST [1:?] OF surface_patch;
DERIVE
  n_u : INTEGER := SIZEOF(segments);
  n_v : INTEGER := SIZEOF(segments[1]);
WHERE
  WR1: [] = QUERY ( s <* segments | n_v <> SIZEOF(s));
  WR2: constraints_rectangular_composite_surface(SELF);
END_ENTITY; -- rectangular_composite_surface

ENTITY rectangular_trimmed_surface
  SUBTYPE OF (bounded_surface);
  basis_surface : surface;
  u1            : parameter_value;
```

```
  u2              : parameter_value;
  v1              : parameter_value;
  v2              : parameter_value;
  usense          : BOOLEAN;
  vsense          : BOOLEAN;
WHERE
  WR1: u1 <> u2;
  WR2: v1 <> v2;
  WR3: ((('CONFIG_CONTROL_DESIGN.ELEMENTARY_SURFACE' IN
       TYPEOF (basis_surface)) AND
       (NOT ('CONFIG_CONTROL_DESIGN.PLANE' IN TYPEOF (basis_surface)))) OR
       ('CONFIG_CONTROL_DESIGN.SURFACE_OF_REVOLUTION' IN
       TYPEOF (basis_surface))) OR (usense = (u2 > u1));
  WR4: (('CONFIG_CONTROL_DESIGN.SPHERICAL_SURFACE' IN
       TYPEOF (basis_surface)) OR
       ('CONFIG_CONTROL_DESIGN.TOROIDAL_SURFACE'
       IN TYPEOF (basis_surface))) OR (vsense = (v2 > v1));
END_ENTITY; -- rectangular_trimmed_surface

ENTITY reparametrised_composite_curve_segment
  SUBTYPE OF (composite_curve_segment);
  param_length : parameter_value;
WHERE
  WR1: param_length > 0;
END_ENTITY; -- reparametrised_composite_curve_segment

ENTITY representation;
  name             : label;
  items            : SET [1:?] OF representation_item;
  context_of_items : representation_context;
END_ENTITY; -- representation

ENTITY representation_context;
  context_identifier : identifier;
  context_type       : text;
INVERSE
  representations_in_context : SET [1:?] OF representation FOR
                               context_of_items;
END_ENTITY; -- representation_context

ENTITY representation_item;
  name : label;
WHERE
  WR1: SIZEOF (using_representations(SELF)) > 0;
```

```
END_ENTITY; -- representation_item

ENTITY representation_map;
  mapping_origin        : representation_item;
  mapped_representation : representation;
INVERSE
  map_usage : SET [1:?] OF mapped_item FOR mapping_source;
WHERE
  WR1: item_in_context(SELF.mapping_origin,
       SELF.mapped_representation.context_of_items);
END_ENTITY; -- representation_map

ENTITY representation_relationship;
  name        : label;
  description : text;
  rep_1       : representation;
  rep_2       : representation;
END_ENTITY; -- representation_relationship

ENTITY representation_relationship_with_transformation
  SUBTYPE OF (representation_relationship);
  transformation_operator : transformation;
WHERE
  WR1: SELF\representation_relationship.rep_1.context_of_items :<>:
       SELF\representation_relationship.rep_2.context_of_items;
END_ENTITY; -- representation_relationship_with_transformation

ENTITY seam_curve
  SUBTYPE OF (surface_curve);
WHERE
  WR1: SIZEOF(SELF\surface_curve.associated_geometry) = 2;
  WR2: associated_surface(SELF\surface_curve.associated_geometry[1]) =
       associated_surface(SELF\surface_curve.associated_geometry[2]);
  WR3: 'CONFIG_CONTROL_DESIGN.PCURVE' IN
       TYPEOF (SELF\surface_curve.associated_geometry[1]);
  WR4: 'CONFIG_CONTROL_DESIGN.PCURVE' IN
       TYPEOF (SELF\surface_curve.associated_geometry[2]);
END_ENTITY; -- seam_curve

ENTITY security_classification;
  name           : label;
  purpose        : text;
  security_level : security_classification_level;
END_ENTITY; -- security_classification
```

```
ENTITY security_classification_assignment
  ABSTRACT SUPERTYPE;
  assigned_security_classification : security_classification;
END_ENTITY; -- security_classification_assignment

ENTITY security_classification_level;
  name : label;
END_ENTITY; -- security_classification_level

ENTITY serial_numbered_effectivity
  SUBTYPE OF (effectivity);
  effectivity_start_id : identifier;
  effectivity_end_id   : OPTIONAL identifier;
END_ENTITY; -- serial_numbered_effectivity

ENTITY shape_aspect;
  name                 : label;
  description          : text;
  of_shape             : product_definition_shape;
  product_definitional : LOGICAL;
END_ENTITY; -- shape_aspect

ENTITY shape_aspect_relationship;
  name                   : label;
  description            : text;
  relating_shape_aspect  : shape_aspect;
  related_shape_aspect   : shape_aspect;
END_ENTITY; -- shape_aspect_relationship

ENTITY shape_definition_representation
  SUBTYPE OF (property_definition_representation);
WHERE
  WR1: ('CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION' IN
       TYPEOF(SELF.definition.definition)) OR
       ('CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION_SHAPE' IN
       TYPEOF(SELF.definition));
  WR2: 'CONFIG_CONTROL_DESIGN.SHAPE_REPRESENTATION' IN
       TYPEOF(SELF.used_representation);
END_ENTITY; -- shape_definition_representation

ENTITY shape_representation
  SUBTYPE OF (representation);
END_ENTITY; -- shape_representation
```

```
ENTITY shape_representation_relationship
  SUBTYPE OF (representation_relationship);
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.SHAPE_REPRESENTATION' IN
       (TYPEOF (SELF\representation_relationship.rep_1) +
       TYPEOF(SELF\representation_relationship.rep_2));
END_ENTITY; -- shape_representation_relationship

ENTITY shell_based_surface_model
  SUBTYPE OF (geometric_representation_item);
  sbsm_boundary : SET [1:?] OF shell;
WHERE
  WR1: constraints_geometry_shell_based_surface_model(SELF);
END_ENTITY; -- shell_based_surface_model

ENTITY shell_based_wireframe_model
  SUBTYPE OF (geometric_representation_item);
  sbwm_boundary : SET [1:?] OF shell;
WHERE
  WR1: constraints_geometry_shell_based_wireframe_model(SELF);
END_ENTITY; -- shell_based_wireframe_model

ENTITY shell_based_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* SELF\representation.items |
       NOT (SIZEOF(['CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL',
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM',
       'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF (it)) = 1
       ))) = 0;
  WR2: SIZEOF (QUERY (it <* SELF\representation.items |
       SIZEOF(['CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL',
       'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'] * TYPEOF (it)) = 1)) >= 1;
  WR3: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
       'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
       sbwm\shell_based_wireframe_model.sbwm_boundary |
       'CONFIG_CONTROL_DESIGN.WIRE_SHELL' IN TYPEOF (sb)) |
       NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
       ws\wire_shell.wire_shell_extent |
       'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (wsb)) |
       NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
       NOT ('CONFIG_CONTROL_DESIGN.EDGE_CURVE' IN
       TYPEOF (el.edge_element)))) = 0))) = 0))) = 0))) = 0;
```

```
WR4: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
     'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
     NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
     sbwm\shell_based_wireframe_model.sbwm_boundary |
     'CONFIG_CONTROL_DESIGN.WIRE_SHELL' IN TYPEOF (sb)) |
     NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
     ws\wire_shell.wire_shell_extent |
     'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (wsb)) |
     NOT (SIZEOF (QUERY (pline_el <* QUERY (el <* eloop\path.edge_list |
     'CONFIG_CONTROL_DESIGN.POLYLINE' IN TYPEOF
     (el.edge_element\edge_curve.edge_geometry)) |
     NOT (SIZEOF (pline_el.edge_element\edge_curve.
     edge_geometry\polyline.points) > 2)))
     = 0))) = 0))) = 0))) = 0;
WR5: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
     'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
     NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
     sbwm\shell_based_wireframe_model.sbwm_boundary |
     'CONFIG_CONTROL_DESIGN.WIRE_SHELL' IN TYPEOF (sb)) |
     NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
     ws\wire_shell.wire_shell_extent |
     'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (wsb)) |
     NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
     NOT (valid_wireframe_edge_curve
     (el.edge_element\edge_curve.edge_geometry,
     'CONFIG_CONTROL_DESIGN')))) = 0))) = 0))) = 0))) = 0;
WR6: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
     'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
     NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
     sbwm\shell_based_wireframe_model.sbwm_boundary |
     'CONFIG_CONTROL_DESIGN.WIRE_SHELL' IN TYPEOF (sb)) |
     NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
     ws\wire_shell.wire_shell_extent |
     'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (wsb)) |
     NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
     NOT(('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN
     TYPEOF (el.edge_element.edge_start))
     AND
     ('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN
     TYPEOF (el.edge_element.edge_end))))) = 0))) = 0))) = 0))) = 0;
WR7: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
     'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
     NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
     sbwm\shell_based_wireframe_model.sbwm_boundary |
```

```
      'CONFIG_CONTROL_DESIGN.WIRE_SHELL' IN TYPEOF (sb)) |
      NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
      ws\wire_shell.wire_shell_extent |
      'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (wsb)) |
      NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
      NOT((valid_wireframe_vertex_point
      (el.edge_element.edge_start\vertex_point.vertex_geometry,
      'CONFIC_CONTROL_DESIGN'))
      AND
      (valid_wireframe_vertex_point
      (el.edge_element.edge_end\vertex_point.vertex_geometry,
      'CONFIG_CONTROL_DESIGN'))))) = 0)))
      = 0))) = 0))) = 0;
 WR8: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
      sbwm\shell_based_wireframe_model.sbwm_boundary |
      'CONFIG_CONTROL_DESIGN.WIRE_SHELL' IN TYPEOF (sb)) |
      NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
      ws\wire_shell.wire_shell_extent |
      'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF (wsb)) |
      NOT (SIZEOF (QUERY (con_edges <* QUERY (el <* eloop\path.edge_list |
      'CONFIG_CONTROL_DESIGN.CONIC' IN
      TYPEOF (el.edge_element\edge_curve.edge_geometry)) |
      NOT ('CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D' IN
      TYPEOF (con_edges.edge_element\edge_curve.edge_geometry\
      conic.position)))) = 0))) = 0))) = 0))) = 0;
 WR9: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
      sbwm\shell_based_wireframe_model.sbwm_boundary |
      'CONFIG_CONTROL_DESIGN.WIRE_SHELL' IN TYPEOF (sb)) |
      NOT (SIZEOF (QUERY (vloop <* QUERY (wsb <*
      ws\wire_shell.wire_shell_extent |
      'CONFIG_CONTROL_DESIGN.VERTEX_LOOP' IN TYPEOF (wsb)) |
      NOT ('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN
      TYPEOF (vloop.loop_vertex)))) = 0))) = 0))) = 0;
WR10: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
      sbwm\shell_based_wireframe_model.sbwm_boundary |
      'CONFIG_CONTROL_DESIGN.WIRE_SHELL' IN TYPEOF (sb)) |
      NOT (SIZEOF (QUERY (vloop <* QUERY (wsb <*
      ws\wire_shell.wire_shell_extent |
```

```
      'CONFIG_CONTROL_DESIGN.VERTEX_LOOP' IN TYPEOF (wsb)) |
      NOT(valid_wireframe_vertex_point
      (vloop\vertex_loop.loop_vertex\vertex_point.vertex_geometry,
      'CONFIG_CONTROL_DESIGN')))) = 0))) = 0))) = 0;
WR11: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (vs <* QUERY (sb <*
      sbwm\shell_based_wireframe_model.sbwm_boundary |
      'CONFIG_CONTROL_DESIGN.VERTEX_SHELL' IN TYPEOF (sb)) |
      NOT ('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN
      TYPEOF (vs\vertex_shell.vertex_shell_extent.loop_vertex))))
      = 0))) = 0;
WR12: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (vs <* QUERY (sb <*
      sbwm\shell_based_wireframe_model.sbwm_boundary |
      'CONFIG_CONTROL_DESIGN.VERTEX_SHELL' IN TYPEOF (sb)) |
      NOT (valid_wireframe_vertex_point
      (vs\vertex_shell.vertex_shell_extent.loop_vertex\
      vertex_point.vertex_geometry,  'CONFIG_CONTROL_DESIGN'))))
      = 0))) = 0;
WR13: SIZEOF (QUERY (mi <* QUERY (item <* SELF\representation.items |
      'CONFIG_CONTROL_DESIGN.MAPPED_ITEM' IN TYPEOF (item)) |
      NOT ('CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_REPRESENTATION'
      IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
      = 0;
END_ENTITY; -- shell_based_wireframe_shape_representation

ENTITY si_unit
SUBTYPE OF (named_unit);
  prefix : OPTIONAL si_prefix;
  name   : si_unit_name;
DERIVE
  SELF\named_unit.dimensions : dimensional_exponents :=
                               dimensions_for_si_unit(SELF.name);
END_ENTITY; -- si_unit

ENTITY specified_higher_usage_occurrence
  SUBTYPE OF (assembly_component_usage);
  upper_usage : assembly_component_usage;
  next_usage  : next_assembly_usage_occurrence;
UNIQUE
  UR1 : upper_usage, next_usage;
WHERE
```

```
  WR1: SELF :<>: upper_usage;
  WR2: SELF\product_definition_relationship.relating_product_definition :=:
       upper_usage.relating_product_definition;
  WR3: SELF\product_definition_relationship.related_product_definition :=:
       next_usage.related_product_definition;
  WR4: upper_usage.related_product_definition :=:
       next_usage.relating_product_definition;
  WR5: NOT ('CONFIG_CONTROL_DESIGN.PROMISSORY_USAGE_OCCURRENCE' IN
       TYPEOF(upper_usage));
END_ENTITY; -- specified_higher_usage_occurrence

ENTITY solid_angle_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.SOLID_ANGLE_UNIT' IN
       TYPEOF(SELF\measure_with_unit.unit_component);
END_ENTITY; -- solid_angle_measure_with_unit

ENTITY solid_angle_unit
SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 0) AND
       (SELF\named_unit.dimensions.mass_exponent = 0) AND
       (SELF\named_unit.dimensions.time_exponent = 0) AND
       (SELF\named_unit.dimensions.electric_current_exponent = 0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent = 0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- solid_angle_unit

ENTITY solid_model
  SUPERTYPE OF (manifold_solid_brep)
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- solid_model

ENTITY spherical_surface
  SUBTYPE OF (elementary_surface);
  radius : positive_length_measure;
END_ENTITY; -- spherical_surface

ENTITY start_request
  SUBTYPE OF (action_request_assignment);
  items : SET [1:?] OF start_request_item;
END_ENTITY; -- start_request
```

```
ENTITY start_work
  SUBTYPE OF (action_assignment);
  items : SET [1:?] OF work_item;
END_ENTITY; -- start_work

ENTITY supplied_part_relationship
  SUBTYPE OF (product_definition_relationship);
END_ENTITY; -- supplied_part_relationship

ENTITY surface
  SUPERTYPE OF (ONEOF (elementary_surface, swept_surface, bounded_surface,
    offset_surface, surface_replica))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- surface

ENTITY surface_curve
  SUPERTYPE OF (ONEOF (intersection_curve, seam_curve))
  SUBTYPE OF (curve);
  curve_3d              : curve;
  associated_geometry   : LIST [1:2] OF pcurve_or_surface;
  master_representation : preferred_surface_curve_representation;
DERIVE
  basis_surface : SET [1:2] OF surface := get_basis_surface(SELF);
WHERE
  WR1: curve_3d.dim = 3;
  WR2: ('CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(associated_geometry[1]))
       OR (master_representation <> pcurve_s1);
  WR3: ('CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(associated_geometry[2]))
       OR (master_representation <> pcurve_s2);
  WR4: NOT ('CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(curve_3d));
END_ENTITY; -- surface_curve

ENTITY surface_of_linear_extrusion
  SUBTYPE OF (swept_surface);
  extrusion_axis : vector;
END_ENTITY; -- surface_of_linear_extrusion

ENTITY surface_of_revolution
  SUBTYPE OF (swept_surface);
  axis_position : axis1_placement;
DERIVE
  axis_line : line := line(axis_position.location,
                           vector (axis_position.z, 1.0));
END_ENTITY; -- surface_of_revolution
```

```
ENTITY surface_patch;
  parent_surface : bounded_surface;
  u_transition   : transition_code;
  v_transition   : transition_code;
  u_sense        : BOOLEAN;
  v_sense        : BOOLEAN;
INVERSE
  using_surfaces : BAG [1:?] OF rectangular_composite_surface FOR
                   segments;
WHERE
  WR1: NOT ('CONFIG_CONTROL_DESIGN.CURVE_BOUNDED_SURFACE' IN
       TYPEOF (parent_surface));
END_ENTITY; -- surface_patch

ENTITY surface_replica
  SUBTYPE OF (surface);
  parent_surface : surface;
  transformation : cartesian_transformation_operator_3d;
WHERE
  WR1: acyclic_surface_replica (SELF,parent_surface);
END_ENTITY; -- surface_replica

ENTITY swept_surface
  SUPERTYPE OF (ONEOF (surface_of_linear_extrusion, surface_of_revolution))
  SUBTYPE OF (surface);
  swept_curve : curve;
END_ENTITY; -- swept_surface

ENTITY topological_representation_item
  SUPERTYPE OF (ONEOF (vertex, edge, face_bound, face, connected_edge_set,
    connected_face_set, vertex_shell, wire_shell, (loop ANDOR path)))
  SUBTYPE OF (representation_item);
END_ENTITY; -- topological_representation_item

ENTITY toroidal_surface
  SUBTYPE OF (elementary_surface);
  major_radius : positive_length_measure;
  minor_radius : positive_length_measure;
END_ENTITY; -- toroidal_surface

ENTITY trimmed_curve
  SUBTYPE OF (bounded_curve);
  basis_curve          : curve;
  trim_1               : SET [1:2] OF trimming_select;
```

```
  trim_2                : SET [1:2] OF trimming_select;
  sense_agreement       : BOOLEAN;
  master_representation : trimming_preference;
WHERE
  WR1: (HIINDEX(trim_1) = 1) XOR (TYPEOF(trim_1[1]) <> TYPEOF(trim_1[2]));
  WR2: (HIINDEX(trim_2) = 1) XOR (TYPEOF(trim_2[1]) <> TYPEOF(trim_2[2]));
END_ENTITY; -- trimmed_curve

ENTITY uncertainty_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  name        : label;
  description : text;
WHERE
  WR1: (NOT ('NUMBER' IN TYPEOF
       (SELF\measure_with_unit.value_component))) OR
       (SELF\measure_with_unit.value_component >= 0);
END_ENTITY; -- uncertainty_measure_with_unit

ENTITY uniform_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- uniform_curve

ENTITY uniform_surface
  SUBTYPE OF (b_spline_surface);
END_ENTITY; -- uniform_surface

ENTITY vector
  SUBTYPE OF (geometric_representation_item);
  orientation : direction;
  magnitude   : length_measure;
WHERE
  WR1: magnitude >= 0;
END_ENTITY; -- vector

ENTITY versioned_action_request;
  id          : identifier;
  version     : label;
  purpose     : text;
  description : text;
END_ENTITY; -- versioned_action_request

ENTITY vertex
  SUBTYPE OF (topological_representation_item);
END_ENTITY; -- vertex
```

```
ENTITY vertex_loop
  SUBTYPE OF (loop);
  loop_vertex : vertex;
END_ENTITY; -- vertex_loop

ENTITY vertex_point
SUBTYPE OF (vertex, geometric_representation_item);
  vertex_geometry : point;
END_ENTITY; -- vertex_point

ENTITY vertex_shell
  SUBTYPE OF (topological_representation_item);
    vertex_shell_extent : vertex_loop;
END_ENTITY; -- vertex_shell

ENTITY volume_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'CONFIG_CONTROL_DESIGN.VOLUME_UNIT' IN TYPEOF(SELF\
       measure_with_unit.unit_component);
END_ENTITY; -- volume_measure_with_unit

ENTITY volume_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 3) AND
       (SELF\named_unit.dimensions.mass_exponent = 0) AND
       (SELF\named_unit.dimensions.time_exponent = 0) AND
       (SELF\named_unit.dimensions.electric_current_exponent = 0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent = 0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- volume_unit

ENTITY week_of_year_and_day_date
  SUBTYPE OF (date);
  week_component : week_in_year_number;
  day_component  : OPTIONAL day_in_week_number;
END_ENTITY; -- week_of_year_and_day_date

ENTITY wire_shell
  SUBTYPE OF (topological_representation_item);
    wire_shell_extent : SET [1:?] OF loop;
  WHERE
```

```
    wr1: NOT mixed_loop_type_set(wire_shell_extent);
END_ENTITY; -- wire_shell


RULE application_context_requires_ap_definition FOR
  (application_context, application_protocol_definition);
WHERE
  WR1: SIZEOF (QUERY (ac <* application_context |
       NOT (SIZEOF (QUERY (apd <* application_protocol_definition |
       (ac :=: apd.application)
       AND
       (apd.application_interpreted_model_schema_name =
       'config_control_design'))) = 1 ))) = 0;
END_RULE; --application_context_requires_ap_definition

RULE approval_requires_approval_date_time FOR (approval,
         approval_date_time);
WHERE
  WR1: SIZEOF (QUERY ( app <* approval |
       NOT (SIZEOF (QUERY (adt <* approval_date_time |
       app :=: adt.dated_approval)) = 1))) = 0;
END_RULE; -- approval_requires_approval_date_time

RULE approval_requires_approval_person_organization FOR (approval,
         approval_person_organization);
WHERE
  WR1: SIZEOF (QUERY (app <* approval |
       NOT (SIZEOF (QUERY (apo <* approval_person_organization |
       app :=: apo.authorized_approval)) >= 1))) = 0;
END_RULE; -- approval_requires_approval_person_organization

RULE approvals_are_assigned FOR
  (approval, approval_assignment);
WHERE
  WR1: SIZEOF (QUERY (app <* approval |
       NOT (SIZEOF (QUERY (aa <* approval_assignment |
       app :=: aa.assigned_approval )) >= 1 ))) = 0;
END_RULE; -- approvals_are_assigned

RULE as_required_quantity FOR (measure_with_unit);
WHERE
  WR1: SIZEOF (QUERY (m <* measure_with_unit |
       ('CONFIG_CONTROL_DESIGN.DESCRIPTIVE_MEASURE' IN
       TYPEOF (m.value_component)) AND
```

```
      (NOT (m.value_component = 'as_required')))) = 0;
END_RULE; -- as_required_quantity

RULE certification_requires_approval FOR (certification,
  cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (cert <* certification |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       cert IN ccda.items )) = 1 ))) = 0;
END_RULE; -- certification_requires_approval

RULE certification_requires_date_time FOR
  (certification, cc_design_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (cert <* certification |
       NOT (SIZEOF (QUERY (ccdta <* cc_design_date_and_time_assignment |
       cert IN ccdta.items )) = 1 ))) = 0;
END_RULE; -- certification_requires_date_time

RULE change_request_requires_approval FOR
  (change_request, cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (cr <* change_request |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       cr IN ccda.items )) = 1 ))) = 0;
END_RULE; -- change_request_requires_approval

RULE change_request_requires_date_time FOR
  (change_request, cc_design_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (cr <* change_request |
       NOT (SIZEOF (QUERY (ccdta <* cc_design_date_and_time_assignment |
       cr IN ccdta.items )) = 1 ))) = 0;
END_RULE; -- change_request_requires_date_time

RULE change_request_requires_person_organization FOR
  (change_request,
  cc_design_person_and_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (cr <* change_request |
       NOT (SIZEOF (QUERY (ccpoa <*
       cc_design_person_and_organization_assignment |
       cr IN ccpoa.items )) >= 1 ))) = 0;
END_RULE; -- change_request_requires_person_organization
```

```
RULE change_requires_approval FOR
  (change, cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (chg <* change |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       chg IN ccda.items )) = 1 ))) = 0;
END_RULE; -- change_requires_approval

RULE change_requires_date_time FOR
  (change, cc_design_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (chg <* change |
       NOT (SIZEOF (QUERY (ccdta <* cc_design_date_and_time_assignment |
       (chg IN ccdta.items)
       AND (ccdta.role.name = 'start_date'))) = 1 ))) = 0;
END_RULE; -- change_requires_date_time

RULE compatible_dimension FOR (cartesian_point, direction,
         representation_context, geometric_representation_context);
WHERE
WR1: SIZEOF (QUERY (x <* cartesian_point |
     SIZEOF (QUERY (y <* geometric_representation_context |
     (item_in_context(x,y)) AND (HIINDEX (x.coordinates) <>
     y.coordinate_space_dimension))) >  0)) = 0;
WR2: SIZEOF (QUERY (x <* direction |
     SIZEOF (QUERY (y <* geometric_representation_context |
     (item_in_context(x,y)) AND (HIINDEX (x.direction_ratios) <>
     y.coordinate_space_dimension))) > 0)) = 0;
END_RULE; -- compatible_dimension

RULE configuration_item_requires_approval FOR
  (configuration_item, cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (ci <* configuration_item |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       ci IN ccda.items )) = 1 ))) = 0;
END_RULE; -- configuration_item_requires_approval

RULE configuration_item_requires_person_organization FOR
  (configuration_item,
  cc_design_person_and_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (ci <* configuration_item |
       NOT (SIZEOF (QUERY (ccdpoa <*
```

```
      cc_design_person_and_organization_assignment |
      ci IN ccdpoa.items )) = 1 ))) = 0;
END_RULE; -- configuration_item_requires_person_organization

RULE contract_requires_approval FOR (contract, cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (c <* contract |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       c IN ccda.items )) = 1 ))) = 0;
END_RULE; -- contract_requires_approval

RULE contract_requires_person_organization FOR
  (contract, cc_design_person_and_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (c <* contract |
       NOT (SIZEOF (QUERY (ccdpoa <*
       cc_design_person_and_organization_assignment |
       c IN ccdpoa.items )) = 1 ))) = 0;
END_RULE; -- contract_requires_person_organization

RULE coordinated_assembly_and_shape FOR (next_assembly_usage_occurrence);
WHERE
  WR1: SIZEOF (QUERY (nauo <* next_assembly_usage_occurrence |
       NOT assembly_shape_is_defined(nauo))) = 0;
END_RULE; -- coordinated_assembly_and_shape

RULE dependent_instantiable_action_directive FOR (action_directive);
WHERE
  WR1: SIZEOF (QUERY (ad <* action_directive |
       NOT (SIZEOF (USEDIN (ad, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_action_directive

RULE dependent_instantiable_approval_status FOR (approval_status);
WHERE
WR1: SIZEOF (QUERY (ast <* approval_status |
     NOT (SIZEOF (USEDIN (ast, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_approval_status

RULE dependent_instantiable_certification_type FOR (certification_type);
WHERE
  WR1: SIZEOF (QUERY (ct <* certification_type |
       NOT (SIZEOF (USEDIN (ct, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_certification_type
```

```
RULE dependent_instantiable_contract_type FOR (contract_type);
WHERE
  WR1: SIZEOF(QUERY ( ct <* contract_type |
       NOT (SIZEOF (USEDIN (ct, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_contract_type

RULE dependent_instantiable_date FOR (date);
WHERE
  WR1: SIZEOF (QUERY (dt <* date |
       NOT (SIZEOF(USEDIN (dt, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_date

RULE dependent_instantiable_date_time_role FOR (date_time_role);
WHERE
  WR1: SIZEOF (QUERY (dtr <* date_time_role |
       NOT (SIZEOF (USEDIN (dtr, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_date_time_role

RULE dependent_instantiable_document_type FOR (document_type);
WHERE
  WR1: SIZEOF (QUERY (dt <* document_type |
       NOT (SIZEOF (USEDIN (dt, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_document_type

RULE dependent_instantiable_named_unit FOR (named_unit);
WHERE
  WR1: SIZEOF (QUERY (nu <* named_unit |
       NOT (SIZEOF (USEDIN (nu, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_named_unit

RULE dependent_instantiable_person_and_organization_role FOR (
        person_and_organization_role);
WHERE
  WR1: SIZEOF (QUERY (poar <* person_and_organization_role |
       NOT (SIZEOF (USEDIN (poar, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_person_and_organization_role

RULE dependent_instantiable_representation_item FOR (representation_item);
WHERE
  WR1: SIZEOF (QUERY (ri <* representation_item |
       NOT (SIZEOF (USEDIN (ri, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_representation_item

RULE dependent_instantiable_security_classification_level FOR (
```

```
        security_classification_level);
WHERE
  WR1: SIZEOF (QUERY (scl <* security_classification_level |
       NOT (SIZEOF (USEDIN (scl, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_security_classification_level

RULE dependent_instantiable_shape_representation FOR (
        shape_representation);
WHERE
  WR1: SIZEOF (QUERY (sr <* shape_representation |
       NOT (SIZEOF(USEDIN(sr, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_shape_representation

RULE design_context_for_property FOR (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pd <* product_definition |
       (SIZEOF (USEDIN (pd, 'CONFIG_CONTROL_DESIGN.' +
       'PROPERTY_DEFINITION.DEFINITION') +
       QUERY (pdr <* USEDIN (pd, 'CONFIG_CONTROL_DESIGN.' +
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
       SIZEOF (USEDIN (pdr, 'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION.' +
       'DEFINITION')) >= 1)) >= 1) AND
       (NOT ('CONFIG_CONTROL_DESIGN.DESIGN_CONTEXT' IN
       TYPEOF (pd.frame_of_reference))))) = 0;
END_RULE; -- design_context_for_property

RULE document_to_product_definition FOR
  (cc_design_specification_reference, product_definition);
WHERE
  WR1: SIZEOF (QUERY (sp <* cc_design_specification_reference |
       NOT ((((('CONFIG_CONTROL_DESIGN.DOCUMENT_RELATIONSHIP.' +
       'RELATING DOCUMENT' IN
       ROLESOF (sp\document_reference.assigned_document)) AND
       (SIZEOF (QUERY (it <* sp.items |
       NOT('CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION' IN
       TYPEOF (it)))) = 0)))
       OR
       (NOT ('CONFIG_CONTROL_DESIGN.DOCUMENT_RELATIONSHIP.' +
       'RELATING_DOCUMENT' IN
       ROLESOF (sp\document_reference.assigned_document)))))) = 0;
END_RULE; -- document_to_product_definition

RULE effectivity_requires_approval FOR
  (effectivity, cc_design_approval);
```

```
WHERE
  WR1: SIZEOF (QUERY (eff <* effectivity |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       eff IN ccda.items )) = 1 ))) = 0;
END_RULE; -- effectivity_requires_approval

RULE geometric_representation_item_3d FOR
  (geometric_representation_item);
WHERE
  WR1: SIZEOF (QUERY (gri <* geometric_representation_item |
       NOT ((dimension_of (gri) = 3) OR
       (SIZEOF (bag_to_set (USEDIN (gri, '')) - bag_to_set (USEDIN (gri,
       'CONFIG_CONTROL_DESIGN.DEFINITIONAL_REPRESENTATION.ITEMS'))) = 0)
       ))) = 0;
END_RULE; -- geometric_representation_item_3d

RULE dependent_instantiable_parametric_representation_context FOR
  (parametric_representation_context);
WHERE
  WR1: SIZEOF (QUERY (prc <* parametric_representation_context |
       NOT (SIZEOF (USEDIN (prc, '')) >= 1))) = 0;
END_RULE;

RULE global_unit_assignment FOR (global_unit_assigned_context);
WHERE
  WR1: SIZEOF (QUERY (guac <* global_unit_assigned_context |
       NOT (SIZEOF (guac.units) = 3))) = 0;
  WR2: SIZEOF (QUERY (guac <* global_unit_assigned_context |
       NOT ((SIZEOF (QUERY (u <* guac.units |
       'CONFIG_CONTROL_DESIGN.LENGTH_UNIT' IN TYPEOF (u))) = 1) AND
       (SIZEOF (QUERY (u <* guac.units |
       'CONFIG_CONTROL_DESIGN.PLANE_ANGLE_UNIT' IN TYPEOF (u))) = 1) AND
       (SIZEOF (QUERY (u <* guac.units |
       'CONFIG_CONTROL_DESIGN.SOLID_ANGLE_UNIT' IN TYPEOF (u))) = 1
       )))) = 0;
END_RULE; -- global_unit_assignment

RULE no_shape_for_make_from FOR
  (design_make_from_relationship);
WHERE
  WR1: SIZEOF (QUERY (dmfr <* design_make_from_relationship |
       NOT (SIZEOF (QUERY (pd <* USEDIN (dmfr, 'CONFIG_CONTROL_DESIGN.' +
       'PROPERTY_DEFINITION.DEFINITION') |
       'CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION_SHAPE' IN TYPEOF (pd))) =
```

```
      0))) = 0;
END_RULE; -- no_shape_for_make_from

RULE no_shape_for_supplied_part FOR
  (supplied_part_relationship);
WHERE
  WR1: SIZEOF (QUERY (spr <* supplied_part_relationship |
       NOT (SIZEOF (QUERY (pd <* USEDIN (spr, 'CONFIG_CONTROL_DESIGN.' +
       'PROPERTY_DEFINITION.DEFINITION') |
       'CONFIG_CONTROL_DESIGN.PRODUCT_DEFINITION_SHAPE' IN TYPEOF (pd))) =
       0))) = 0;
END_RULE; -- no_shape_for_supplied_part

RULE acu_requires_security_classification FOR
  (assembly_component_usage,
   cc_design_security_classification);
WHERE
  WR1: SIZEOF (QUERY (acu <* assembly_component_usage |
       NOT (SIZEOF (QUERY (ccdsc <* cc_design_security_classification |
       acu IN ccdsc.items )) = 1 ))) = 0;
END_RULE; -- acu_requires_security_classification

RULE product_concept_requires_configuration_item FOR
  (product_concept, configuration_item);
WHERE
  WR1: SIZEOF (QUERY (pc <* product_concept |
       NOT (SIZEOF (QUERY (ci <* configuration_item |
       pc :=: ci.item_concept)) >=1 ))) = 0;
END_RULE; -- product_concept_requires_configuration_item

RULE product_definition_requires_approval FOR
  (product_definition, cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (pd <* product_definition |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       pd IN ccda.items )) = 1 ))) = 0;
END_RULE; -- product_definition_requires_approval

RULE product_definition_requires_date_time FOR
  (product_definition, cc_design_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (pd <* product_definition |
       NOT (SIZEOF (QUERY (ccdta <* cc_design_date_and_time_assignment |
       pd IN ccdta.items )) = 1 ))) = 0;
```

```
END_RULE; -- product_definition_requires_date_time

RULE product_definition_requires_person_organization FOR
  (product_definition,
  cc_design_person_and_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (pd <* product_definition |
       NOT (SIZEOF (QUERY (ccdpoa <*
       cc_design_person_and_organization_assignment |
       pd IN ccdpoa.items )) = 1 ))) = 0;
END_RULE; -- product_definition_requires_person_organization

RULE product_requires_person_organization FOR
  (product, cc_design_person_and_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (prod <* product |
       NOT (SIZEOF (QUERY (ccdpoa <*
       cc_design_person_and_organization_assignment |
       prod IN ccdpoa.items )) = 1 ))) = 0;
END_RULE; -- product_requires_person_organization

RULE product_requires_product_category FOR
  (product, product_related_product_category);
WHERE
  WR1: SIZEOF (QUERY (prod <* product |
       NOT (SIZEOF (QUERY (prpc <* product_related_product_category |
       (prod IN prpc.products) AND
       (prpc.name IN ['assembly', 'inseparable_assembly', 'detail',
       'customer_furnished_equipment'])))) = 1))) = 0;
END_RULE; -- product_requires_product_category

RULE product_requires_version FOR (product, product_definition_formation);
WHERE
  WR1: SIZEOF (QUERY (prod <* product |
       NOT (SIZEOF (QUERY (pdf <* product_definition_formation |
       prod :=: pdf.of_product )) >= 1 ))) = 0;
END_RULE; -- product_requires_version

RULE product_version_requires_approval FOR  (product_definition_formation,
  cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (pdf <* product_definition_formation |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       pdf IN ccda.items )) = 1 ))) = 0;
```

```
END_RULE; -- product_version_requires_approval

RULE product_version_requires_person_organization FOR
  (product_definition_formation,
  cc_design_person_and_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (pdf <* product_definition_formation |
       NOT (SIZEOF (QUERY (ccdpoa <*
       cc_design_person_and_organization_assignment |
       (pdf IN ccdpoa.items) AND (ccdpoa.role.name = 'creator'))) = 1 ))) = 0;
  WR2: SIZEOF (QUERY (pdf <* product_definition_formation |
       NOT (SIZEOF (QUERY (ccdpoa <*
       cc_design_person_and_organization_assignment |
       (pdf IN ccdpoa.items) AND
       (ccdpoa.role.name IN ['design_supplier', 'part_supplier'])))) >= 1)))
       = 0;
END_RULE; -- product_version_requires_person_organization

RULE product_version_requires_security_classification FOR
  (product_definition_formation, cc_design_security_classification);
WHERE
  WR1: SIZEOF (QUERY (pdf <* product_definition_formation |
       NOT (SIZEOF (QUERY (ccdsc <* cc_design_security_classification |
       pdf IN ccdsc.items )) = 1 ))) = 0;
END_RULE; -- product_version_requires_security_classification

RULE restrict_action_request_status FOR (action_request_status);
WHERE
  WR1: SIZEOF (QUERY (ars <* action_request_status |
       NOT (ars.status IN
       ['proposed', 'in_work', 'issued', 'hold']))) = 0;
END_RULE; -- restrict_action_request_status

RULE restrict_approval_status FOR (approval_status);
WHERE
  WR1: SIZEOF (QUERY (ast <* approval_status |
       NOT (ast.name IN
       ['approved', 'not_yet_approved', 'disapproved', 'withdrawn']))) = 0;
END_RULE; -- restrict_approval_status

RULE restrict_certification_type FOR (certification_type);
WHERE
  WR1: SIZEOF (QUERY (ct <* certification_type |
       NOT (ct.description IN
```

```
      ['design_supplier', 'part_supplier']))) = 0;
END_RULE; -- restrict_certification_type

RULE restrict_contract_type FOR (contract_type);
WHERE
  WR1: SIZEOF (QUERY (ct <* contract_type |
       NOT (ct.description IN ['fixed_price', 'cost_plus']))) = 0;
END_RULE; -- restrict_contract_type

RULE restrict_date_time_role FOR (date_time_role);
WHERE
  WR1: SIZEOF (QUERY (dtr <* date_time_role |
       NOT (dtr.name IN ['creation_date', 'request_date', 'release_date',
       'start_date', 'contract_date', 'certification_date', 'sign_off_date',
       'classification_date', 'declassification_date']))) = 0;
END_RULE; -- restrict_date_time_role

RULE restrict_document_type FOR (document_type);
WHERE
  WR1: SIZEOF (QUERY (dt <* document_type |
       NOT (dt.product_data_type IN
       ['material_specification', 'process_specification',
       'design_specification', 'surface_finish_specification',
       'cad_filename', 'drawing']))) = 0;
END_RULE; -- restrict_document_type

RULE restrict_person_organization_role FOR (person_and_organization_role);
WHERE
  WR1: SIZEOF (QUERY (por <* person_and_organization_role |
       NOT (por.name IN ['request_recipient', 'initiator', 'part_supplier',
       'design_supplier', 'configuration_manager', 'contractor',
       'classification_officer', 'creator', 'design_owner']))) = 0;
END_RULE; -- restrict_person_organization_role

RULE restrict_product_category_value FOR (product_related_product_category);
WHERE
  WR1: SIZEOF (QUERY (prpc <* product_related_product_category |
       NOT (prpc.name IN ['assembly', 'detail',
       'customer_furnished_equipment', 'inseparable_assembly', 'cast',
       'coined', 'drawn', 'extruded', 'forged', 'formed', 'machined',
       'molded', 'rolled', 'sheared'])))
       = 0;
END_RULE; -- restrict_product_category_value
```

```
RULE restrict_security_classification_level FOR
  (security_classification_level);
WHERE
  WR1: SIZEOF (QUERY (scl <* security_classification_level |
       NOT (scl.name IN ['unclassified', 'classified', 'proprietary',
       'confidential', 'secret', 'top_secret']))) = 0;
END_RULE; -- restrict_security_classification_level

RULE security_classification_optional_date_time FOR
  (security_classification, cc_design_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (sc <* security_classification |
       NOT (SIZEOF (QUERY (ccdta <* cc_design_date_and_time_assignment |
       (sc IN ccdta.items) AND
       ('declassification_date' = ccdta.role.name))) <= 1 ))) = 0;
END_RULE; -- security_classification_optional_date_time

RULE security_classification_requires_approval FOR
  (security_classification, cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (sc <* security_classification |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       sc IN ccda.items )) = 1 ))) = 0;
END_RULE; -- security_classification_requires_approval

RULE security_classification_requires_date_time FOR
  (security_classification, cc_design_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (sc <* security_classification |
       NOT (SIZEOF (QUERY (ccdta <* cc_design_date_and_time_assignment |
       (sc IN ccdta.items) AND
       ('classification_date' = ccdta.role.name))) = 1 ))) = 0;
END_RULE; -- security_classification_requires_date_time

RULE security_classification_requires_person_organization FOR
  (security_classification,
  cc_design_person_and_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (sc <* security_classification |
       NOT (SIZEOF (QUERY (ccdpoa <*
       cc_design_person_and_organization_assignment |
       sc IN ccdpoa.items )) = 1 ))) = 0;
END_RULE; -- security_classification_requires_person_organization
```

```
RULE start_request_requires_approval FOR
  (start_request, cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (sr <* start_request |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       sr IN ccda.items )) = 1 ))) = 0;
END_RULE; -- start_request_requires_approval

RULE start_request_requires_date_time FOR
  (start_request, cc_design_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (sr <* start_request |
       NOT (SIZEOF (QUERY (ccdta <* cc_design_date_and_time_assignment |
       sr IN ccdta.items )) = 1 ))) = 0;
END_RULE; -- start_request_requires_date_time

RULE start_request_requires_person_organization FOR (start_request,
  cc_design_person_and_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (sr <* start_request |
       NOT (SIZEOF (QUERY (ccdpoa <*
       cc_design_person_and_organization_assignment |
       sr IN ccdpoa.items )) >= 1 ))) = 0;
END_RULE; -- start_request_requires_person_organization

RULE start_work_requires_approval FOR
  (start_work, cc_design_approval);
WHERE
  WR1: SIZEOF (QUERY (sw <* start_work |
       NOT (SIZEOF (QUERY (ccda <* cc_design_approval |
       sw IN ccda.items )) = 1 ))) = 0;
END_RULE; -- start_work_requires_approval

RULE start_work_requires_date_time FOR
  (start_work, cc_design_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (sw <* start_work |
       NOT (SIZEOF (QUERY (ccdta <* cc_design_date_and_time_assignment |
       (sw IN ccdta.items)
       AND (ccdta.role.name = 'start_date'))) = 1 ))) = 0;
END_RULE; -- start_work_requires_date_time

RULE subtype_mandatory_action FOR (action);
WHERE
```

```
  WR1: SIZEOF (QUERY (act <* action |
       NOT ('CONFIG_CONTROL_DESIGN.DIRECTED_ACTION' IN TYPEOF(act)))) = 0;
END_RULE; -- subtype_mandatory_action

RULE subtype_mandatory_effectivity FOR
  (effectivity);
WHERE
  WR1: SIZEOF (QUERY (eff <* effectivity |
       NOT ((SIZEOF (['CONFIG_CONTROL_DESIGN.SERIAL_NUMBERED_EFFECTIVITY',
       'CONFIG_CONTROL_DESIGN.LOT_EFFECTIVITY',
       'CONFIG_CONTROL_DESIGN.DATED_EFFECTIVITY'] *
       TYPEOF (eff)) = 1) AND
       ('CONFIG_CONTROL_DESIGN.CONFIGURATION_EFFECTIVITY' IN TYPEOF(eff)))))
       = 0;
END_RULE; -- subtype_mandatory_effectivity

RULE subtype_mandatory_product_context FOR (product_context);
WHERE
  WR1: SIZEOF (QUERY (pc <* product_context |
       NOT ('CONFIG_CONTROL_DESIGN.MECHANICAL_CONTEXT' IN TYPEOF(pc)))) = 0;
END_RULE; -- subtype_mandatory_product_context

RULE subtype_mandatory_product_definition_formation FOR
  (product_definition_formation);
WHERE
  WR1: SIZEOF (QUERY (pdf <* product_definition_formation |
       NOT ('CONFIG_CONTROL_DESIGN.' +
       'PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE' IN
       TYPEOF(pdf)))) = 0;
END_RULE; -- subtype_mandatory_product_definition_formation

RULE subtype_mandatory_product_definition_usage FOR
  (product_definition_usage);
WHERE
  WR1: SIZEOF (QUERY (pdu <* product_definition_usage |
       NOT ('CONFIG_CONTROL_DESIGN.' +
       'ASSEMBLY_COMPONENT_USAGE' IN TYPEOF(pdu)))) = 0;
END_RULE; -- subtype_mandatory_product_definition_usage

RULE subtype_mandatory_representation FOR (representation);
WHERE
  WR1: SIZEOF (QUERY (rep <* representation |
       NOT ('CONFIG_CONTROL_DESIGN.SHAPE_REPRESENTATION' IN TYPEOF(rep))))
       = 0;
```

```
END_RULE; -- subtype_mandatory_representation

RULE subtype_mandatory_representation_context FOR
  (representation_context);
WHERE
  WR1: SIZEOF (QUERY (rep_cntxt <* representation_context |
       NOT ('CONFIG_CONTROL_DESIGN.GEOMETRIC_REPRESENTATION_CONTEXT' IN
       TYPEOF(rep_cntxt)))) = 0;
END_RULE; -- subtype_mandatory_representation_context

RULE subtype_mandatory_shape_representation FOR
  (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (sr <* shape_representation |
       NOT ((SIZEOF (['CONFIG_CONTROL_DESIGN.' +
       'ADVANCED_BREP_SHAPE_REPRESENTATION',
       'CONFIG_CONTROL_DESIGN.FACETED_BREP_SHAPE_REPRESENTATION',
       'CONFIG_CONTROL_DESIGN.MANIFOLD_SURFACE_SHAPE_REPRESENTATION',
       'CONFIG_CONTROL_DESIGN.EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION',
       'CONFIG_CONTROL_DESIGN.SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION',
       'CONFIG_CONTROL_DESIGN.' +
       'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION',
       'CONFIG_CONTROL_DESIGN.' +
       'GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION'] *
       TYPEOF (sr)) = 1) OR
       (SIZEOF (QUERY (it <* sr\representation.items |
       NOT ('CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D' IN TYPEOF (it))))
       = 0) OR
       (SIZEOF (QUERY (sdr <* QUERY (pdr <* USEDIN (sr,
       'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION_REPRESENTATION.' +
       'USED_REPRESENTATION') |
       'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION' IN
       TYPEOF (pdr)) |
       NOT (SIZEOF (['CONFIG_CONTROL_DESIGN.SHAPE_ASPECT',
       'CONFIG_CONTROL_DESIGN.SHAPE_ASPECT_RELATIONSHIP'] * TYPEOF
       (sdr.definition)) = 1))) = 0))) = 0;
END_RULE; -- subtype_mandatory_shape_representation

RULE unique_version_change_order_rule FOR (change);
WHERE
  WR1: SIZEOF (QUERY (c <* change |
       NOT (unique_version_change_order (c.assigned_action)))) = 0;
END_RULE; -- unique_version_change_order_rule
```

```
RULE versioned_action_request_requires_solution FOR
  (versioned_action_request, action_request_solution);
WHERE
  WR1: SIZEOF (QUERY (ar <* versioned_action_request |
       NOT (SIZEOF (QUERY (ars <* action_request_solution |
       ar :=: ars.request)) >= 1))) = 0;
END_RULE; -- versioned_action_request_requires_solution

RULE versioned_action_request_requires_status FOR
  (versioned_action_request, action_request_status);
WHERE
  WR1: SIZEOF (QUERY (ar <* versioned_action_request |
       NOT (SIZEOF (QUERY (ars <* action_request_status |
       ar :=: ars.assigned_request)) = 1))) = 0;
END_RULE; -- versioned_action_request_requires_status


FUNCTION acyclic_curve_replica
  (rep: curve_replica; parent: curve): BOOLEAN;
IF NOT ('CONFIG_CONTROL_DESIGN.CURVE_REPLICA' IN TYPEOF(parent)) THEN
  RETURN(TRUE);
END_IF;
IF parent :=: rep THEN
  RETURN(FALSE);
ELSE
  RETURN(acyclic_curve_replica(rep,parent\curve_replica.parent_curve));
END_IF;

END_FUNCTION; -- acyclic_curve_replica

FUNCTION acyclic_mapped_representation
  (parent_set: SET OF representation;
  children_set: SET OF representation_item): BOOLEAN;

LOCAL
  i : INTEGER;
  x : SET OF representation_item;
  y : SET OF representation_item;
END_LOCAL;
x := QUERY ( z <* children_set | ('CONFIG_CONTROL_DESIGN.MAPPED_ITEM'
    IN TYPEOF(z)) );
IF SIZEOF(x) > 0 THEN
  REPEAT i := 1 TO HIINDEX(x) BY 1;
    IF x[i]\mapped_item.mapping_source.mapped_representation IN
```

```
        parent_set THEN
      RETURN(FALSE);
    END_IF;
    IF NOT acyclic_mapped_representation(parent_set + x[i]\mapped_item
        .mapping_source.mapped_representation,x[i]\mapped_item.
        mapping_source.mapped_representation.items) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_IF;
x := children_set - x;
IF SIZEOF(x) > 0 THEN
  REPEAT i := 1 TO HIINDEX(x) BY 1;
    y := QUERY ( z <* bag_to_set(USEDIN(x[i],'')) | (
        'CONFIG_CONTROL_DESIGN.REPRESENTATION_ITEM' IN TYPEOF(z)) );
    IF NOT acyclic_mapped_representation(parent_set,y) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_IF;
RETURN(TRUE);

END_FUNCTION; -- acyclic_mapped_representation

FUNCTION acyclic_point_replica
  (rep: point_replica; parent: point): BOOLEAN;
IF NOT ('CONFIG_CONTROL_DESIGN.POINT_REPLICA' IN TYPEOF(parent)) THEN
  RETURN(TRUE);
END_IF;
IF parent :=: rep THEN
  RETURN(FALSE);
ELSE
  RETURN(acyclic_point_replica(rep,parent\point_replica.parent_pt));
END_IF;

END_FUNCTION; -- acyclic_point_replica

FUNCTION acyclic_product_category_relationship
  (relation: product_category_relationship;
  children: SET OF product_category): LOGICAL;

LOCAL
  i              : INTEGER;
  x              : SET OF product_category_relationship;
```

```
  local_children : SET OF product_category;
END_LOCAL;
REPEAT i := 1 TO HIINDEX(children) BY 1;
  IF relation.category :=: children[i] THEN
    RETURN(FALSE);
  END_IF;
END_REPEAT;
x := bag_to_set(USEDIN(relation.category,'CONFIG_CONTROL_DESIGN.' +
    'PRODUCT_CATEGORY_RELATIONSHIP.SUB_CATEGORY'));
local_children := children + relation.category;
IF SIZEOF(x) > 0 THEN
  REPEAT i := 1 TO HIINDEX(x) BY 1;
    IF NOT acyclic_product_category_relationship(x[i],local_children)
         THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_IF;
RETURN(TRUE);

END_FUNCTION; -- acyclic_product_category_relationship

FUNCTION acyclic_product_definition_relationship
  (relation: product_definition_relationship;
  relatives: SET OF product_definition;
  specific_relation: STRING): LOGICAL;

LOCAL
  i               : INTEGER;
  x               : SET OF product_definition_relationship;
  local_relatives : SET OF product_definition;
END_LOCAL;
REPEAT i := 1 TO HIINDEX(relatives) BY 1;
  IF relation.relating_product_definition :=: relatives[i] THEN
    RETURN(FALSE);
  END_IF;
END_REPEAT;
x := bag_to_set(USEDIN(relation.relating_product_definition,
    specific_relation));
local_relatives := relatives + relation.relating_product_definition;
IF SIZEOF(x) > 0 THEN
  REPEAT i := 1 TO HIINDEX(x) BY 1;
    IF NOT acyclic_product_definition_relationship(x[i],
        local_relatives,specific_relation) THEN
```

```
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_IF;
RETURN(TRUE);

END_FUNCTION; -- acyclic_product_definition_relationship

FUNCTION acyclic_surface_replica
  (rep: surface_replica; parent: surface): BOOLEAN;
IF NOT ('CONFIG_CONTROL_DESIGN.SURFACE_REPLICA' IN TYPEOF(parent))
     THEN
  RETURN(TRUE);
END_IF;
IF parent :=: rep THEN
  RETURN(FALSE);
ELSE
  RETURN(acyclic_surface_replica(rep,parent\surface_replica.
      parent_surface));
END_IF;

END_FUNCTION; -- acyclic_surface_replica

FUNCTION assembly_shape_is_defined
  (assy: next_assembly_usage_occurrence) : BOOLEAN;
  LOCAL
        sdr_set : SET OF shape_definition_representation := [];
        srr_set : SET OF shape_representation_relationship := [];
  END_LOCAL;

  -- Gather all instances of shape_definition_representation where the
  -- component part has a representation defined for it.

  sdr_set := QUERY (pdr <* bag_to_set (USEDIN
             (assy.related_product_definition,
             'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION_REPRESENTATION.' +
             'DEFINITION.DEFINITION')) |
             'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION' IN
             TYPEOF (pdr)) +
             QUERY (pdr <* bag_to_set (
             USEDIN (assy.related_product_definition,
             'CONFIG_CONTROL_DESIGN.' +
             'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION.' +
             'DEFINITION.RELATED_PRODUCT_DEFINITION')) |
```

```
          'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION' IN
          TYPEOF (pdr));

-- If there is a representation defined for the component part

IF SIZEOF (sdr_set) > 0 THEN

  -- For each representation of the shape of the component part gather all
  -- instances of shape_representation_relationship where the
  -- representation of component part is related to another
  -- representation.

  REPEAT i := 1 TO HIINDEX (sdr_set);
    srr_set := QUERY (rr <* bag_to_set (
               USEDIN (sdr_set[i]\
               property_definition_representation.used_representation,
               'CONFIG_CONTROL_DESIGN.REPRESENTATION_RELATIONSHIP.REP_2')) |
               'CONFIG_CONTROL_DESIGN.SHAPE_REPRESENTATION_RELATIONSHIP' IN
               TYPEOF (rr));

    -- If there is a shape_representation_relationship where the component
    -- part's shape_representation is related to another shape_-
    -- representation.

    IF SIZEOF (srr_set) > 0 THEN

      -- For each shape_representation_relationship in that set

      REPEAT j := 1 TO HIINDEX (srr_set);

        -- If the other shape_representation in the shape_representation_-
        -- relationship is the  shape_representation of the assembly
        -- product_definition in at least one instance.

        IF SIZEOF (QUERY (pdr <* bag_to_set (USEDIN
          (srr_set[j]\representation_relationship.rep_1,
          'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION_REPRESENTATION.' +
          'USED_REPRESENTATION')) |
          'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION' IN
          TYPEOF (pdr)) *
          QUERY (pdr <* bag_to_set (
          USEDIN (assy.relating_product_definition,
          'CONFIG_CONTROL_DESIGN.PROPERTY_DEFINITION_REPRESENTATION.' +
          'USED_REPRESENTATION.DEFINITION')) |
```

```
            'CONFIG_CONTROL_DESIGN.SHAPE_DEFINITION_REPRESENTATION'
            IN TYPEOF (pdr))) >= 1 THEN

            -- If the shape_representation_relationship and the product_-
            -- definition_relationship of each occurrence of the component
            -- and assembly relationship is not given via the
            -- context_dependent_shape_representation then return FALSE


            IF SIZEOF (QUERY (cdsr <*
              USEDIN (assy, 'CONFIG_CONTROL_DESIGN.' +
              'CONTEXT_DEPENDENT_SHAPE_REPRESENTATION.' +
              'REPRESENTED_PRODUCT_RELATION') |
              NOT ((cdsr.represented_product_relation.definition :=: assy) AND
              (cdsr.representation_relation :=: srr_set[j]))))) > 0
              THEN RETURN (FALSE);
            END_IF;
          END_IF;
        END_REPEAT;
      END_IF;
    END_REPEAT;
  END_IF;

  -- If the shape of the component is not specified or there are no
  -- violations then return TRUE

  RETURN (TRUE);
END_FUNCTION; -- assembly_shape_is_defined

FUNCTION associated_surface
  (arg: pcurve_or_surface): surface;

LOCAL
  surf : surface;
END_LOCAL;
IF 'CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(arg) THEN
  surf := arg.basis_surface;
ELSE
  surf := arg;
END_IF;
RETURN(surf);

END_FUNCTION; -- associated_surface
```

```
FUNCTION bag_to_set
  (the_bag : BAG OF GENERIC:intype) : SET OF GENERIC:intype;
  LOCAL
    the_set : SET OF GENERIC:intype := [];
    i       : INTEGER;
  END_LOCAL;

  IF SIZEOF (the_bag) > 0 THEN
    REPEAT i := 1 to HIINDEX (the_bag);
      the_set := the_set + the_bag[i];
    END_REPEAT;
  END_IF;

  RETURN (the_set);

END_FUNCTION; -- bag_to_set

FUNCTION base_axis
  (dim: INTEGER;
  axis1, axis2, axis3: direction): LIST [2:3] OF direction;

LOCAL
  u      : LIST [2:3] OF direction;
  vec    : direction;
  factor : REAL;
END_LOCAL;
IF dim = 3 THEN
  u[3] := NVL(normalise(axis3),direction([0,0,1]));
  u[1] := first_proj_axis(u[3],axis1);
  u[2] := second_proj_axis(u[3],u[1],axis2);
ELSE
  u[3] := ?;
  IF EXISTS(axis1) THEN
    u[1] := normalise(axis1);
    u[2] := orthogonal_complement(u[1]);
    IF EXISTS(axis2) THEN
      factor := dot_product(axis2,u[2]);
      IF factor < 0 THEN
        u[2].direction_ratios[1] := -u[2].direction_ratios[1];
        u[2].direction_ratios[2] := -u[2].direction_ratios[2];
      END_IF;
    END_IF;
  ELSE
    IF EXISTS(axis2) THEN
```

```
      u[2] := normalise(axis2);
      u[1] := orthogonal_complement(u[2]);
      u[1].direction_ratios[1] := -u[1].direction_ratios[1];
      u[1].direction_ratios[2] := -u[1].direction_ratios[2];
    ELSE
      u[1].direction_ratios[1] := 1;
      u[1].direction_ratios[2] := 0;
      u[2].direction_ratios[1] := 0;
      u[2].direction_ratios[2] := 1;
    END_IF;
  END_IF;
END_IF;
RETURN(u);

END_FUNCTION; -- base_axis

FUNCTION basis_curve_check
  (cv: curve; schema_name : STRING) : BOOLEAN;

  -- let those types pass that do not have references to curves

  IF SIZEOF ([schema_name + '.B_SPLINE_CURVE',
    schema_name + '.CONIC',
    schema_name + '.LINE',
    schema_name + '.POLYLINE'] * TYPEOF (cv)) = 1 THEN
    RETURN(TRUE);
  ELSE

    -- check in case curve_replica is the type of the input curve

    IF schema_name + '.CURVE_REPLICA' IN TYPEOF (cv) THEN
      RETURN (basis_curve_check(cv\curve_replica.parent_curve, schema_name));
    ELSE

      --check in case offset_curve_3d is the type of the input curve

      IF schema_name + '.OFFSET_CURVE_3D' IN TYPEOF (cv) THEN
        RETURN(basis_curve_check
        (cv\offset_curve_3d.basis_curve, schema_name));
      ELSE

          -- check in case pcurve is the type of the input curve

        IF schema_name + '.PCURVE' IN TYPEOF(cv) THEN
```

```
IF SIZEOF ([schema_name + '.B_SPLINE_CURVE',
  schema_name + '.CONIC',
  schema_name + '.LINE',
  schema_name + '.POLYLINE'] *
  TYPEOF (cv\pcurve.reference_to_curve\representation.items[1]))
  = 1 THEN
  RETURN (basis_surface_check (cv\pcurve.basis_surface,
  schema_name));
ELSE
  IF schema_name + '.CURVE_REPLICA'
    IN TYPEOF
    (cv\pcurve.reference_to_curve\representation.items[1]) THEN
    RETURN ((basis_curve_check
    (cv\pcurve.reference_to_curve\representation.items[1],
    schema_name)) AND
    (basis_surface_check (cv\pcurve.basis_surface,
    schema_name)));
  END_IF;
END_IF;
ELSE

--check in case surface_curve is the type of the input curve

IF schema_name + '.SURFACE_CURVE' IN TYPEOF(cv) THEN
  IF SIZEOF ([schema_name + '.B_SPLINE_CURVE',
    schema_name + '.CONIC',
    schema_name + '.LINE',
    schema_name + '.POLYLINE'] *
    TYPEOF (cv\surface_curve.curve_3d)) = 1 THEN

    REPEAT i := 1 TO SIZEOF (cv\surface_curve.associated_geometry);


      -- do for one or two associated_geometrys:

      IF schema_name + '.SURFACE' IN
        TYPEOF (cv\surface_curve.associated_geometry[i]) THEN
        IF NOT basis_surface_check
          (cv\surface_curve.associated_geometry[i],
          schema_name) THEN
          RETURN(FALSE);
        END_IF;
      ELSE
        IF schema_name + '.PCURVE' IN
```

```
        TYPEOF (cv\surface_curve.associated_geometry[i]) THEN
        IF NOT basis_curve_check
          (cv\surface_curve.associated_geometry[i],
          schema_name) THEN RETURN(FALSE);
        END_IF;
      END_IF;
    END_IF;
  END_REPEAT;
ELSE
  IF schema_name + '.CURVE_REPLICA' IN TYPEOF
    (cv\surface_curve.curve_3d) THEN
    IF NOT (basis_curve_check
      (cv\curve_replica.parent_curve, schema_name)) THEN
      RETURN (FALSE);
    END_IF;
  ELSE
    IF schema_name + '.OFFSET_CURVE_3D' IN TYPEOF
      (cv\surface_curve.curve_3d) THEN
      IF NOT (basis_curve_check
        (cv\offset_curve_3d.basis_curve, schema_name)) THEN
        RETURN (FALSE);
      END_IF;
    ELSE
      REPEAT i := 1 TO SIZEOF
      (cv\surface_curve.associated_geometry);

        -- do for one or two associated_geometrys:

        IF schema_name + '.SURFACE' IN
          TYPEOF (cv\surface_curve.associated_geometry[i]) THEN
          IF NOT basis_surface_check
            (cv\surface_curve.associated_geometry[i],
            schema_name) THEN
            RETURN(FALSE);
          END_IF;
        ELSE
          IF schema_name + '.PCURVE' IN
            TYPEOF (cv\surface_curve.associated_geometry[i]) THEN

            IF NOT basis_curve_check
              (cv\surface_curve.associated_geometry[i],
              schema_name) THEN
              RETURN(FALSE);
            END_IF;
```

```
                    END_IF;
                  END_IF;
                END_REPEAT;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION; -- basis_curve_check

FUNCTION basis_surface_check
  (surf : surface; schema_name : STRING) : BOOLEAN;


-- let those surface types pass that do not have references to surfaces

  IF SIZEOF([schema_name + '.B_SPLINE_SURFACE',
    schema_name + '.ELEMENTARY_SURFACE'] * TYPEOF(surf)) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF schema_name + '.SWEPT_SURFACE' IN TYPEOF (surf) THEN
      RETURN (basis_curve_check
      (surf\swept_surface.swept_curve, schema_name));
    ELSE

      -- check in case offset_surface is the type of the input surface

      IF schema_name + '.OFFSET_SURFACE' IN TYPEOF (surf) THEN
        RETURN(basis_surface_check
        (surf\offset_surface.basis_surface, schema_name));
      ELSE

        -- check in case surface_replica is the type of the input surface

        IF schema_name + '.SURFACE REPLICA' IN TYPEOF(surf) THEN
          RETURN(basis_surface_check
          (surf\surface_replica.parent_surface, schema_name));
        END_IF;
      END_IF;
    END_IF;
```

```
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- basis_surface_check

FUNCTION boolean_choose
  (b: BOOLEAN;
  choice1, choice2: GENERIC): GENERIC;
IF b THEN
  RETURN(choice1);
ELSE
  RETURN(choice2);
END_IF;

END_FUNCTION; -- boolean_choose

FUNCTION build_2axes
  (ref_direction: direction): LIST [2:2] OF direction;

LOCAL
  u : LIST [2:2] OF direction;
END_LOCAL;
u[1] := NVL(normalise(ref_direction),direction([1,0]));
u[2] := orthogonal_complement(u[1]);
RETURN(u);

END_FUNCTION; -- build_2axes

FUNCTION build_axes
  (axis, ref_direction: direction): LIST [3:3] OF direction;

LOCAL
  u : LIST [3:3] OF direction;
END_LOCAL;
u[3] := NVL(normalise(axis),direction([0,0,1]));
u[1] := first_proj_axis(u[3],ref_direction);
u[2] := normalise(cross_product(u[3],u[1])).orientation;
RETURN(u);

END_FUNCTION; -- build_axes

FUNCTION cc_design_date_time_correlation
  (e : cc_design_date_and_time_assignment ) : BOOLEAN;
  LOCAL
    dt_role : STRING;
```

```
END_LOCAL;
  dt_role := e\date_and_time_assignment.role.name;
CASE dt_role OF
  'creation_date'          : IF SIZEOF (e.items) <>
                               SIZEOF (QUERY (x <* e.items |
                               'CONFIG_CONTROL_DESIGN.' +
                               'PRODUCT_DEFINITION'
                               IN TYPEOF (x)))
                               THEN RETURN(FALSE);
                             END_IF;
  'request_date'           : IF SIZEOF (e.items) <>
                               SIZEOF (QUERY (x <* e.items |
                               SIZEOF (
                               ['CONFIG_CONTROL_DESIGN.CHANGE_REQUEST' +
                               'CONFIG_CONTROL_DESIGN.START_REQUEST'] *
                               TYPEOF (x)) = 1))
                               THEN RETURN(FALSE);
                             END_IF;
  'release_date'           : IF SIZEOF (e.items) <>
                               SIZEOF (QUERY (x <* e.items |
                               SIZEOF (
                               ['CONFIG_CONTROL_DESIGN.CHANGE' +
                               'CONFIG_CONTROL_DESIGN.START_WORK'] *
                               TYPEOF (x)) = 1))
                               THEN RETURN(FALSE);
                             END_IF;
  'start_date'             : IF SIZEOF (e.items) <>
                               SIZEOF (QUERY (x <* e.items |
                               SIZEOF (
                               ['CONFIG_CONTROL_DESIGN.CHANGE' +
                               'CONFIG_CONTROL_DESIGN.START_WORK'] *
                               TYPEOF (x)) = 1))
                               THEN RETURN(FALSE);
                             END_IF;
  'sign_off_date'          : IF SIZEOF (e.items) <>
                               SIZEOF (QUERY (x <* e.items |
                               'CONFIG_CONTROL_DESIGN.' +
                               'APPROVAL_PERSON_ORGANIZATION'
                               IN TYPEOF (x)))
                               THEN RETURN(FALSE);
                             END_IF;
  'contract_date'          : IF SIZEOF (e.items) <>
                               SIZEOF (QUERY (x <* e.items |
                               'CONFIG_CONTROL_DESIGN.CONTRACT'
```

```
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'certification_date'    : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'CONFIG_CONTROL_DESIGN.CERTIFICATION'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'classification_date'   : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'CONFIG_CONTROL_DESIGN.' +
                                'SECURITY_CLASSIFICATION'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'declassification_date' : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'CONFIG_CONTROL_DESIGN.' +
                                'SECURITY_CLASSIFICATION'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    OTHERWISE : RETURN(TRUE);
  END_CASE;
  RETURN (TRUE);
END_FUNCTION; -- cc_design_date_time_correlation

FUNCTION cc_design_person_and_organization_correlation
  (e : cc_design_person_and_organization_assignment ) : BOOLEAN;
  LOCAL
    po_role : STRING;
  END_LOCAL;
    po_role := e\person_and_organization_assignment.role.name;
  CASE po_role OF
    'request_recipient'      : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                SIZEOF(['CONFIG_CONTROL_DESIGN.' +
                                'CHANGE_REQUEST',
                                'CONFIG_CONTROL_DESIGN.' +
                                'START_REQUEST'] *
                                TYPEOF (x)) = 1))
                                THEN RETURN(FALSE);
                               END_IF;
```

```
'initiator'              : IF SIZEOF (e.items) <>
                             SIZEOF (QUERY (x <* e.items |
                             SIZEOF(['CONFIG_CONTROL_DESIGN.' +
                             'CHANGE_REQUEST',
                             'CONFIG_CONTROL_DESIGN.' +
                             'START_REQUEST',
                             'CONFIG_CONTROL_DESIGN.' +
                             'START_WORK',
                             'CONFIG_CONTROL_DESIGN.' +
                             'CHANGE'] *
                             TYPEOF (x)) = 1))
                             THEN RETURN(FALSE);
                           END_IF;
'creator'                : IF SIZEOF (e.items) <>
                             SIZEOF (QUERY (x <* e.items |
                             SIZEOF (['CONFIG_CONTROL_DESIGN.' +
                             'PRODUCT_DEFINITION_FORMATION',
                             'CONFIG_CONTROL_DESIGN.' +
                             'PRODUCT_DEFINITION'] *
                             TYPEOF (x)) = 1))
                             THEN RETURN (FALSE);
                           END_IF;
'part_supplier'          : IF SIZEOF (e.items) <>
                             SIZEOF (QUERY (x <* e.items |
                             'CONFIG_CONTROL_DESIGN.' +
                             'PRODUCT_DEFINITION_FORMATION'
                             IN TYPEOF (x)))
                             THEN RETURN(FALSE);
                           END_IF;
'design_supplier'        : IF SIZEOF (e.items) <>
                             SIZEOF (QUERY (x <* e.items |
                             'CONFIG_CONTROL_DESIGN.' +
                             'PRODUCT_DEFINITION_FORMATION'
                             IN TYPEOF (x)))
                             THEN RETURN(FALSE);
                           END_IF;
'design_owner'           : IF SIZEOF (e.items) <>
                             SIZEOF (QUERY (x <* e.items |
                             'CONFIG_CONTROL_DESIGN.PRODUCT'
                             IN TYPEOF (x)))
                             THEN RETURN(FALSE);
                           END_IF;
'configuration_manager'  : IF SIZEOF (e.items) <>
                             SIZEOF (QUERY (x <* e.items |
```

```
                                'CONFIG_CONTROL_DESIGN.' +
                                'CONFIGURATION_ITEM'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'contractor'             : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'CONFIG_CONTROL_DESIGN.CONTRACT'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'classification_officer' : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'CONFIG_CONTROL_DESIGN.' +
                                'SECURITY_CLASSIFICATION'
                                IN TYPEOF (x))) THEN
                                RETURN(FALSE);
                              END_IF;
    OTHERWISE : RETURN(TRUE);
  END_CASE;
  RETURN (TRUE);
END_FUNCTION; -- cc_design_person_and_organization_correlation

FUNCTION conditional_reverse
  (p: BOOLEAN; an_item: reversible_topology): reversible_topology;
IF p THEN
  RETURN(an_item);
ELSE
  RETURN(topology_reversed(an_item));
END_IF;

END_FUNCTION; -- conditional_reverse

FUNCTION constraints_composite_curve_on_surface
  (c: composite_curve_on_surface): BOOLEAN;

LOCAL
  n_segments : INTEGER := SIZEOF(c.segments);
END_LOCAL;
REPEAT k := 1 TO n_segments BY 1;
  IF ((NOT ('CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(c\composite_curve
      .segments[k].parent_curve))) AND (NOT (
      'CONFIG_CONTROL_DESIGN.SURFACE_CURVE' IN TYPEOF(c\composite_curve
      .segments[k].parent_curve)))) AND (NOT (
```

```
      'CONFIG_CONTROL_DESIGN.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF(c\
      composite_curve.segments[k].parent_curve))) THEN
    RETURN(FALSE);
  END_IF;
END_REPEAT;
RETURN(TRUE);

END_FUNCTION; -- constraints_composite_curve_on_surface

FUNCTION constraints_geometry_shell_based_surface_model
  (m: shell_based_surface_model): BOOLEAN;

LOCAL
  result : BOOLEAN := TRUE;
END_LOCAL;
REPEAT j := 1 TO SIZEOF(m.sbsm_boundary) BY 1;
  IF (NOT ('CONFIG_CONTROL_DESIGN.OPEN_SHELL' IN TYPEOF(m.
      sbsm_boundary[j]))) AND (NOT (
      'CONFIG_CONTROL_DESIGN.CLOSED_SHELL' IN TYPEOF(m.sbsm_boundary[j])))
       THEN
    result := FALSE;
    RETURN(result);
  END_IF;
END_REPEAT;
RETURN(result);

END_FUNCTION; -- constraints_geometry_shell_based_surface_model

FUNCTION constraints_geometry_shell_based_wireframe_model
  (m: shell_based_wireframe_model): BOOLEAN;

LOCAL
  result : BOOLEAN := TRUE;
END_LOCAL;
REPEAT j := 1 TO SIZEOF(m.sbwm_boundary) BY 1;
  IF (NOT ('CONFIG_CONTROL_DESIGN.WIRE_SHELL' IN TYPEOF(m.
      sbwm_boundary[j]))) AND (NOT (
      'CONFIG_CONTROL_DESIGN.VERTEX_SHELL' IN TYPEOF(m.sbwm_boundary[j])))
       THEN
    result := FALSE;
    RETURN(result);
  END_IF;
END_REPEAT;
RETURN(result);
```

```
END_FUNCTION; -- constraints_geometry_shell_based_wireframe_model

FUNCTION constraints_param_b_spline
  (degree, up_knots, up_cp: INTEGER;
  knot_mult: LIST OF INTEGER;
  knots: LIST OF parameter_value): BOOLEAN;

LOCAL
  k      : INTEGER;
  l      : INTEGER;
  sum    : INTEGER;
  result : BOOLEAN := TRUE;
END_LOCAL;
sum := knot_mult[1];
REPEAT i := 2 TO up_knots BY 1;
  sum := sum + knot_mult[i];
END_REPEAT;
IF (((degree < 1) OR (up_knots < 2)) OR (up_cp < degree)) OR (sum <> (
   (degree + up_cp) + 2)) THEN
  result := FALSE;
  RETURN(result);
END_IF;
k := knot_mult[1];
IF (k < 1) OR (k > (degree + 1)) THEN
  result := FALSE;
  RETURN(result);
END_IF;
REPEAT i := 2 TO up_knots BY 1;
  IF (knot_mult[i] < 1) OR (knots[i] <= knots[i - 1]) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
  k := knot_mult[i];
  IF (i < up_knots) AND (k > degree) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
  IF (i = up_knots) AND (k > (degree + 1)) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
END_REPEAT;
RETURN(result);
```

```
END_FUNCTION; -- constraints_param_b_spline

FUNCTION constraints_rectangular_composite_surface
  (s: rectangular_composite_surface): BOOLEAN;
REPEAT i := 1 TO s.n_u BY 1;
  REPEAT j := 1 TO s.n_v BY 1;
    IF NOT (('CONFIG_CONTROL_DESIGN.B_SPLINE_SURFACE' IN TYPEOF(s.
        segments[i][j].parent_surface)) OR (
        'CONFIG_CONTROL_DESIGN.RECTANGULAR_TRIMMED_SURFACE' IN TYPEOF(s
        .segments[i][j].parent_surface))) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_REPEAT;
REPEAT i := 1 TO s.n_u - 1 BY 1;
  REPEAT j := 1 TO s.n_v BY 1;
    IF s.segments[i][j].u_transition = discontinuous THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_REPEAT;
REPEAT i := 1 TO s.n_u BY 1;
  REPEAT j := 1 TO s.n_v - 1 BY 1;
    IF s.segments[i][j].v_transition = discontinuous THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_REPEAT;
RETURN(TRUE);

END_FUNCTION; -- constraints_rectangular_composite_surface

FUNCTION cross_product
  (arg1, arg2: direction): vector;

LOCAL
  v2     : LIST [3:3] OF REAL;
  v1     : LIST [3:3] OF REAL;
  mag    : REAL;
  res    : direction;
  result : vector;
END_LOCAL;
IF ((NOT EXISTS(arg1)) OR (arg1.dim = 2)) OR ((NOT EXISTS(arg2)) OR (
    arg2.dim = 2)) THEN
```

```
  RETURN(?);
ELSE
  BEGIN
    v1 := normalise(arg1).direction_ratios;
    v2 := normalise(arg2).direction_ratios;
    res.direction_ratios[1] := (v1[2] * v2[3]) - (v1[3] * v2[2]);
    res.direction_ratios[2] := (v1[3] * v2[1]) - (v1[1] * v2[3]);
    res.direction_ratios[3] := (v1[1] * v2[2]) - (v1[2] * v2[1]);
    mag := 0;
    REPEAT i := 1 TO 3 BY 1;
      mag := mag + (res.direction_ratios[i] * res.direction_ratios[i]);
    END_REPEAT;
    IF mag > 0 THEN
      result.orientation := res;
      result.magnitude := SQRT(mag);
    ELSE
      result.orientation := arg1;
      result.magnitude := 0;
    END_IF;
    RETURN(result);
  END;
END_IF;

END_FUNCTION; -- cross_product

FUNCTION curve_weights_positive
  (b: rational_b_spline_curve): BOOLEAN;

LOCAL
  result : BOOLEAN := TRUE;
END_LOCAL;
REPEAT i := 0 TO b.upper_index_on_control_points BY 1;
  IF b.weights[i] <= 0 THEN
    result := FALSE;
    RETURN(result);
  END_IF;
END_REPEAT;
RETURN(result);

END_FUNCTION; -- curve_weights_positive

FUNCTION derive_dimensional_exponents
  (x: unit): dimensional_exponents;
```

```
LOCAL
  result : dimensional_exponents := dimensional_exponents(0,0,0,0,0,0,
           0);
END_LOCAL;
  result := x.dimensions;
RETURN(result);

END_FUNCTION; -- derive_dimensional_exponents

FUNCTION dimension_of
  (item : geometric_representation_item) :
  dimension_count;
  LOCAL
    x : SET OF representation;
    y : representation_context;
  END_LOCAL;

  -- Find the set of representation in which the item is used.

  x := using_representations(item);

  -- If there is such a representation, then determine the dimension_count
  -- of the associated geometric_representation_context - note that the
  -- RULE compatible_dimension ensures that the context_of_items is of type
  -- geometric_representation_context and has the same dimension_count for
  -- all values of x.

  IF SIZEOF(x) > 0 THEN
    y := x[1].context_of_items;
    RETURN (y\geometric_representation_context.coordinate_space_dimension);
  END_IF;
END_FUNCTION; -- dimension_of

FUNCTION dimensions_for_si_unit
  (n: si_unit_name): dimensional_exponents;
CASE n OF
  metre          :        RETURN(dimensional_exponents(1,0,0,0,0,0,0));
  gram           :        RETURN(dimensional_exponents(0,1,0,0,0,0,0));
  second         :        RETURN(dimensional_exponents(0,0,1,0,0,0,0));
  ampere         :        RETURN(dimensional_exponents(0,0,0,1,0,0,0));
  kelvin         :        RETURN(dimensional_exponents(0,0,0,0,1,0,0));
  mole           :        RETURN(dimensional_exponents(0,0,0,0,0,1,0));
  candela        :        RETURN(dimensional_exponents(0,0,0,0,0,0,1));
  radian         :        RETURN(dimensional_exponents(0,0,0,0,0,0,0));
```

```
  steradian       :        RETURN(dimensional_exponents(0,0,0,0,0,0,0));
  hertz           :        RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
  newton          :        RETURN(dimensional_exponents(1,1,-2,0,0,0,0));
  pascal          :        RETURN(dimensional_exponents(-1,1,-2,0,0,0,0));
  joule           :        RETURN(dimensional_exponents(2,1,-2,0,0,0,0));
  watt            :        RETURN(dimensional_exponents(2,1,-3,0,0,0,0));
  coulomb         :        RETURN(dimensional_exponents(0,0,1,1,0,0,0));
  volt            :        RETURN(dimensional_exponents(2,1,-3,-1,0,0,0));
  farad           :        RETURN(dimensional_exponents(-2,-1,4,1,0,0,0));
  ohm             :        RETURN(dimensional_exponents(2,1,-3,-2,0,0,0));
  siemens         :        RETURN(dimensional_exponents(-2,-1,3,2,0,0,0));
  weber           :        RETURN(dimensional_exponents(2,1,-2,-1,0,0,0));
  tesla           :        RETURN(dimensional_exponents(0,1,-2,-1,0,0,0));
  henry           :        RETURN(dimensional_exponents(2,1,-2,-2,0,0,0));
  degree_celsius  :        RETURN(dimensional_exponents(0,0,0,0,1,0,0));
  lumen           :        RETURN(dimensional_exponents(0,0,0,0,0,0,1));
  lux             :        RETURN(dimensional_exponents(-2,0,0,0,0,0,1));
  becquerel       :        RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
  gray            :        RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
  sievert         :        RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
  END_CASE;

END_FUNCTION; -- dimensions_for_si_unit

FUNCTION dot_product
  (arg1, arg2: direction): REAL;

LOCAL
  ndim   : INTEGER;
  scalar : REAL;
  vec1   : direction;
  vec2   : direction;
END_LOCAL;
IF (NOT EXISTS(arg1)) OR (NOT EXISTS(arg2)) THEN
  scalar := ?;
ELSE
  IF arg1.dim <> arg2.dim THEN
    scalar := ?;
  ELSE
    BEGIN
      vec1 := normalise(arg1);
      vec2 := normalise(arg2);
      ndim := arg1.dim;
      scalar := 0;
```

```
      REPEAT i := 1 TO ndim BY 1;
        scalar := scalar + (vec1.direction_ratios[i] * vec2.
            direction_ratios[i]);
      END_REPEAT;
    END;
  END_IF;
END_IF;
RETURN(scalar);

END_FUNCTION; -- dot_product

FUNCTION edge_reversed
  (an_edge: edge): edge;

LOCAL
  the_reverse : edge;
END_LOCAL;
IF 'CONFIG_CONTROL_DESIGN.ORIENTED_EDGE' IN TYPEOF(an_edge) THEN
  the_reverse := oriented_edge(an_edge\oriented_edge.edge_element,NOT
      an_edge\oriented_edge.orientation);
ELSE
  the_reverse := oriented_edge(an_edge,FALSE);
END_IF;
RETURN(the_reverse);

END_FUNCTION; -- edge_reversed

FUNCTION face_bound_reversed
  (a_face_bound: face_bound): face_bound;

LOCAL
  the_reverse : face_bound;
END_LOCAL;
IF 'CONFIG_CONTROL_DESIGN.FACE_OUTER_BOUND' IN TYPEOF(a_face_bound)
     THEN
  the_reverse := face_bound(a_face_bound\face_bound.bound,
                   (NOT (a_face_bound\face_bound.orientation)) ) ||
                   face_outer_bound ();
ELSE
  the_reverse := face_bound(a_face_bound.bound,NOT a_face_bound.
      orientation);
END_IF;
RETURN(the_reverse);
```

```
END_FUNCTION; -- face_bound_reversed

FUNCTION face_reversed
  (a_face: face): face;

LOCAL
  the_reverse : face;
END_LOCAL;
IF 'CONFIG_CONTROL_DESIGN.ORIENTED_FACE' IN TYPEOF(a_face) THEN
  the_reverse := oriented_face(a_face\oriented_face.face_element,NOT
      a_face\oriented_face.orientation);
ELSE
  the_reverse := oriented_face(a_face,FALSE);
END_IF;
RETURN(the_reverse);

END_FUNCTION; -- face_reversed

FUNCTION first_proj_axis
  (z_axis, arg: direction): direction;

LOCAL
  x_vec  : vector;
  v      : direction;
  z      : direction;
  x_axis : direction;
END_LOCAL;
IF ((NOT EXISTS(z_axis)) OR (NOT EXISTS(arg))) OR (arg.dim <> 3) THEN
  x_axis := ?;
ELSE
  z_axis := normalise(z_axis);
  IF NOT EXISTS(arg) THEN
    IF z_axis <> direction([1,0,0]) THEN
      v := direction([1,0,0]);
    ELSE
      v := direction([0,1,0]);
    END_IF;
  ELSE
    IF cross_product(arg,z).magnitude = 0 THEN
      RETURN(?);
    ELSE
      v := normalise(arg);
    END_IF;
  END_IF;
```

```
  x_vec := scalar_times_vector(dot_product(v,z),z_axis);
  x_axis := vector_difference(v,x_vec).orientation;
  x_axis := normalise(x_axis);
END_IF;
RETURN(x_axis);

END_FUNCTION; -- first_proj_axis

FUNCTION gbsf_check_curve
  (cv : curve; schema_name : STRING) : BOOLEAN;

  -- check whether the input has the right type

  IF NOT (schema_name + '.CURVE' IN TYPEOF (cv)) THEN RETURN(TRUE);
  END_IF;


  -- let those types pass that do not have any further references
  -- respectively rules to be applied

  IF SIZEOF ([schema_name + '.CIRCLE', schema_name + '.ELLIPSE']
    * TYPEOF(cv)) = 1 THEN RETURN(TRUE);
  ELSE

    -- the b_spline_curve shall not self intersect

    IF (schema_name + '.B_SPLINE_CURVE' IN TYPEOF(cv))
      THEN RETURN(NOT (cv\b_spline_curve.self_intersect));
    ELSE

      -- if the curve is a composite_curve, all of its segments shall be valid



      IF schema_name + '.COMPOSITE_CURVE' IN TYPEOF(cv) THEN
        RETURN (SIZEOF (QUERY (seg <* cv\composite_curve.segments |
        NOT (gbsf_check_curve(seg.parent_curve, schema_name)))) = 0);
      ELSE

        -- the curve_replica references other curves that need to be checked

        IF schema_name + '.CURVE_REPLICA' IN TYPEOF(cv) THEN
          RETURN (gbsf_check_curve
          (cv\curve_replica.parent_curve, schema_name));
```

```
ELSE

  -- offset_curve_3d references a curve and shall not self intersect

  IF schema_name + '.OFFSET_CURVE_3D' IN TYPEOF(cv) THEN
    RETURN ((gbsf_check_curve
    (cv\offset_curve_3d.basis_curve, schema_name))
    AND
    (NOT cv\offset_curve_3d.self_intersect));
  ELSE

    -- pcurve references a curve - indirectly, and a basis_surface

    IF schema_name + '.PCURVE' IN TYPEOF(cv) THEN
      RETURN
      ((gbsf_check_curve
      (cv\pcurve.reference_to_curve\representation.items[1],
      schema_name)) AND
      (gbsf_check_surface
      (cv\pcurve.basis_surface, schema_name)));
    ELSE

      -- polyline shall have at least 3 points and shall only
      -- be used to represent an intersection_curve

      IF schema_name + '.POLYLINE' IN TYPEOF(cv) THEN
        IF (SIZEOF (cv\polyline.points) > 3)  AND
          (SIZEOF ((bag_to_set (USEDIN (cv,'')) -
          bag_to_set (USEDIN (cv,
          schema_name +
          '.INTERSECTION_CURVE.BASIS_CURVE')))) = 0)
          THEN RETURN (TRUE);
        END_IF;
      ELSE

        -- surface_curve references a curve_3d and one or two
        -- pcurves or one or two surface_curves or one of each

        IF schema_name + '.SURFACE_CURVE' IN TYPEOF(cv) THEN

          -- if the curve reference is correct, check also the rest

          IF gbsf_check_curve
          (cv\surface_curve.curve_3d, schema_name) THEN
```

```
      REPEAT i := 1 TO SIZEOF
      (cv\surface_curve.associated_geometry);

      -- do for one or two associated_geometrys:

      IF schema_name + '.SURFACE' IN TYPEOF
        (cv\surface_curve.associated_geometry[i]) THEN
        IF NOT gbsf_check_surface
          (cv\surface_curve.associated_geometry[i],
          schema_name) THEN
          RETURN(FALSE);
        END_IF;
      ELSE
        IF schema_name + '.PCURVE' IN TYPEOF
          (cv\surface_curve.associated_geometry[i]) THEN
          IF NOT gbsf_check_curve
            (cv\surface_curve.associated_geometry[i],
            schema_name) THEN
            RETURN(FALSE);
          END_IF;
        END_IF;
      END_IF;
    END_REPEAT;
    RETURN(TRUE);
  END_IF;
ELSE

  -- if the curve is a trimmed_curve

  IF (schema_name + '.TRIMMED_CURVE') IN TYPEOF (cv) THEN

  -- if a line, parabola or hyperbola is being trimmed,
  -- then valid

    IF SIZEOF ([schema_name + '.LINE',
    schema_name + '.PARABOLA',
    schema_name + '.HYPERBOLA'] *
    TYPEOF (cv\trimmed_curve.basis_curve)) = 1
    THEN RETURN (TRUE);

      -- otherwise, recursively check the basis curve

      ELSE RETURN (gbsf_check_curve
        (cv\trimmed_curve.basis_curve, schema_name));
```

```
                    END_IF;
                  END_IF;
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- gbsf_check_curve

FUNCTION gbsf_check_point (pnt : point; schema_name : STRING) : BOOLEAN;

  -- check whether the input has the right type

  IF NOT (schema_name + '.POINT' IN TYPEOF (pnt)) THEN RETURN(TRUE);
  END_IF;

  IF schema_name + '.CARTESIAN_POINT' IN TYPEOF (pnt) THEN
    RETURN(TRUE);
  ELSE

  -- a point_on_curve needs to be checked for the validity of its curve;
  -- further references down the tree are taken care of by the function
  -- gbsf_check_curve

    IF schema_name + '.POINT_ON_CURVE' IN TYPEOF (pnt) THEN
      RETURN (gbsf_check_curve
      (pnt\point_on_curve.basis_curve, schema_name));
    ELSE

    -- a point_on_surface needs to be checked for the validity of its
    -- surface further references down the tree are taken care of by
    -- the function gbsf_check_surface

      IF schema_name + '.POINT_ON_SURFACE' IN TYPEOF (pnt) THEN
        RETURN(gbsf_check_surface
        (pnt\point_on_surface.basis_surface, schema_name));
      ELSE

      -- a degenerate_pcurve needs to be checked for the validity of its
      -- defining geometry further references down the tree are taken care
```

```
      -- of by the functions gbsf_check_curve and gbsf_check_surface;
      -- both shall return true to get a valid point

        IF schema_name + '.DEGENERATE_PCURVE' IN TYPEOF (pnt) THEN
          RETURN
          ((gbsf_check_curve
          (pnt\degenerate_pcurve.reference_to_curve\representation.items[1],
          schema_name))
          AND (gbsf_check_surface
          (pnt\degenerate_pcurve.basis_surface, schema_name)));
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- gbsf_check_point

FUNCTION gbsf_check_surface (sf : surface; schema_name : STRING) : BOOLEAN;

  -- check whether the input has the right type

  IF NOT (schema_name + '.SURFACE') IN TYPEOF (sf) THEN RETURN(TRUE);
  END_IF;

  -- b_spline_surface has a self_intersect attribute that shall be false

  IF schema_name + '.B_SPLINE_SURFACE' IN TYPEOF(sf) THEN
    RETURN(NOT (sf\b_spline_surface.self_intersect));
  ELSE

    -- basis surface types return true

    IF SIZEOF ([schema_name + '.SPHERICAL_SURFACE',
      schema_name + '.TOROIDAL_SURFACE'] * TYPEOF(sf)) = 3 THEN
      RETURN(TRUE);
    ELSE

      IF schema_name + '.CURVE_BOUNDED_SURFACE' IN TYPEOF(sf) THEN

      -- if there is a simple basis_surface, check the curves

        IF SIZEOF ([schema_name + '.CONICAL_SURFACE',
          schema_name + '.CYLINDRICAL_SURFACE',
          schema_name + '.PLANE'] * TYPEOF(sf)) = 1 THEN
```

```
      RETURN(SIZEOF (QUERY (bcurve <*
      sf\curve_bounded_surface.boundaries |
      NOT (gbsf_check_curve(bcurve, schema_name)))) = 0);

    ELSE

    -- recursively check the basis_surface and then the curves

      IF gbsf_check_surface
      (sf\curve_bounded_surface.basis_surface, schema_name) THEN
        RETURN(SIZEOF (QUERY (bcurve <*
        sf\curve_bounded_surface.boundaries |
        NOT (gbsf_check_curve(bcurve, schema_name)))) = 0);
      END_IF;
    END_IF;
  ELSE

    -- offset_surface references a surface and shall not self_intersect

    IF schema_name + '.OFFSET_SURFACE' IN TYPEOF(sf) THEN
      RETURN ((gbsf_check_surface
      (sf\offset_surface.basis_surface, schema_name))
      AND
      (NOT (sf\offset_surface.self_intersect)));
    ELSE

      -- rectangular_composite_surface references a matrix of surfaces

      IF schema_name + '.RECTANGULAR_COMPOSITE_SURFACE' IN TYPEOF(sf) THEN


        REPEAT i := 1 TO SIZEOF
          (sf\rectangular_composite_surface.segments);
          REPEAT j := 1 TO SIZEOF
            (sf\rectangular_composite_surface.segments[i]);
            IF NOT (gbsf_check_surface
              (sf\rectangular_composite_surface.segments[i][j].
              \parent_surface, schema_name))
              THEN RETURN (FALSE);
            END_IF;
          END_REPEAT;
        END_REPEAT;
        RETURN(TRUE);
      ELSE
```

```
            -- rectangular_trimmed_surface has a basis_surface

            IF schema_name + '.RECTANGULAR_TRIMMED_SURFACE' IN TYPEOF(sf) THEN

              IF  SIZEOF ([schema_name + '.CONICAL_SURFACE',
                schema_name + '.CYLINDRICAL_SURFACE',
                schema_name + '.PLANE'] * TYPEOF
                (sf\rectangular_trimmed_surface.basis_surface)) = 1 THEN
                RETURN (TRUE);
              ELSE
                RETURN(gbsf_check_surface
                (sf\rectangular_trimmed_surface.basis_surface, schema_name));
              END_IF;
            ELSE

              -- parent_surfaces need to be checked for their validity

              IF schema_name + '.SURFACE_REPLICA' IN TYPEOF (sf) THEN
                RETURN (gbsf_check_surface
                (sf\surface_replica.parent_surface, schema_name));
              ELSE

                -- and the swept_surface must have a valid swept_curve

                IF schema_name + '.SWEPT_SURFACE' IN TYPEOF(sf) THEN
                  RETURN(gbsf_check_curve
                  (sf\swept_surface.swept_curve, schema_name));
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- gbsf_check_surface

FUNCTION get_basis_surface
  (c: curve_on_surface): SET [0:2] OF surface;

LOCAL
  surfs : SET [0:2] OF surface;
  n     : INTEGER;
```

```
END_LOCAL;
surfs := [];
IF 'CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(c) THEN
  surfs := [c\pcurve.basis_surface];
ELSE
  IF 'CONFIG_CONTROL_DESIGN.SURFACE_CURVE' IN TYPEOF(c) THEN
    n := SIZEOF(c\surface_curve.associated_geometry);
    REPEAT i := 1 TO n BY 1;
      surfs := surfs + associated_surface(c\surface_curve.
          associated_geometry[i]);
    END_REPEAT;
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF(c)
     THEN
  n := SIZEOF(c\composite_curve_on_surface.segments);
  surfs := get_basis_surface(c\composite_curve_on_surface.
           segments[1].parent_curve);
  IF n > 1 THEN
    REPEAT i := 2 TO n BY 1;
      surfs := surfs * get_basis_surface(c\composite_curve_on_surface.
               segments[i].parent_curve);
    END_REPEAT;
  END_IF;
END_IF;
RETURN(surfs);

END_FUNCTION; -- get_basis_surface

FUNCTION item_in_context
  (item: representation_item;
  cntxt: representation_context): BOOLEAN;

LOCAL
  i : INTEGER;
  y : BAG OF representation_item;
END_LOCAL;
IF SIZEOF(USEDIN(item,'CONFIG_CONTROL_DESIGN.REPRESENTATION.ITEMS') *
    cntxt.representations_in_context) > 0 THEN
  RETURN(TRUE);
ELSE
  y := QUERY ( z <* USEDIN(item,'') | (
      'CONFIG_CONTROL_DESIGN.REPRESENTATION_ITEM' IN TYPEOF(z)) );
  IF SIZEOF(y) > 0 THEN
```

```
    REPEAT i := 1 TO HIINDEX(y) BY 1;
      IF item_in_context(y[i],cntxt) THEN
        RETURN(TRUE);
      END_IF;
    END_REPEAT;
  END_IF;
END_IF;
RETURN(FALSE);

END_FUNCTION; -- item_in_context

FUNCTION leap_year
  (year: year_number): BOOLEAN;
IF (((year MOD 4) = 0) AND ((year MOD 100) <> 0)) OR ((year MOD 400) =
    0) THEN
  RETURN(TRUE);
ELSE
  RETURN(FALSE);
END_IF;

END_FUNCTION; -- leap_year

FUNCTION list_face_loops
  (f: face): LIST [0:?] OF loop;

LOCAL
  loops : LIST [0:?] OF loop := [];
END_LOCAL;
REPEAT i := 1 TO SIZEOF(f.bounds) BY 1;
  loops := loops + f.bounds[i].bound;
END_REPEAT;
RETURN(loops);

END_FUNCTION; -- list_face_loops

FUNCTION list_of_topology_reversed
  (a_list: list_of_reversible_topology_item
  ): list_of_reversible_topology_item;

LOCAL
  the_reverse : list_of_reversible_topology_item;
END_LOCAL;
the_reverse := [];
REPEAT i := 1 TO SIZEOF(a_list) BY 1;
```

```
  the_reverse := topology_reversed(a_list[i]) + the_reverse;
END_REPEAT;
RETURN(the_reverse);

END_FUNCTION; -- list_of_topology_reversed

FUNCTION list_to_array
  (lis: LIST [0:?] OF GENERIC:t;
  low, u: INTEGER
  ): ARRAY [low:u] OF GENERIC:t;

LOCAL
  n   : INTEGER;
  res : ARRAY [low:u] OF GENERIC:t;
END_LOCAL;
n := SIZEOF(lis);
IF n <> ((u - low) + 1) THEN
  RETURN(?);
ELSE
  REPEAT i := 1 TO n BY 1;
    res[(low + i) - 1] := lis[i];
  END_REPEAT;
  RETURN(res);
END_IF;

END_FUNCTION; -- list_to_array

FUNCTION list_to_set
  (l: LIST [0:?] OF GENERIC:t): SET OF GENERIC:t;

LOCAL
  s : SET OF GENERIC:t := [];
END_LOCAL;
REPEAT i := 1 TO SIZEOF(l) BY 1;
  s := s + l[i];
END_REPEAT;
RETURN(s);

END_FUNCTION; -- list_to_set

FUNCTION local_relatives_of_product_definitions
   (definition_set: SET OF product_definition;
   total_definitions: SET OF product_definition;
   relation_subtype: STRING): SET OF product_definition;
```

```
LOCAL
  i           : INTEGER;
  local_def   : SET OF product_definition := [];
  local_pdr   : SET OF product_definition_relationship := [];
  local_total : SET OF product_definition := [];
END_LOCAL;
REPEAT i := 1 TO HIINDEX(definition_set) BY 1;
  local_pdr := local_pdr + bag_to_set(USEDIN(definition_set[i],
      relation_subtype + '.RELATING_PRODUCT_DEFINITION'));
END_REPEAT;
REPEAT i := 1 TO HIINDEX(local_pdr) BY 1;
  local_def := local_def + local_pdr[i].related_product_definition;
END_REPEAT;
IF (SIZEOF(local_def) - SIZEOF(total_definitions)) = 0 THEN
  RETURN(local_def);
ELSE
  local_total := total_definitions + local_def;
  RETURN(local_def + local_relatives_of_product_definitions(local_def
      - total_definitions,local_total,relation_subtype));
END_IF;

END_FUNCTION; -- local_relatives_of_product_definitions

FUNCTION make_array_of_array
  (lis: LIST [1:?] OF LIST [1:?] OF GENERIC:t;
  low1, u1, low2, u2: INTEGER
  ): ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;

LOCAL
  n2   : INTEGER;
  n1   : INTEGER;
  res  : ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;
  res1 : LIST [1:?] OF ARRAY [low2:u2] OF GENERIC:t;
END_LOCAL;
n1 := SIZEOF(lis);
n2 := SIZEOF(lis[1]);
IF (n1 <> ((u1 - low1) + 1)) AND (n2 <> ((u2 - low2) + 1)) THEN
  RETURN(?);
END_IF;
REPEAT i := 1 TO n1 BY 1;
  IF SIZEOF(lis[i]) <> n2 THEN
    RETURN(?);
  END_IF;
END_REPEAT;
```

```
REPEAT i := 1 TO n1 BY 1;
  resl[i] := list_to_array(lis[i],low2,u2);
END_REPEAT;
res := list_to_array(resl,low1,u1);
RETURN(res);

END_FUNCTION; -- make_array_of_array

FUNCTION mixed_loop_type_set
  (l: SET [0:?] OF loop): LOGICAL;

LOCAL
  i              : INTEGER;
  poly_loop_type : LOGICAL;
END_LOCAL;
IF SIZEOF(l) <= 1 THEN
  RETURN(FALSE);
END_IF;
poly_loop_type := 'CONFIG_CONTROL_DESIGN.POLY_LOOP' IN TYPEOF(l[1]);
REPEAT i := 2 TO SIZEOF(l) BY 1;
  IF ('CONFIG_CONTROL_DESIGN.POLY_LOOP' IN TYPEOF(l[i])) <>
      poly_loop_type THEN
    RETURN(TRUE);
  END_IF;
END_REPEAT;
RETURN(FALSE);

END_FUNCTION; -- mixed_loop_type_set

FUNCTION msb_shells
  (brep: manifold_solid_brep;
  schema_name: STRING): SET [1:?] OF closed_shell;
IF (schema_name + '.BREP_WITH_VOIDS') IN TYPEOF(brep) THEN
  RETURN(brep\brep_with_voids.voids + brep.outer);
ELSE
  RETURN([brep.outer]);
END_IF;

END_FUNCTION; -- msb_shells

FUNCTION normalise
  (arg: vector_or_direction): vector_or_direction;

LOCAL
```

```
  ndim   : INTEGER;
  v      : direction;
  vec    : vector;
  mag    : REAL;
  result : vector_or_direction;
END_LOCAL;
IF NOT EXISTS(arg) THEN
  result := ?;
ELSE
  ndim := arg.dim;
  IF 'CONFIG_CONTROL_DESIGN.VECTOR' IN TYPEOF(arg) THEN
    BEGIN
      vec := arg;
      v := arg.orientation;
      IF arg.magnitude = 0 THEN
        RETURN(?);
      ELSE
        vec.magnitude := 1;
      END_IF;
    END;
  ELSE
    v := arg;
  END_IF;
  mag := 0;
  REPEAT i := 1 TO ndim BY 1;
    mag := mag + (v.direction_ratios[i] * v.direction_ratios[i]);
  END_REPEAT;
  IF mag > 0 THEN
    mag := SQRT(mag);
    REPEAT i := 1 TO ndim BY 1;
      v.direction_ratios[i] := v.direction_ratios[i] / mag;
    END_REPEAT;
    IF 'CONFIG_CONTROL_DESIGN.VECTOR' IN TYPEOF(arg) THEN
      vec.orientation := v;
      result := vec;
    ELSE
      result := v;
    END_IF;
  ELSE
    RETURN(?);
  END_IF;
END_IF;
RETURN(result);
```

```
END_FUNCTION; -- normalise

FUNCTION orthogonal_complement
  (vec: direction): direction;

LOCAL
  result : direction;
END_LOCAL;
IF (vec.dim <> 2) OR (NOT EXISTS(vec)) THEN
  RETURN(?);
ELSE
  result.direction_ratios[1] := -vec.direction_ratios[2];
  result.direction_ratios[2] := vec.direction_ratios[1];
  RETURN(result);
END_IF;

END_FUNCTION; -- orthogonal_complement

FUNCTION path_head_to_tail
  (a_path: path): LOGICAL;

LOCAL
  n : INTEGER;
  p : LOGICAL := TRUE;
END_LOCAL;
n := SIZEOF(a_path.edge_list);
REPEAT i := 2 TO n BY 1;
  p := p AND (a_path.edge_list[i - 1].edge_end :=: a_path.edge_list[i]
      .edge_start);
END_REPEAT;
RETURN(p);

END_FUNCTION; -- path_head_to_tail

FUNCTION path_reversed
  (a_path: path): path;

LOCAL
  the_reverse : path;
END_LOCAL;
IF 'CONFIG_CONTROL_DESIGN.ORIENTED_PATH' IN TYPEOF(a_path) THEN
  the_reverse := oriented_path(a_path\oriented_path.path_element,NOT
      a_path\oriented_path.orientation);
ELSE
```

```
  the_reverse := oriented_path(a_path,FALSE);
END_IF;
RETURN(the_reverse);

END_FUNCTION; -- path_reversed

FUNCTION relatives_of_product_definitions
  (definition_set: SET OF product_definition;
  relation_subtype: STRING): SET OF product_definition;

FUNCTION local_relatives_of_product_definitions
  (definition_set: SET OF product_definition;
  total_definitions: SET OF product_definition;
  relation_subtype: STRING): SET OF product_definition;

  LOCAL
    i           : INTEGER;
    local_def   : SET OF product_definition := [];
    local_pdr   : SET OF product_definition_relationship := [];
    local_total : SET OF product_definition := [];
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(definition_set) BY 1;
    local_pdr := local_pdr + bag_to_set(USEDIN(definition_set[i],
        relation_subtype + '.RELATING_PRODUCT_DEFINITION'));
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(local_pdr) BY 1;
    local_def := local_def + local_pdr[i].related_product_definition;
  END_REPEAT;
  IF (SIZEOF(local_def) - SIZEOF(total_definitions)) = 0 THEN
    RETURN(local_def);
  ELSE
    local_total := total_definitions + local_def;
    RETURN(local_def + local_relatives_of_product_definitions(
        local_def - total_definitions,local_total,relation_subtype));
  END_IF;

END_FUNCTION; -- local_relatives_of_product_definitions
RETURN(local_relatives_of_product_definitions(definition_set,
    definition_set,relation_subtype));

END_FUNCTION; -- relatives_of_product_definitions

FUNCTION scalar_times_vector
  (scalar: REAL;
```

```
  vec: vector_or_direction): vector;

LOCAL
  v      : direction;
  mag    : REAL;
  result : vector;
END_LOCAL;
IF (NOT EXISTS(scalar)) OR (NOT EXISTS(vec)) THEN
  result := ?;
ELSE
  IF 'CONFIG_CONTROL_DESIGN.VECTOR' IN TYPEOF(vec) THEN
    v := vec.orientation;
    mag := scalar * vec.magnitude;
  ELSE
    v := vec;
    mag := scalar;
  END_IF;
  IF mag < 0 THEN
    REPEAT i := 1 TO SIZEOF(v.direction_ratios) BY 1;
      v.direction_ratios[i] := -v.direction_ratios[i];
    END_REPEAT;
    mag := -mag;
  END_IF;
  result.orientation := normalise(v);
  result.magnitude := mag;
END_IF;
RETURN(result);

END_FUNCTION; -- scalar_times_vector

FUNCTION second_proj_axis
  (z_axis, x_axis, arg: direction): direction;

LOCAL
  temp   : vector;
  v      : direction;
  y_axis : vector;
END_LOCAL;
IF NOT EXISTS(arg) THEN
  v := direction([0,1,0]);
ELSE
  v := arg;
END_IF;
temp := scalar_times_vector(dot_product(v,z_axis),z_axis);
```

```
y_axis := vector_difference(v,temp);
temp := scalar_times_vector(dot_product(v,x_axis),x_axis);
y_axis := vector_difference(y_axis,temp);
y_axis := normalise(y_axis);
RETURN(y_axis.orientation);

END_FUNCTION; -- second_proj_axis

FUNCTION set_of_topology_reversed
  (a_set: set_of_reversible_topology_item) :
  set_of_reversible_topology_item;

LOCAL
  the_reverse : set_of_reversible_topology_item;
END_LOCAL;
the_reverse := [];
REPEAT i := 1 TO SIZEOF(a_set) BY 1;
  the_reverse := the_reverse + topology_reversed(a_set[i]);
END_REPEAT;
RETURN(the_reverse);

END_FUNCTION; -- set_of_topology_reversed

FUNCTION shell_reversed
  (a_shell: shell): shell;

LOCAL
  the_reverse : shell;
END_LOCAL;
IF 'CONFIG_CONTROL_DESIGN.ORIENTED_OPEN_SHELL' IN TYPEOF(a_shell)
     THEN
  the_reverse := oriented_open_shell(a_shell\oriented_open_shell.
      open_shell_element,NOT a_shell\oriented_open_shell.orientation);
ELSE
  IF 'CONFIG_CONTROL_DESIGN.OPEN_SHELL' IN TYPEOF(a_shell) THEN
    the_reverse := oriented_open_shell(a_shell,FALSE);
  ELSE
    IF 'CONFIG_CONTROL_DESIGN.ORIENTED_CLOSED_SHELL' IN TYPEOF(a_shell)
         THEN
      the_reverse := oriented_closed_shell(a_shell\
          oriented_closed_shell.closed_shell_element,NOT a_shell\
          oriented_closed_shell.orientation);
    ELSE
      IF 'CONFIG_CONTROL_DESIGN.CLOSED_SHELL' IN TYPEOF(a_shell) THEN
```

```
        the_reverse := oriented_closed_shell(a_shell,FALSE);
      ELSE
        the_reverse := ?;
      END_IF;
    END_IF;
  END_IF;
END_IF;
RETURN(the_reverse);

END_FUNCTION; -- shell_reversed

FUNCTION surface_weights_positive
  (b: rational_b_spline_surface): BOOLEAN;

LOCAL
  result : BOOLEAN := TRUE;
END_LOCAL;
REPEAT i := 0 TO b.u_upper BY 1;
  REPEAT j := 0 TO b.v_upper BY 1;
    IF b.weights[i][j] <= 0 THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
END_REPEAT;
RETURN(result);

END_FUNCTION; -- surface_weights_positive

FUNCTION topology_reversed
  (an_item: reversible_topology): reversible_topology;
IF 'CONFIG_CONTROL_DESIGN.EDGE' IN TYPEOF(an_item) THEN
  RETURN(edge_reversed(an_item));
END_IF;
IF 'CONFIG_CONTROL_DESIGN.PATH' IN TYPEOF(an_item) THEN
  RETURN(path_reversed(an_item));
END_IF;
IF 'CONFIG_CONTROL_DESIGN.FACE_BOUND' IN TYPEOF(an_item) THEN
  RETURN(face_bound_reversed(an_item));
END_IF;
IF 'CONFIG_CONTROL_DESIGN.FACE' IN TYPEOF(an_item) THEN
  RETURN(face_reversed(an_item));
END_IF;
IF 'CONFIG_CONTROL_DESIGN.SHELL' IN TYPEOF(an_item) THEN
```

```
  RETURN(shell_reversed(an_item));
END_IF;
IF 'SET' IN TYPEOF(an_item) THEN
  RETURN(set_of_topology_reversed(an_item));
END_IF;
IF 'LIST' IN TYPEOF(an_item) THEN
  RETURN(list_of_topology_reversed(an_item));
END_IF;
RETURN(?);

END_FUNCTION; -- topology_reversed

FUNCTION unique_version_change_order (c : action) : BOOLEAN;
  LOCAL
    ords       : action_directive := c\directed_action.directive;
    assign     : SET OF change_request := [];
    versions   : SET OF product_definition_formation := [];
  END_LOCAL;

  -- build the set of change_requests that are the assigned
  -- versioned_action_requests incorporated by the action_directive
  REPEAT i := 1 TO SIZEOF(ords.requests);
  assign := assign + QUERY (ara <* bag_to_set (USEDIN (ords.requests[i],
            'CONFIG_CONTROL_DESIGN.ACTION_REQUEST_ASSIGNMENT.' +
            'ASSIGNED_ACTION_REQUEST')) |
            'CONFIG_CONTROL_DESIGN.CHANGE_REQUEST' IN TYPEOF (ara));
  END_REPEAT;

  -- gather the product_definition_formations that are referenced by the
  -- change_requests

  REPEAT k := 1 TO SIZEOF(assign);
    versions := versions + assign[k].items;
  END_REPEAT;

  -- check that no product_definition_formation reference the same
  -- instance of product

  RETURN (SIZEOF (QUERY (vers <* versions |
          NOT (SIZEOF (QUERY (other_vers <* versions - vers |
          vers.of_product :=: other_vers.of_product)) = 0))) = 0);
END_FUNCTION; -- unique_version_change_order

FUNCTION using_representations
```

```
(item : representation_item)
: SET OF representation;
LOCAL
  results             : SET OF representation;
  result_bag          : BAG OF representation;
  intermediate_items : SET OF representation_item;
  i                   : INTEGER;
END_LOCAL;

-- Find the representations in which the item is used and add to the
-- results set.

result_bag := USEDIN(item,'CONFIG_CONTROL_DESIGN.REPRESENTATION.ITEMS');

IF SIZEOF(result_bag) > 0 THEN
  REPEAT i := 1 TO HIINDEX(result_bag);
    results := results + result_bag[i];
  END_REPEAT;
END_IF;

-- Find the set of representation_items in which item is used.

intermediate_items := QUERY(z <* bag_to_set( USEDIN(item , '')) |
  'CONFIG_CONTROL_DESIGN.REPRESENTATION_ITEM' IN TYPEOF(z));

-- If the set of intermediate items is not empty;

IF SIZEOF(intermediate_items) > 0 THEN

  -- For each element in the set, recursively add the
  -- using_representations of that element.

  REPEAT i := 1 TO HIINDEX(intermediate_items);
    results := results + using_representations(intermediate_items[i]);
  END_REPEAT;
END_IF;

-- Return the set of representation in which the input item is
-- used directly and indirectly (through intervening
-- representation_items).

RETURN (results);
END_FUNCTION; -- using_representations
```

```
FUNCTION valid_calendar_date
  (date: calendar_date): LOGICAL;
IF NOT ((1 <= date.day_component) AND (date.day_component <= 31))
     THEN
  RETURN(FALSE);
END_IF;
CASE date.month_component OF
  4 :          RETURN((1 <= date.day_component) AND (date.day_component
       <= 30));
  6 :          RETURN((1 <= date.day_component) AND (date.day_component
       <= 30));
  9 :          RETURN((1 <= date.day_component) AND (date.day_component
       <= 30));
  11  :          RETURN((1 <= date.day_component) AND (date.
       day_component <= 30));
  2 :          BEGIN
      IF leap_year(date.year_component) THEN
        RETURN((1 <= date.day_component) AND (date.day_component <= 29));
      ELSE
        RETURN((1 <= date.day_component) AND (date.day_component <= 28));
      END_IF;
    END;
  END_CASE;

END_FUNCTION; -- valid_calendar_date

FUNCTION valid_geometrically_bounded_wf_curve
  (crv   : curve; schma : STRING) : BOOLEAN;

  -- check for valid basic curves

  IF SIZEOF ([schma + '.POLYLINE',
    schma + '.B_SPLINE_CURVE',
    schma + '.ELLIPSE',
    schma + '.CIRCLE'] * TYPEOF (crv)) = 1
    THEN RETURN (TRUE);
  ELSE

    -- if the curve is a trimmed_curve

    IF (schma + '.TRIMMED_CURVE') IN TYPEOF (crv) THEN

      -- if a line, parabola or hyperbola is being trimmed, then valid
```

```
      IF SIZEOF ([schma + '.LINE',
        schma + '.PARABOLA',
        schma + '.HYPERBOLA'] * TYPEOF (crv\trimmed_curve.basis_curve)) = 1
        THEN RETURN (TRUE);

        -- otherwise, recursively check the basis_curve

        ELSE RETURN (valid_geometrically_bounded_wf_curve
        (crv\trimmed_curve.basis_curve, schma));
      END_IF;
    ELSE

      -- recursively check the offset_curves basis curve

      IF (schma + '.OFFSET_CURVE_3D') IN TYPEOF (crv)
        THEN RETURN (valid_geometrically_bounded_wf_curve
        (crv\offset_curve_3d.basis_curve, schma));
      ELSE

        -- recursively check the curve_replicas parent curve

        IF (schma + '.CURVE_REPLICA') IN TYPEOF (crv) THEN
          RETURN (valid_geometrically_bounded_wf_curve
          (crv\curve_replica.parent_curve, schma));
        ELSE

          -- recursively check the composite_curves segments

          IF (schma + '.COMPOSITE_CURVE') IN TYPEOF (crv) THEN
            RETURN (SIZEOF (QUERY (ccs <* crv\composite_curve.segments |
            NOT (valid_geometrically_bounded_wf_curve
            (ccs.parent_curve, schma))))
            = 0);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION; -- valid_geometrically_bounded_wf_curve

FUNCTION valid_geometrically_bounded_wf_point
  (pnt : point; schma: STRING) : BOOLEAN;
```

--check for valid basis type

```
  IF (schma + '.CARTESIAN_POINT') IN TYPEOF (pnt)
   THEN RETURN (TRUE);
  ELSE

    -- if the input type is a point_on_curve then check for a valid
    -- geometrically bounded curve type as the basis

    IF (schma + '.POINT_ON_CURVE') IN TYPEOF (pnt)
      THEN RETURN (valid_geometrically_bounded_wf_curve
      (pnt\point_on_curve.basis_curve, schma));
    ELSE

      -- if the input type is a point_replica check for a valid parent point

      IF (schma + '.POINT_REPLICA') IN TYPEOF (pnt)
        THEN RETURN (valid_geometrically_bounded_wf_point
        (pnt\point_replica.parent_pt, schma));
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION; -- valid_geometrically_bounded_wf_point

FUNCTION valid_time
  (time: local_time): BOOLEAN;
IF EXISTS(time.second_component) THEN
  RETURN(EXISTS(time.minute_component));
ELSE
  RETURN(TRUE);
END_IF;

END_FUNCTION; -- valid_time

FUNCTION valid_units
  (m: measure_with_unit): BOOLEAN;
IF 'CONFIG_CONTROL_DESIGN.LENGTH_MEASURE' IN TYPEOF(m.value_component)
     THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(1,0,0,0,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
```

```
IF 'CONFIG_CONTROL_DESIGN.MASS_MEASURE' IN TYPEOF(m.value_component)
     THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,1,0,0,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.TIME_MEASURE' IN TYPEOF(m.value_component)
     THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,0,1,0,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.ELECTRIC_CURRENT_MEASURE' IN TYPEOF(m.
    value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,0,0,1,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.THERMODYNAMIC_TEMPERATURE_MEASURE' IN
    TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,0,0,0,1,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.AMOUNT_OF_SUBSTANCE_MEASURE' IN TYPEOF(m.
    value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,0,0,0,0,1,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.LUMINOUS_INTENSITY_MEASURE' IN TYPEOF(m.
    value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,0,0,0,0,0,1) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.PLANE_ANGLE_MEASURE' IN TYPEOF(m.
    value_component) THEN
```

```
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,0,0,0,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.SOLID_ANGLE_MEASURE' IN TYPEOF(m.
    value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,0,0,0,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.AREA_MEASURE' IN TYPEOF(m.value_component)
     THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(2,0,0,0,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.VOLUME_MEASURE' IN TYPEOF(m.value_component)
     THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(3,0,0,0,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.RATIO_MEASURE' IN TYPEOF(m.value_component)
     THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,0,0,0,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.POSITIVE_LENGTH_MEASURE' IN TYPEOF(m.
    value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(1,0,0,0,0,0,0) THEN
    RETURN(FALSE);
  END_IF;
END_IF;
IF 'CONFIG_CONTROL_DESIGN.POSITIVE_PLANE_ANGLE_MEASURE' IN TYPEOF(m.
    value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0,0,0,0,0,0,0) THEN
```

```
    RETURN(FALSE);
  END_IF;
END_IF;
RETURN(TRUE);

END_FUNCTION; -- valid_units

FUNCTION valid_wireframe_edge_curve
  (crv   : curve; schma : STRING) : BOOLEAN;

  -- check for valid basic curve types

  IF SIZEOF ([schma + '.LINE',
    schma + '.CIRCLE',
    schma + '.ELLIPSE',
    schma + '.PARABOLA',
    schma + '.HYPERBOLA',
    schma + '.B_SPLINE_CURVE',
    schma + '.POLYLINE'] * TYPEOF (crv)) = 1
    THEN RETURN (TRUE);
  ELSE

   -- recursively check for valid parent curves for curve_replica

    IF (schma + '.CURVE_REPLICA') IN TYPEOF(crv)
      THEN RETURN
      (valid_wireframe_edge_curve (crv\curve_replica.parent_curve, schma));
    ELSE

      -- recursively check for valid basis curves for offset_curve

      IF (schma + '.OFFSET_CURVE_3D') IN TYPEOF (crv)
        THEN RETURN
        (valid_wireframe_edge_curve (crv\offset_curve_3d.basis_curve, schma));
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION; -- valid_wireframe_edge_curve

FUNCTION valid_wireframe_vertex_point
  (pnt   : point; schma : STRING) : BOOLEAN;

  -- check for valid basic point type
```

```
  IF (schma + '.CARTESIAN_POINT' IN TYPEOF (pnt))
    THEN RETURN (TRUE);
  ELSE

    -- recursively check for valid basic point type as parents for a
    -- point_replica

    IF (schma + '.POINT_REPLICA') IN TYPEOF (pnt)
      THEN RETURN (valid_wireframe_vertex_point
      (pnt\point_replica.parent_pt, schma));
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION; -- valid_wireframe_vertex_point

FUNCTION vector_difference
  (arg1, arg2: vector_or_direction): vector;

LOCAL
  ndim   : INTEGER;
  mag2   : REAL;
  mag1   : REAL;
  mag    : REAL;
  res    : direction;
  vec1   : direction;
  vec2   : direction;
  result : vector;
END_LOCAL;
IF ((NOT EXISTS(arg1)) OR (NOT EXISTS(arg2))) OR (arg1.dim <> arg2.dim)
     THEN
  result := ?;
ELSE
  BEGIN
    IF 'CONFIG_CONTROL_DESIGN.VECTOR' IN TYPEOF(arg1) THEN
      mag1 := arg1.magnitude;
      vec1 := arg1.orientation;
    ELSE
      mag1 := 1;
      vec1 := arg1;
    END_IF;
    IF 'CONFIG_CONTROL_DESIGN.VECTOR' IN TYPEOF(arg2) THEN
      mag2 := arg2.magnitude;
      vec2 := arg2.orientation;
    ELSE
```

```
      mag2 := 1;
      vec2 := arg2;
    END_IF;
    vec1 := normalise(vec1);
    vec2 := normalise(vec2);
    ndim := SIZEOF(vec1.direction_ratios);
    mag := 0;
    REPEAT i := 1 TO ndim BY 1;
      res.direction_ratios[i] := (mag1 * vec1.direction_ratios[i]) - (
          mag2 * vec2.direction_ratios[i]);
      mag := mag + (res.direction_ratios[i] * res.direction_ratios[i]);
    END_REPEAT;
    IF mag > 0 THEN
      result.magnitude := SQRT(mag);
      result.orientation := res;
    ELSE
      result.magnitude := 0;
      result.orientation := vec1;
    END_IF;
  END;
END_IF;
RETURN(result);

END_FUNCTION; -- vector_difference

END_SCHEMA; -- config_control_design
(*
```

附 录 B
（标准的附录）
实体的应用解释模型短名

本附录为本标准出现的每个实体名称都提供了唯一的指定短名。该实体长名从EXPRESS语言的陈述中提取，其情况见附录A。

表B1提供了本标准的应用解释模型中规定的实体短名。关于该短名使用方面的要求见GB/T 16656的各实施方法。

本标准还包括一块1.44MBDOS格式化的磁盘，该磁盘上以数字形式提供了本附录的内容。该文件名为ANNEX-B。

该数字文件是ASCII码形式的文件，并含有短名，其短名后接逗号“,”，逗号后接该实体名，最后由回车结束。该短名在长度上不超过六个字符。该实体、类型或功能名是一个可变长度的字符串。

表B1 应用解释模型的实体短名

长 名	短 名
ACTION	ACTION
ACTION_ASSIGNMENT	ACTASS
ACTION_DIRECTIVE	ACTDRC
ACTION_METHOD	ACTMTH
ACTION_REQUEST_ASSIGNMENT	ACRQAS
ACTION_REQUEST_SOLUTION	ACRQSL
ACTION_REQUEST_STATUS	ACRQST
ACTION_STATUS	ACTSTT
ADDRESS	ADDRSS
ADVANCED_BREP_SHAPE_REPRESENTATION	ABSR
ADVANCED_FACE	ADVFC
ALTERNATE_PRODUCT_RELATIONSHIP	ALPRRL
APPLICATION_CONTEXT	APPCNT
APPLICATION_CONTEXT_ELEMENT	APCNEL
APPLICATION_PROTOCOL_DEFINITION	APPRDF
APPROVAL	APPRVL
APPROVAL_ASSIGNMENT	APPASS
APPROVAL_DATE_TIME	APDTTM
APPROVAL_PERSON_ORGANIZATION	APPROR
APPROVAL_RELATIONSHIP	APPRLT
APPROVAL_ROLE	APPRL
APPROVAL_STATUS	APPSTT
AREA_MEASURE_WITH_UNIT	AMWU

表 B1(续)

长　　名	短　名
AREA_UNIT	ARUNT
ASSEMBLY_COMPONENT_USAGE	ASCMUS
ASSEMBLY_COMPONENT_USAGE_SUBSTITUTE	ACUS
AXIS1_PLACEMENT	AX1PLC
AXIS2_PLACEMENT_2D	A2PL2D
AXIS2_PLACEMENT_3D	A2PL3D
BEZIER_CURVE	BZRCRV
BEZIER_SURFACE	BZRSRF
BOUNDARY_CURVE	BNDCR
BOUNDED_CURVE	BNDCRV
BOUNDED_SURFACE	BNDSRF
BREP_WITH_VOIDS	BRWTVD
B_SPLINE_CURVE	BSPCR
B_SPLINE_CURVE_WITH_KNOTS	BSCWK
B_SPLINE_SURFACE	BSPSR
B_SPLINE_SURFACE_WITH_KNOTS	BSSWK
CALENDAR_DATE	CLNDT
CARTESIAN_POINT	CRTPNT
CARTESIAN_TRANSFORMATION_OPERATOR	CRTROP
CARTESIAN_TRANSFORMATION_OPERATOR_2D	CTO2
CARTESIAN_TRANSFORMATION_OPERATOR_3D	CTO3
CC_DESIGN_APPROVAL	CCDSAP
CC_DESIGN_CERTIFICATION	CCDSCR
CC_DESIGN_CONTRACT	CCDSCN
CC_DESIGN_DATE_AND_TIME_ASSIGNMENT	CDDATA
CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT	CDPAOA
CC_DESIGN_SECURITY_CLASSIFICATION	CDSC
CC_DESIGN_SPECIFICATION_REFERENCE	CDS
CERTIFICATION	CRTFCT
CERTIFICATION_ASSIGNMENT	CRTASS
CERTIFICATION_TYPE	CRTTYP
CHANGE	CHANGE
CHANGE_REQUEST	CHNRQS
CIRCLE	CIRCLE

表 B1(续)

长　　名	短　名
CLOSED_SHELL	CLSSHL
COMPOSITE_CURVE	CMPCRV
COMPOSITE_CURVE_ON_SURFACE	CCOS
COMPOSITE_CURVE_SEGMENT	CMCRSG
CONFIGURATION_DESIGN	CNFDSG
CONFIGURATION_EFFECTIVITY	CNFEFF
CONFIGURATION_ITEM	CNFITM
CONIC	CONIC
CONICAL_SURFACE	CNCSRF
CONNECTED_EDGE_SET	CNEDST
CONNECTED_FACE_SET	CNFCST
CONTEXT_DEPENDENT_SHAPE_REPRESENTATION	CDSR
CONTEXT_DEPENDENT_UNIT	CNDPUN
CONTRACT	CNTRCT
CONTRACT_ASSIGNMENT	CNTASS
CONTRACT_TYPE	CNTTYP
CONVERSION_BASED_UNIT	CNBSUN
COORDINATED_UNIVERSAL_TIME_OFFSET	CUTO
CURVE	CURVE
CURVE_BOUNDED_SURFACE	CRBNSR
CURVE_REPLICA	CRVRPL
CYLINDRICAL_SURFACE	CYLSRF
DATE	DATE
DATED_EFFECTIVITY	DTDEFF
DATE_AND_TIME	DTANTM
DATE_AND_TIME_ASSIGNMENT	DATA
DATE_TIME_ROLE	DTTMRL
DEFINITIONAL_REPRESENTATION	DFNRPR
DEGENERATE_PCURVE	DGNPCR
DESIGN_CONTEXT	DSGCNT
DESIGN_MAKE_FROM_RELATIONSHIP	DMFR
DIMENSIONAL_EXPONENTS	DMNEXP
DIRECTED_ACTION	DRCACT
DIRECTION	DRCTN

表 B1(续)

长　　名	短　　名
DOCUMENT	DCMNT
DOCUMENT_REFERENCE	DCMRFR
DOCUMENT_RELATIONSHIP	DCMRLT
DOCUMENT_TYPE	DCMTYP
DOCUMENT_USAGE_CONSTRAINT	DCUSCN
DOCUMENT_WITH_CLASS	DCWTCL
EDGE	EDGE
EDGE_BASED_WIREFRAME_MODEL	EBWM
EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION	EBWSR
EDGE_CURVE	EDGCRV
EDGE_LOOP	EDGLP
EFFECTIVITY	EFFCTV
ELEMENTARY_SURFACE	ELMSRF
ELLIPSE	ELLPS
EVALUATED_DEGENERATE_PCURVE	EVDGPC
EXECUTED_ACTION	EXCACT
FACE	FACE
FACETED_BREP	FCTBR
FACETED_BREP_SHAPE_REPRESENTATION	FBSR
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSRF
FUNCTIONALLY_DEFINED_TRANSFORMATION	FNDFTR
GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION	GBSSR
GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION	GBWSR
GEOMETRIC_CURVE_SET	GMCRST
GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
GEOMETRIC_SET	GMTST
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT	GC
GLOBAL_UNIT_ASSIGNED_CONTEXT	GUAC
HYPERBOLA	HYPRBL
INTERSECTION_CURVE	INTCRV
ITEM_DEFINED_TRANSFORMATION	ITDFTR

表 B1(续)

长　　名	短　　名
LENGTH_MEASURE_WITH_UNIT	LMWU
LENGTH_UNIT	LNGUNT
LINE	LINE
LOCAL_TIME	LCLTM
LOOP	LOOP
LOT_EFFECTIVITY	LTEFF
MANIFOLD_SOLID_BREP	MNSLBR
MANIFOLD_SURFACE_SHAPE_REPRESENTATION	MSSR
MAPPED_ITEM	MPPITM
MASS_MEASURE_WITH_UNIT	MMWU
MASS_UNIT	MSSUNT
MEASURE_WITH_UNIT	MSWTUN
MECHANICAL_CONTEXT	MCHCNT
NAMED_UNIT	NMDUNT
NEXT_ASSEMBLY_USAGE_OCCURRENCE	NAUO
OFFSET_CURVE_2D	OFCR2D
OFFSET_CURVE_3D	OFCR3D
OFFSET_SURFACE	OFFSRF
OPEN_SHELL	OPNSHL
ORDINAL_DATE	ORDDT
ORGANIZATION	ORGNZT
ORGANIZATIONAL_ADDRESS	ORGADD
ORGANIZATIONAL_PROJECT	ORGPRJ
ORGANIZATION_RELATIONSHIP	ORGRLT
ORIENTED_CLOSED_SHELL	ORCLSH
ORIENTED_EDGE	ORNEDG
ORIENTED_FACE	ORNFC
ORIENTED_OPEN_SHELL	OROPSH
ORIENTED_PATH	ORNPTH
OUTER_BOUNDARY_CURVE	OTBNCR
PARABOLA	PRBL
PARAMETRIC_REPRESENTATION_CONTEXT	PRRPCN
PATH	PATH
PCURVE	PCURVE

表 B1(续)

长　　名	短　　名
PERSON	PERSON
PERSONAL_ADDRESS	PRSADD
PERSON_AND_ORGANIZATION	PRANOR
PERSON_AND_ORGANIZATION_ASSIGNMENT	PAOA
PERSON_AND_ORGANIZATION_ROLE	PAOR
PLACEMENT	PLCMNT
PLANE	PLANE
PLANE_ANGLE_MEASURE_WITH_UNIT	PAMWU
PLANE_ANGLE_UNIT	PLANUN
POINT	POINT
POINT_ON_CURVE	PNONCR
POINT_ON_SURFACE	PNONSR
POINT_REPLICA	PNTRPL
POLYLINE	PLYLN
POLY_LOOP	PLYLP
PRODUCT	PRDCT
PRODUCT_CATEGORY	PRDCTG
PRODUCT_CATEGORY_RELATIONSHIP	PRCTRL
PRODUCT_CONCEPT	PRDCNC
PRODUCT_CONCEPT_CONTEXT	PRCNCN
PRODUCT_CONTEXT	PRDCNT
PRODUCT_DEFINITION	PRDDFN
PRODUCT_DEFINITION_CONTEXT	PRDFCN
PRODUCT_DEFINITION_EFFECTIVITY	PRDFEF
PRODUCT_DEFINITION_FORMATION	PRDFFR
PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE	PDFWSS
PRODUCT_DEFINITION_RELATIONSHIP	PRDFRL
PRODUCT_DEFINITION_SHAPE	PRDFSH
PRODUCT_DEFINITION_USAGE	PRDFUS
PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS	PDWAD
PRODUCT_RELATED_PRODUCT_CATEGORY	PRPC
PROMISSORY_USAGE_OCCURRENCE	PRUSOC
PROPERTY_DEFINITION	PRPDFN
PROPERTY_DEFINITION_REPRESENTATION	PRDFRP

表 B1(续)

长　　名	短　　名
QUANTIFIED_ASSEMBLY_COMPONENT_USAGE	QACU
QUASI_UNIFORM_CURVE	QSUNCR
QUASI_UNIFORM_SURFACE	QSUNSR
RATIONAL_B_SPLINE_CURVE	RBSC
RATIONAL_B_SPLINE_SURFACE	RBSS
RECTANGULAR_COMPOSITE_SURFACE	RCCMSR
RECTANGULAR_TRIMMED_SURFACE	RCTRSR
REPARAMETRISED_COMPOSITE_CURVE_SEGMENT	RCCS
REPRESENTATION	RPRSNT
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_MAP	RPRMP
REPRESENTATION_RELATIONSHIP	RPRRLT
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION	RRWT
SEAM_CURVE	SMCRV
SECURITY_CLASSIFICATION	SCRCLS
SECURITY_CLASSIFICATION_ASSIGNMENT	SCCLAS
SECURITY_CLASSIFICATION_LEVEL	SCCLLV
SERIAL_NUMBERED_EFFECTIVITY	SRNMEF
SHAPE_ASPECT	SHPASP
SHAPE_ASPECT_RELATIONSHIP	SHASRL
SHAPE_DEFINITION_REPRESENTATION	SHDFRP
SHAPE_REPRESENTATION	SHPRPR
SHAPE_REPRESENTATION_RELATIONSHIP	SHRPRL
SHELL_BASED_SURFACE_MODEL	SBSM
SHELL_BASED_WIREFRAME_MODEL	SBWM
SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION	SBWSR
SI_UNIT	SUNT
SOLID_ANGLE_MEASURE_WITH_UNIT	SAMWU
SOLID_ANGLE_UNIT	SLANUN
SOLID_MODEL	SLDMDL
SPECIFIED_HIGHER_USAGE_OCCURRENCE	SHUO
SPHERICAL_SURFACE	SPHSRF
START_REQUEST	STRRQS

表 B1(完)

长　　名	短　　名
START_WORK	STRWRK
SUPPLIED_PART_RELATIONSHIP	SPPRRL
SURFACE	SRFC
SURFACE_CURVE	SRFCRV
SURFACE_OF_LINEAR_EXTRUSION	SL
SURFACE_OF_REVOLUTION	SROFRV
SURFACE_PATCH	SRFPTC
SURFACE_REPLICA	SRFRPL
SWEPT_SURFACE	SWPSRF
TOPOLOGICAL_REPRESENTATION_ITEM	TPRPIT
TOROIDAL_SURFACE	TRDSRF
TRIMMED_CURVE	TRMCRV
UNCERTAINTY_MEASURE_WITH_UNIT	UMWU
UNIFORM_CURVE	UNFCRV
UNIFORM_SURFACE	UNFSRF
VECTOR	VECTOR
VERSIONED_ACTION_REQUEST	VRACRQ
VERTEX	VERTEX
VERTEX_LOOP	VRTLP
VERTEX_POINT	VRTPNT
VERTEX_SHELL	VRTSHL
VOLUME_MEASURE_WITH_UNIT	VMWU
VOLUME_UNIT	VLMUNT
WEEK_OF_YEAR_AND_DAY_DATE	WOYADD
WIRE_SHELL	WRSHL

附　录　C
（标准的附录）
协议实现一致性声明问卷

综述：

协议实现和一致性声明问卷支持对本标准各实现要求评估的一致性评定。此 PICS 问卷包括实验条件下(IUT)获得关于该实现的静态信息的一系列问题。这个信息即用于静态地评定有效的任选项选择，又用于使一个适当的一致性测试(动态评测)在一段时间具有形体化。

为了一致性实现可能的使用，本标准中标识了一些任选项。其中一些任选项可根据使用/不使用进行动态地(循环一时间)选择，例如，一个实体的 OPTION 属性。其他的任选项应根据使用/不使用进行

静态地(配置—时间)选择,如:一致性级别中定义的几何的某特定形式。

问题:

为了简化引用,需要产品或系统的标识,其中包含测试 STEP 实现和/或由此使该测试 STEP 实现得以完成。

1. 产品/系统标识(或名字):_____

一个一致性实现应至少提供本标准一致性级别 1 的功能度。本标准定义了六个一致性级别。每个级别规定了本标准应用解释模型结构的一个子集。这些级别的详细说明在本文档的 6.1.2 条。

2. 一致性的级别声明(功能度)—循环选择

1)—(CM 唯一)

2)—(CM & 曲面 & 线框 & 无拓扑)

3)—(CM & 线框 & 拓扑)

4)—(CM & 流形曲面 & 拓扑)

5)—(CM & 小平面边界表示)

6)—(CM & 高级的边界表示)

本标准的一致性可以在一个或多个不同的实现方法中实现。这些实现方法定义了本标准要求的交换行为类型。

3. 实现形状的声明—循环选择

交换结构(GB/T 16656.21)

如果该实现接收的数据与本标准中选择的一致性级别的要求不一致,或与选择实现方法的第 20 系列部分(Parts)的要求不一致,它应执行缺省回答。应静态设定缺省回答。

4. 缺省回答:_____

一致性实现方法在经过一系列不要求进行修正的动态评测(正在测试)后应保持选择静态任选项。在用户环境下,一致性实现应永远保持选择静态任选项的准备,或者它应提供用户任意控制静态任选项的变更或设置,或二者都有(根据其任选项情况定)。

5. IUT 给某用户提供各静态任选项的变更或设置的自由处理吗?

是/否?

6. 如果选取:是,那么取哪一个?

(a) 一致性级别(或各级别):_____

(b) 缺省回答:_____

一个一致性声明应包括至少一个团体相信的该实现的一致性的认证。

7. 评估人(或多人)(测试人/签证人/鉴定人):_____

附 录 D

(标准的附录)

实现方法规定要求

本标准的一致性应在一个或多个实现方法中获得。这些实现方法定义了本标准所要求的交换情况的各类型。当前定义了第一个实现方法:GB/T 16656.21。

对于一交换结构,其文件格式应根据其语法,在 GB/T 16656.21 中定义的 EXPRESS 语言映射和本标准附录 A 中定义的应用解释模型进行编码。此交换结构的头用概要名 'config_control_design' 标识本标准的职能。

附 录 E
（标准的附录）
信息对象注册

E1 文档标识

当本标准作为一个开放系统中的信息对象时，为了给其提供无二义的标识，将对象标示符 iso standard 10303 part (203) version(1)分配给了本标准。这个值的含义在 ISO/IEC 8824-1 中定义，并且在 GB/T 16656.1 中作了进一步说明。

E2 模式标识

GB/T 16656.1 进一步说明了怎样使用 ISO/IEC 8824-1 去标识单个模式。本标准包含两个这样的模式，为了在一个开放系统中给该模式提供无二义的标识，每个模式都分配有一个唯一的对象标示符。

E2.1 配置控制设计详细模式

对象标示符 iso standard 10303 part (203) version(1)object(1)config_control_design_schema(1)分配给了此配置控制设计详细模式(见附录 A)。

E2.2 配置控制设计简单模式

对象标示符 iso standard 10303 part (203)version(1)object(1)config_control_design-schema(2)分配给了此配置控制设计简单模式(见 5.2)。

附 录 F
（提示的附录）
应用活动模型

为了帮助理解在这个应用协议中定义的范围和信息的各要求提供了此应用活动模型。作为活动和数据二者的定义集合或作为活动图的集合给出该模型。它包含的活动超出了本应用协议的主题。本附录中给出的定义不能代替本标准正文中给出的定义。本附录的图使用了 IDEFO 标记。图 F1 给出了与每个活动箱相关的各箭头的简明解释。每个活动可以被分解，以便提供更多的细节。如果已经分解了一个活动，则包括有一本单行本图。

正像任何 IDEFO 模型一样，应用活动模型依赖于一特殊的视点和目的。此应用活动模型的视点来自设计工程师，此应用活动模型的目的是要阐明本应用协议的相关环境和范围。

F1 应用活动模型定义

在应用活动图中使用了下列术语。标有星号的术语超出了本应用协议的范围。

F1.1 管理数据：与产品设计有关的数据起源于设计管理。此类数据有：公司名称、供应商代码、安全分类和合同号。

F1.2 分析和测试数据*：在设计工艺产生的用于评价设计的数据。

F1.3 分析和测试模型*：用于评价设计的产品设计数据的表达。

F1.4 分析和测试设计*：对照它的要求评定产品的设计工艺。

F1.5 装配布置：单个的各个零件怎样安装在一起构成装配件的一种表达。

F1.6 装配要求：关于产品设计方面的限制，该限制与各零件安装在一起构成装配件的方式相匹配。

F1.7 更改要求计划：针对该工程设计更改的一个要求，进行正式工程答复描述的一文档。

F1.8 概念设计布置：设计思想怎样直观化变成产品的一种表达。

F1.9 概念化的产品设计*：领会设计思想、各要求和各种设计限制，并把它们转变成一正式的设计描述的工艺过程。这个工艺过程包括了概念设计布置和装配要求。

F1.10 配置控制的产品设计数据：产品设计数据存入正式的配置控制系统。

F1.11 配置控制的反馈*：与工程设计批准工艺过程有关的，来自标准、操作和工艺规程的偏差值。

F1.12 合同要求*：在与产品设计相匹配的法定文档中规定的各项。

F1.13 控制产品设计数据*：正式的产品标识，诸如：零件号和零件版本，它确保与既定的各要求、各标准、各操作和各工艺规程相一致。

F1.14 定义和正式提出产品设计：合成产品设计的工艺过程。这个工艺过程包括概念设计、初步设计和详细设计，并通过理论分析、设计模拟以及交替进行设计评估来完成。

F1.15 设计更改要求：对一个现存设计的任何形状、配合或者功能方面变更的正式要求。

F1.16 设计概念*：对该产品的各设计思想的正式说法。

F1.17 设计规定*：影响设计创作的限制，例如技术规定、成本/利润。

F1.18 设计反馈(设计替代)*：对于正式提出一新设计所使用的对原设计或替代设计的分析。

F1.19 设计思想*：对一个所提出的产品设计的形状、配合和功能等各要求概念化。

F1.20 设计集成结构：许多装配件以及子装配件如何能够安装在一起以形成各功能系统的表达。

F1.21 开发产品设计：在创建、分析、测试和发放一产品的形状、配合和功能的表达中所进行的活动。

F1.22 分发请求*：给一部门中适当的人提前送去其请求的该产品设计正式文档的复印件。

F1.23 分发跟踪信息：标识各部门或个人收到已发行的工程产品设计数据份数的数据。

F1.24 草图的修改：在一个特定部门里，根据标准绘草图工作程序进行草图更改。

F1.25 工程更改授权：为了标识该设计，发放或更改一个工程文档。它规定该设计要做什么并授权该设计更改活动。

F1.26 现存产品信息：关于产品的官方设计信息。

F1.27 检查、工艺、材料要求：满足对产品设计的形状、配合和功能要求的各检查要求、已完成的各工艺和各材料特性。

F1.28 生命周期要求*：由设计要求的信息，该信息贯彻了承载能力、工艺性和可维护性。

F1.29 经营设计*：为该设计工艺所创建的各进度表、各预算及各聘用职员要求的行为。

F1.30 管理指令、进度表、预算*：贯彻产品设计的管理计划活动所产生的结果。

F1.31 产品配置标识：对被批准的设计标识，它使用在产品结构中，如零件号、有效配置项。

F1.32 产品设计数据：由设计工艺产生的信息，它定义一产品的形状、配合和功能。

F1.33 产品要求*：规定产品功能和特性的依据，其包括使命、成本、功能和市场要求。

F1.34 产品结构：在一产品的定义中各项与每个其他项的关系。

F1.35 计划指令*：描述计划方针的文档，如对各标准技术、用户方向及特殊情况要求的修正。

F1.36 发放的工程产品设计数据：在一适当的工程授权控制下已经被发放的工程批准产品设计数据。

F1.37 发放状态：提供什么时间、什么内容及面向谁的等已被宣传的数据的信息情况。

F1.38 调查记录单*：除去由于产品设计的开发原因，而用于描述各工艺或各革新的任何文档。如贸易磋商、培训和专利描述。

F1.39 规范、标准、实践和过程：为确保各产品间的通用性而发布的影响设计活动的正式文档。

F1.40 标准件信息：来自于标准部门的产品设计数据。

F1.41 标准工具、硬件：现有的夹具和机器，它是设计工艺中的考虑对象。

F1.42 合成与精加工配置：从该设计的一个工段到整个产品都能确保其形状、配合及功能的工艺。

F1.43 工具要求*：支持产品设计的测试、机械加工和装配所必备的工具。

F2 应用活动模型图

在下列图中，范围以外的各活动和数据标有星号。

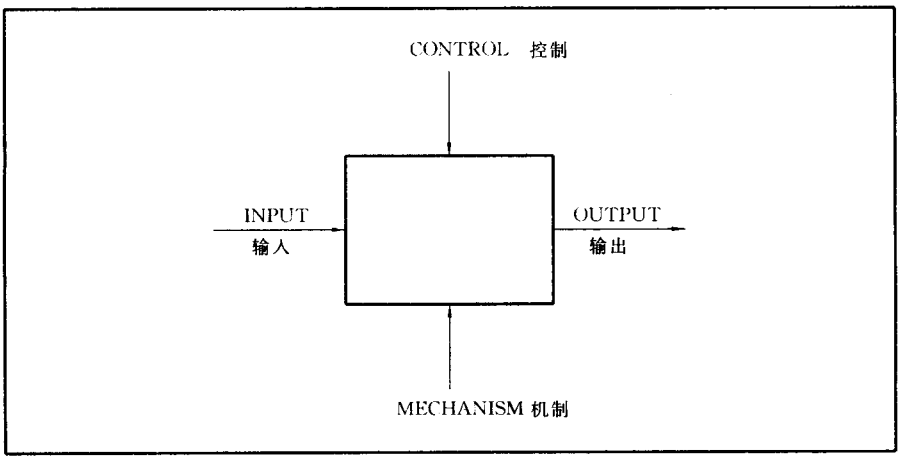


图 F1 IDEF0 方法的基本标志法

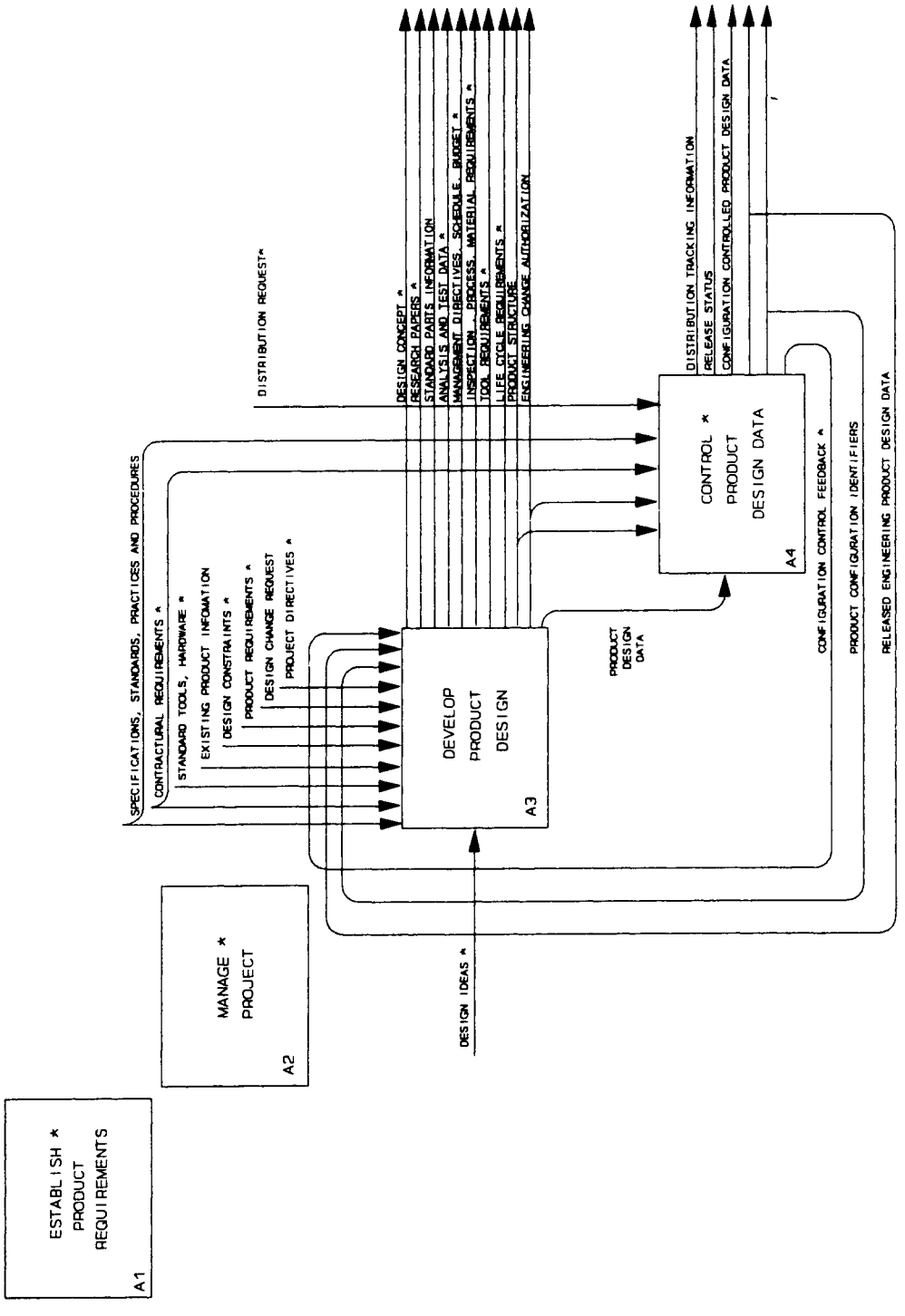


图 F2 用 IDEF0 方法描述的第 A0 层管理产品开发模块

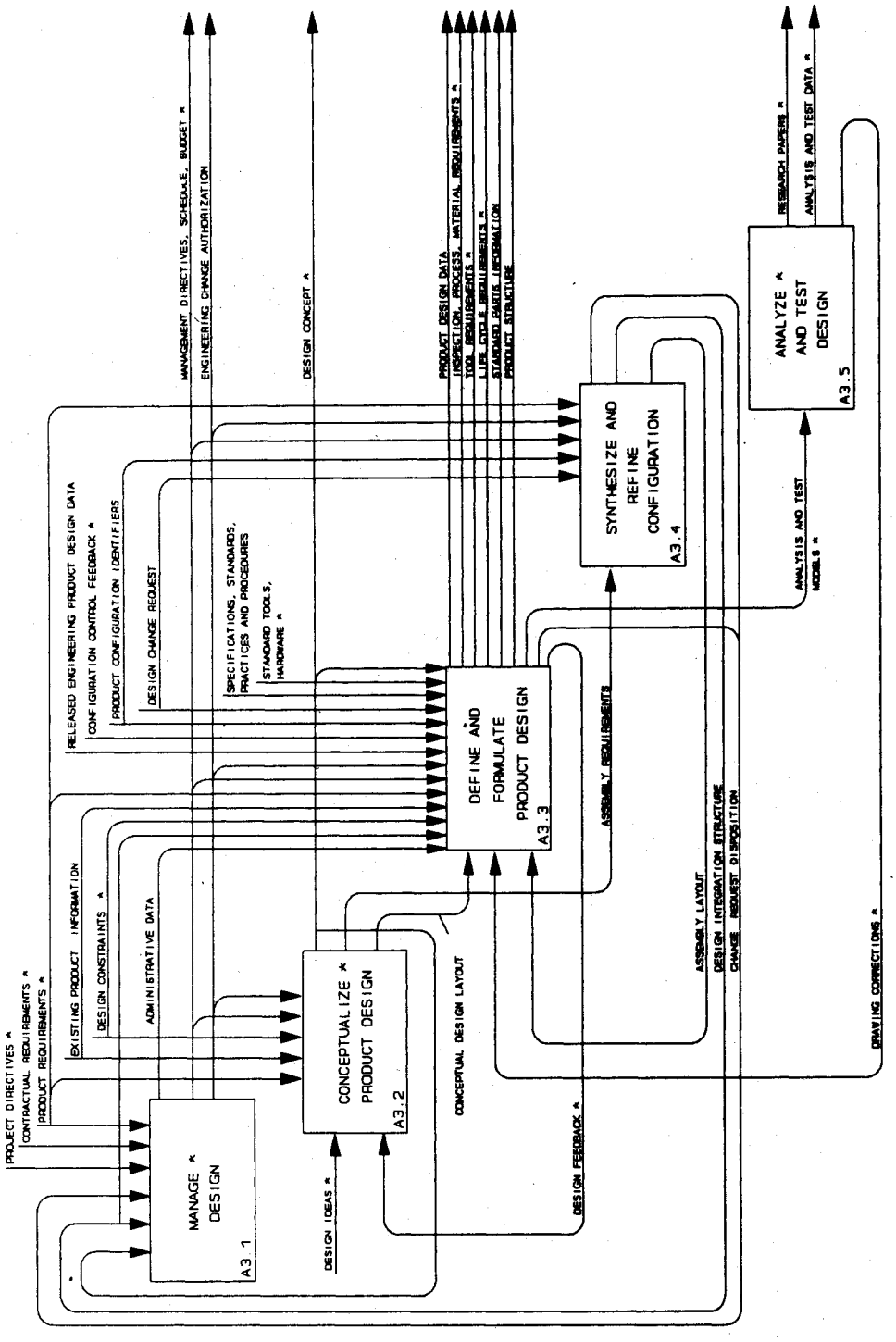


图 F3　用 IDEF0 方法描述的第 A3 层开发产品设计模块

附 录 G
（提示的附录）
应用参考模型

本附录为配置控制设计应用协议提供了应用参考模型。该应用参考模型给出了本标准第4章中规定的应用对象的结构和组件一种几何表示。该应用参考模型独立于任何实现方法。

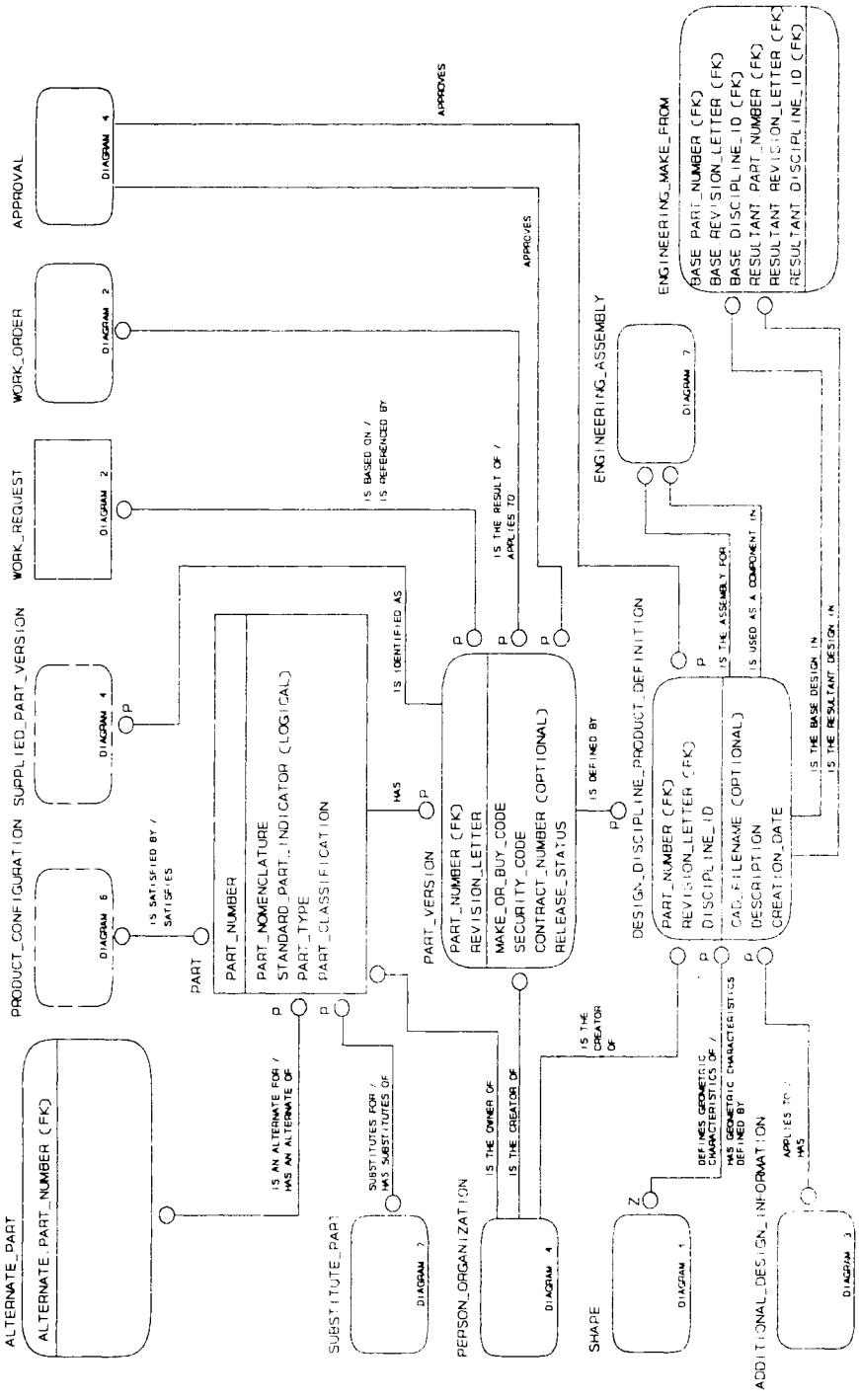


图 G1 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-1

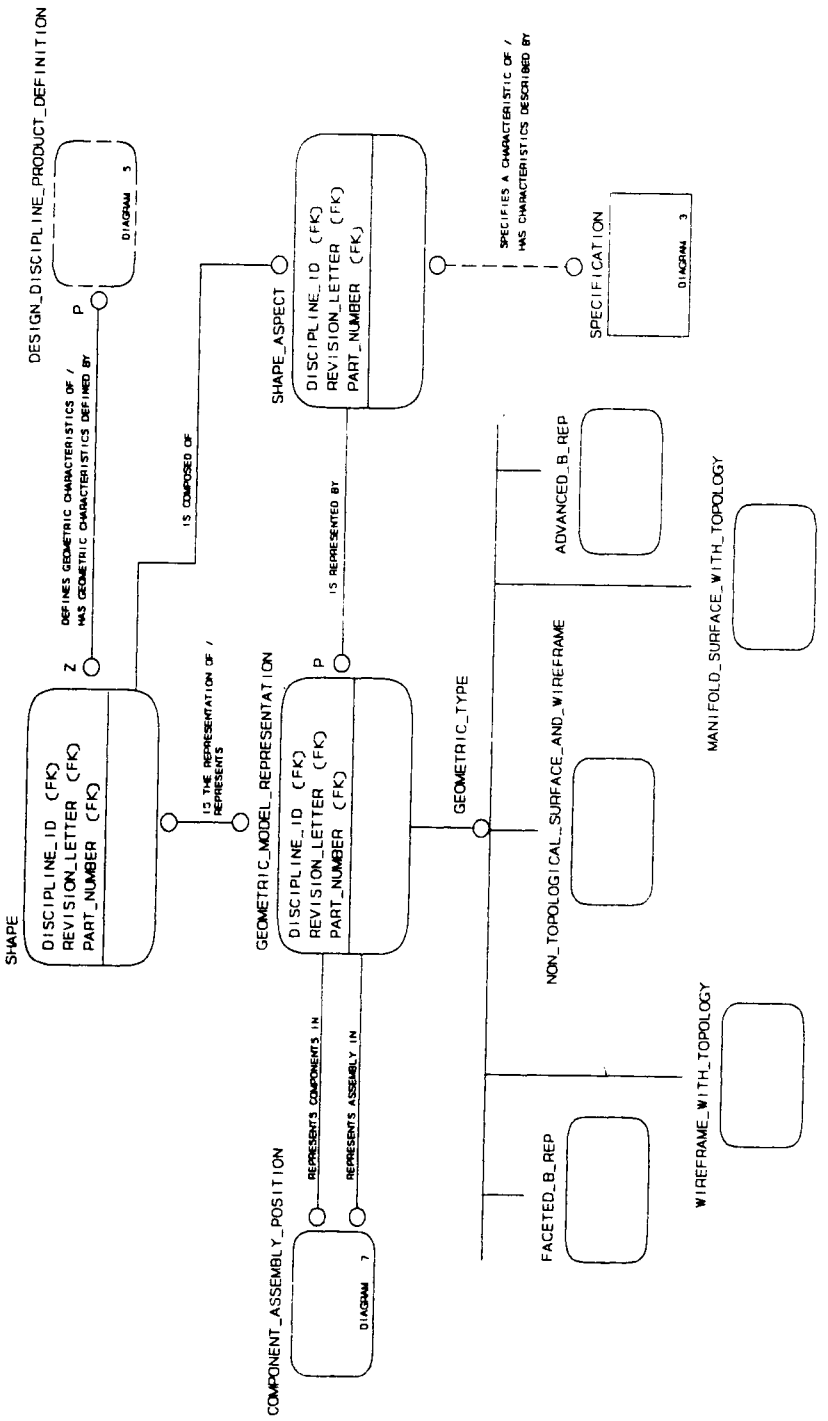
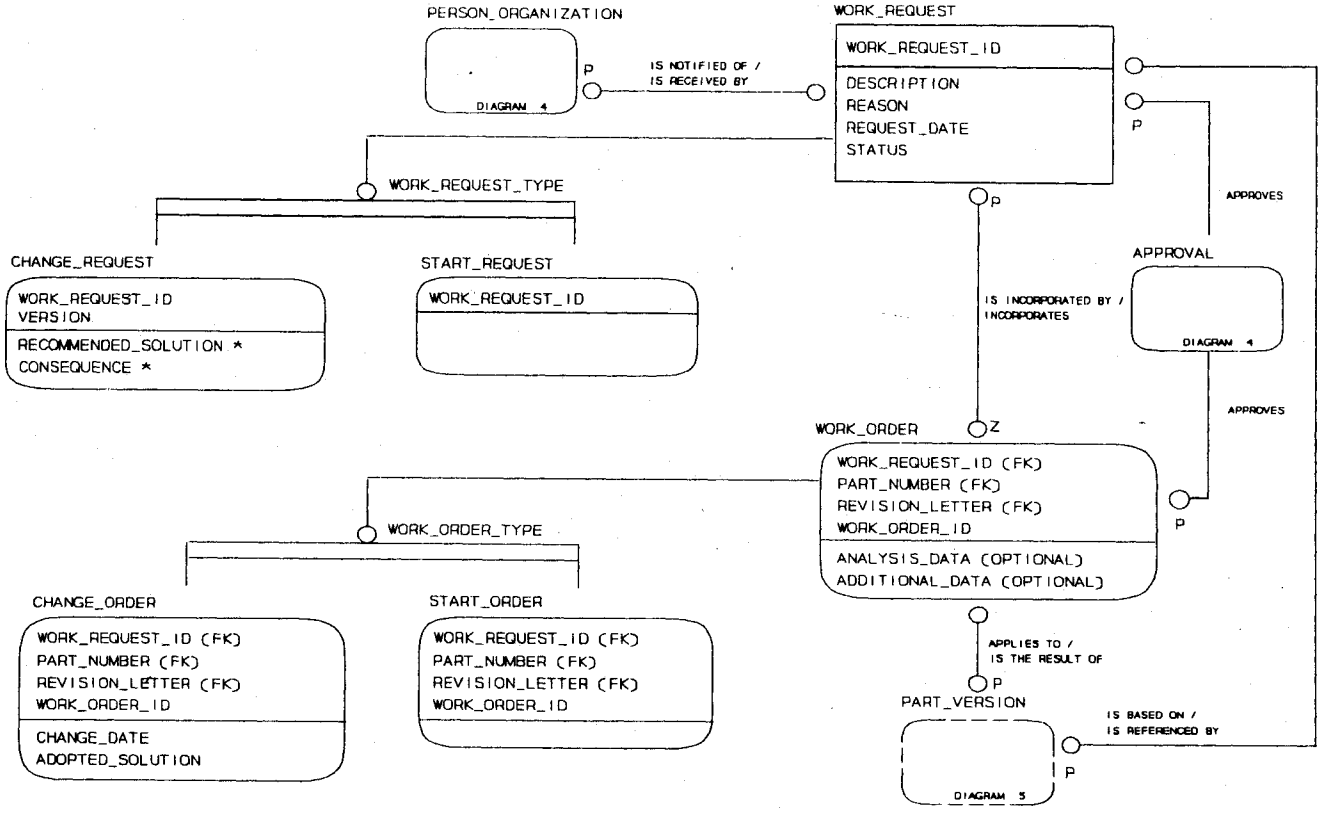


图 G2　用 IDEF1X 方法描述的应用参考模型(ARM)图 7-2



* 表示能够被组合成(一个或多个)的属性

注：如果一WORK-REQUEST(工作请求)要求>1(多于一个)的 PART_VERSION(零件版本)的工作，那么这些 PART-VERSION(零件版本)应该是属于不同零件的。

图 G3　用 IDEF1X 方法描述的应用参考模型(ARM)图 7-3

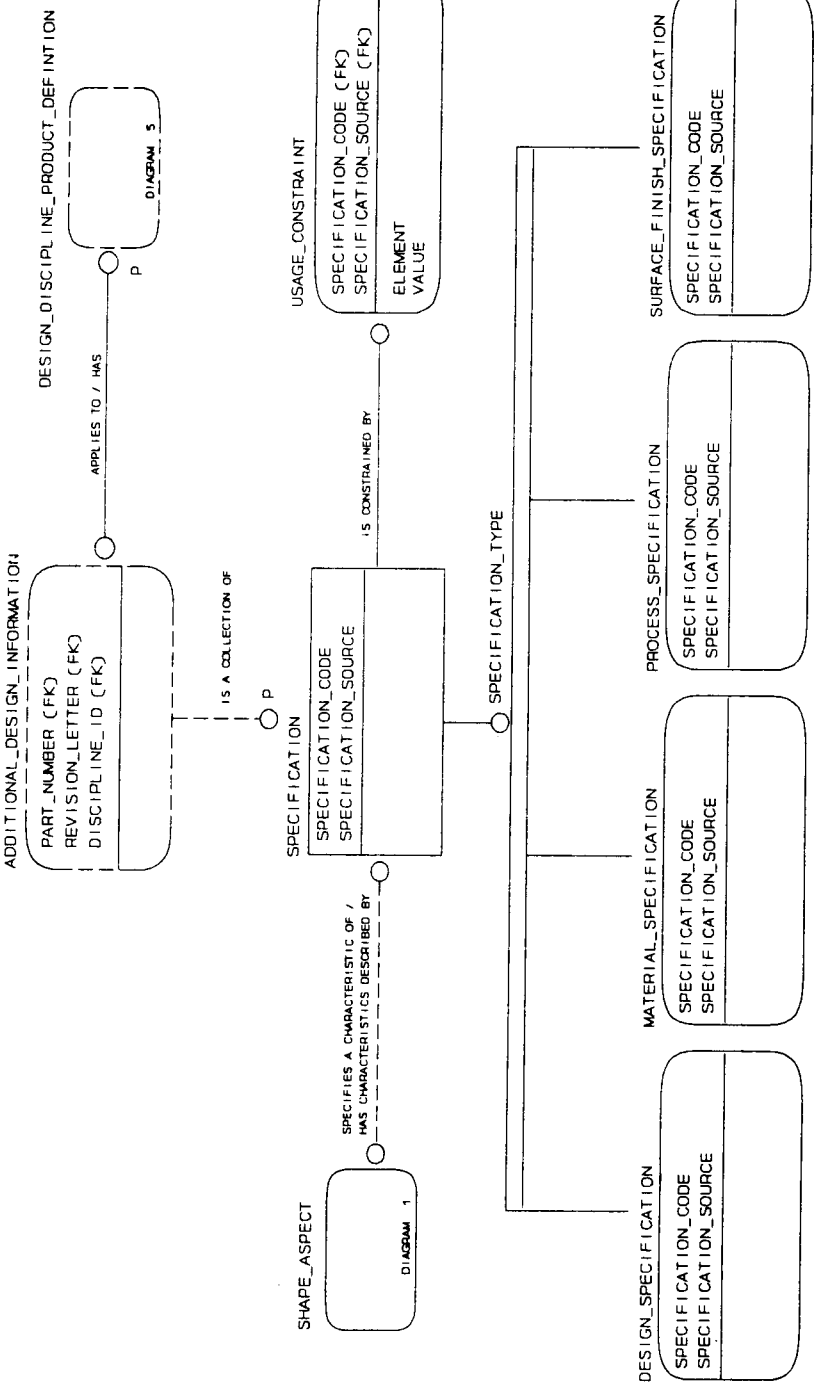


图 G4 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-1

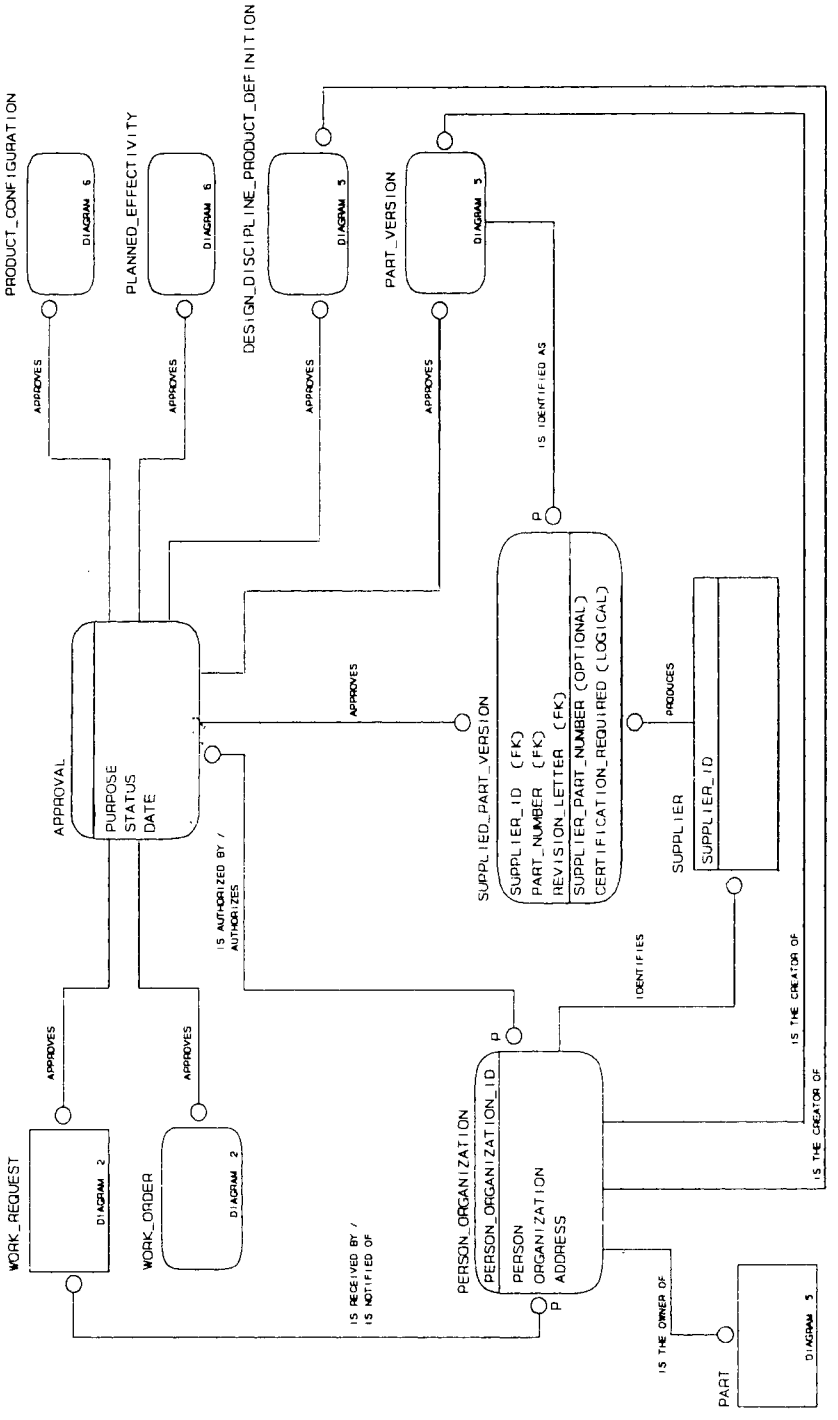


图 G5 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-5

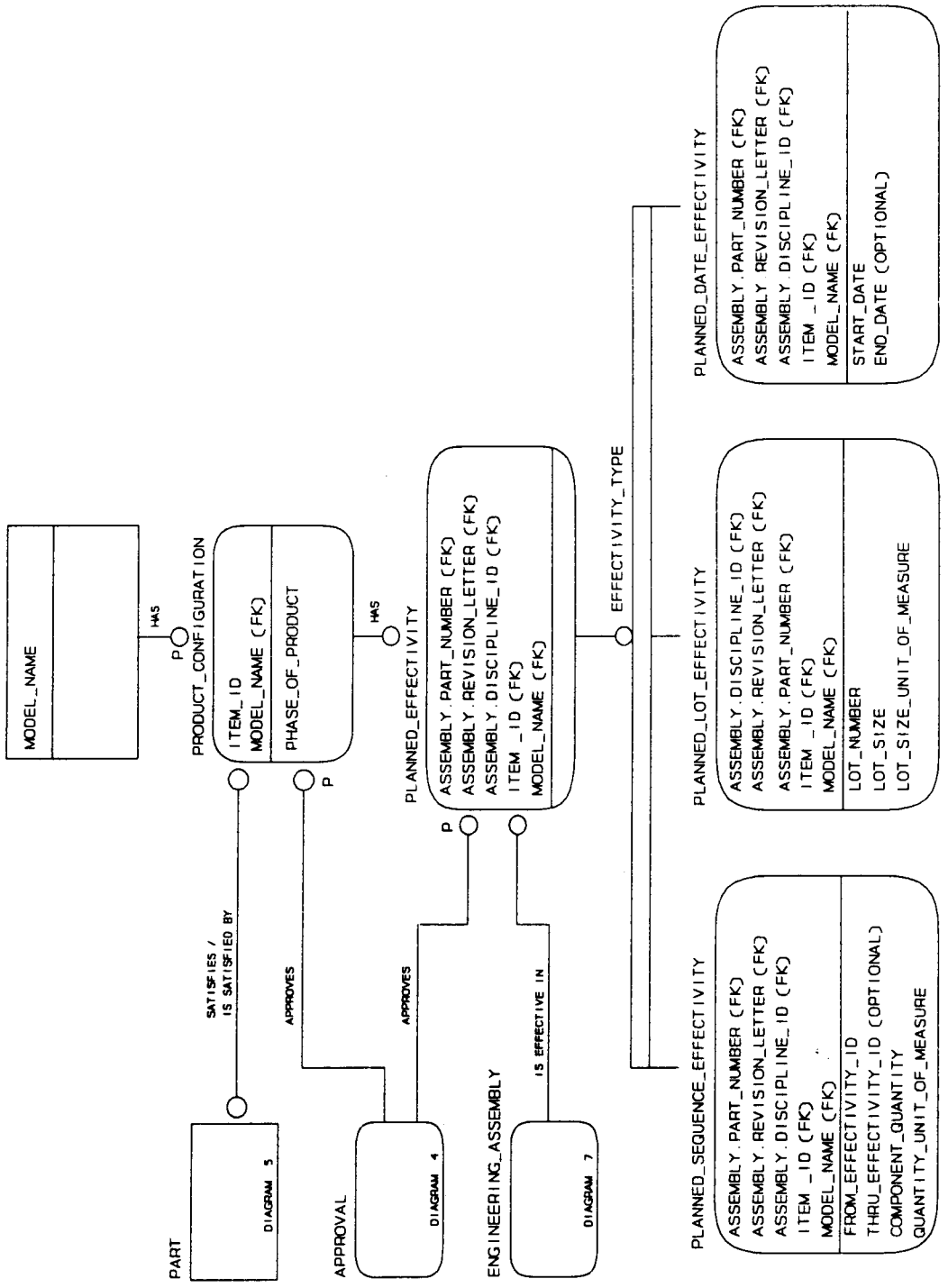


图 G6 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-6

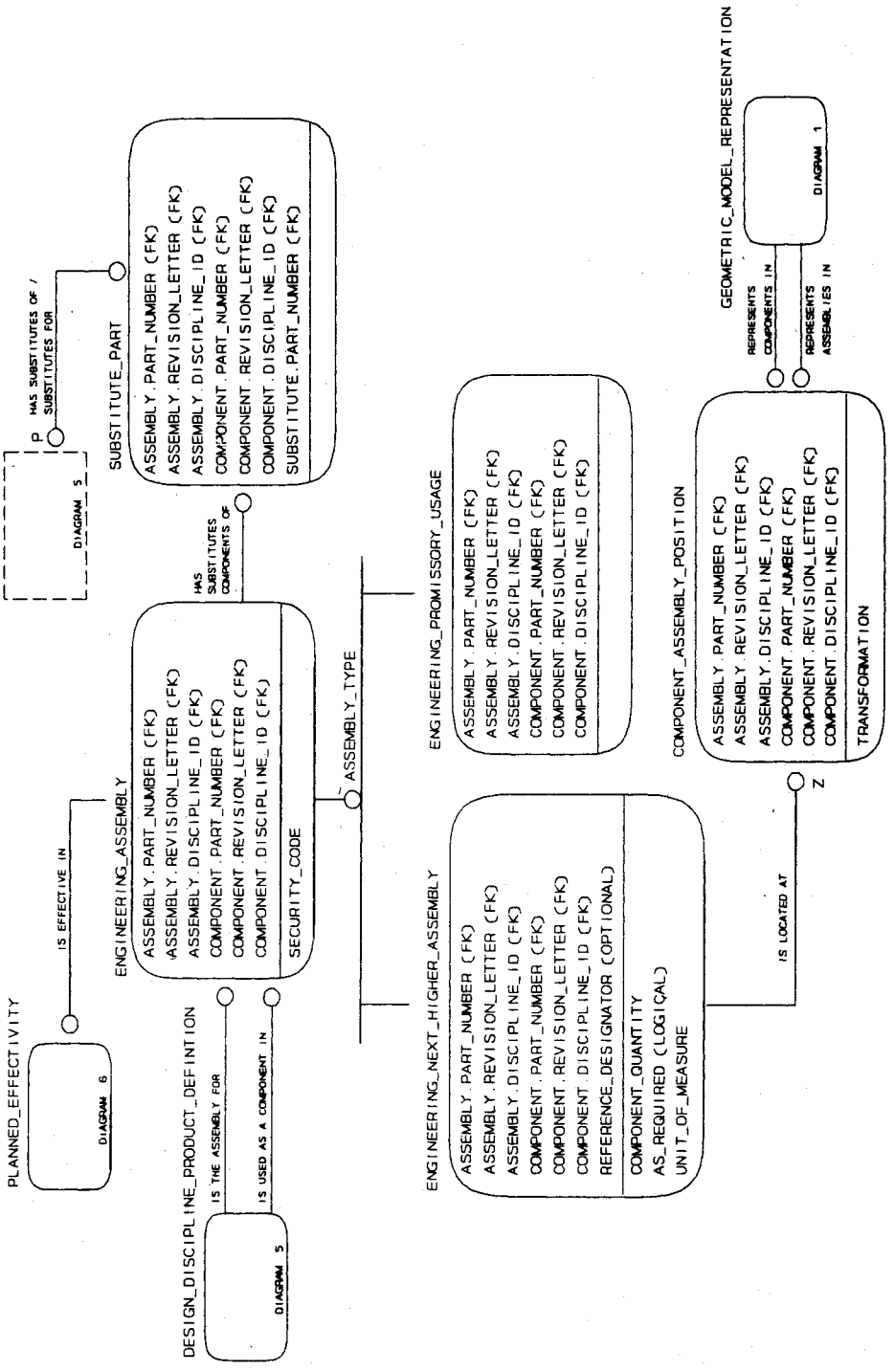


图 G7 用 IDEF1X 方法描述的应用参考模型(ARM)图 7-7

附 录 H
（提示的附录）
应用解释模型 EXPRESS-G

下面的图与附录 A 中给出的应用解释模型 EXPRESS 语言详表相对应。该图使用了 EXPRESS 语言的 EXPRESS-G 图解符号表示法。EXPRESS-G 的定义在 GB/T 16656.11 的附录 A 中。注意该页中间的注明资料来源对应的是此附录的图号，而不是目录上的图号。

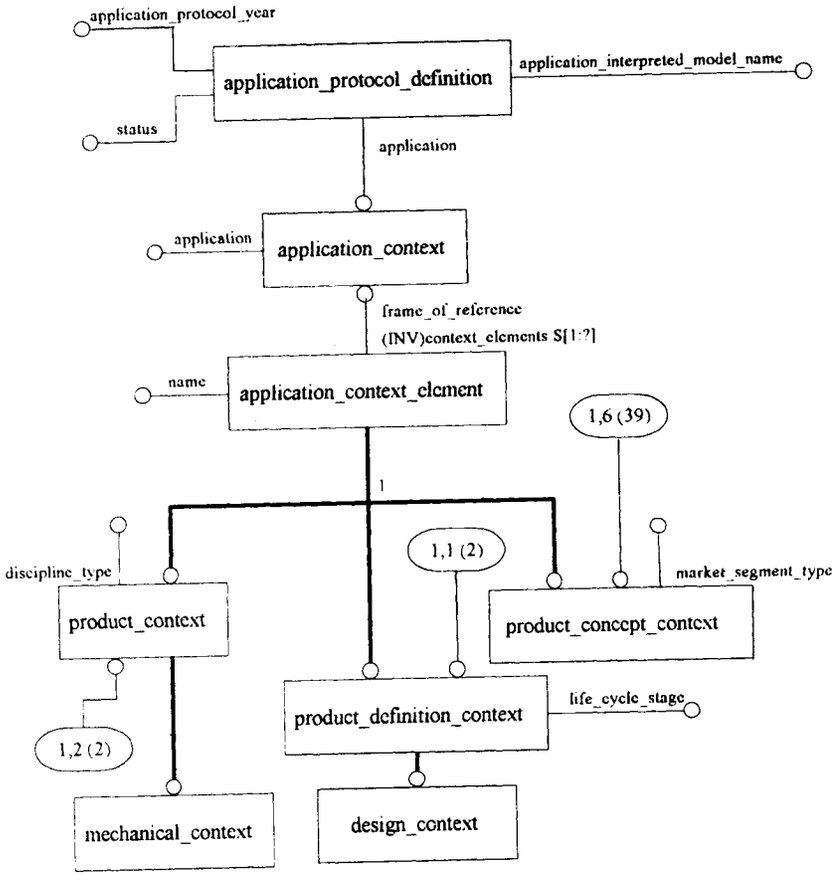


图 H1 应用相关环境-应用解释模型(AIM)EXPRESS-G 图 39-1

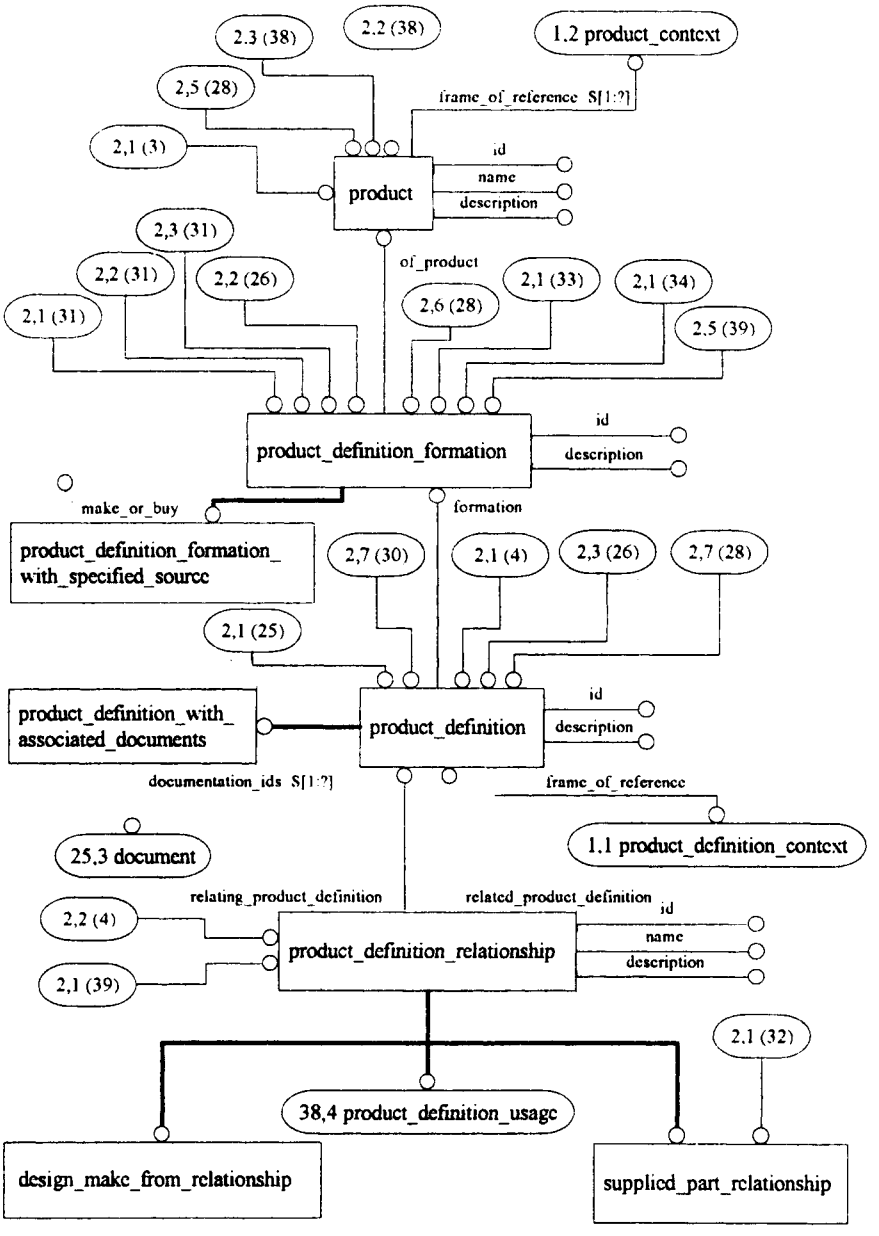


图 H2　产品定义-应用解释模型(AIM)EXPRESS-G 图 39-2

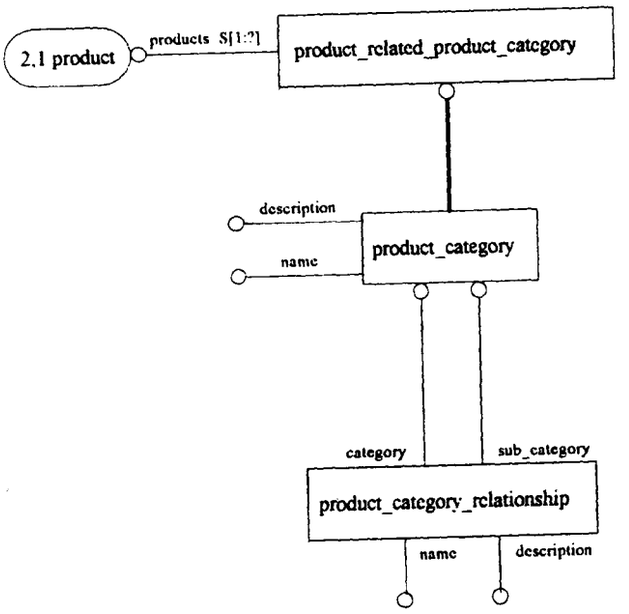


图 H3　产品目录-应用解释模型(AIM)EXPRESS-G 图 39-3

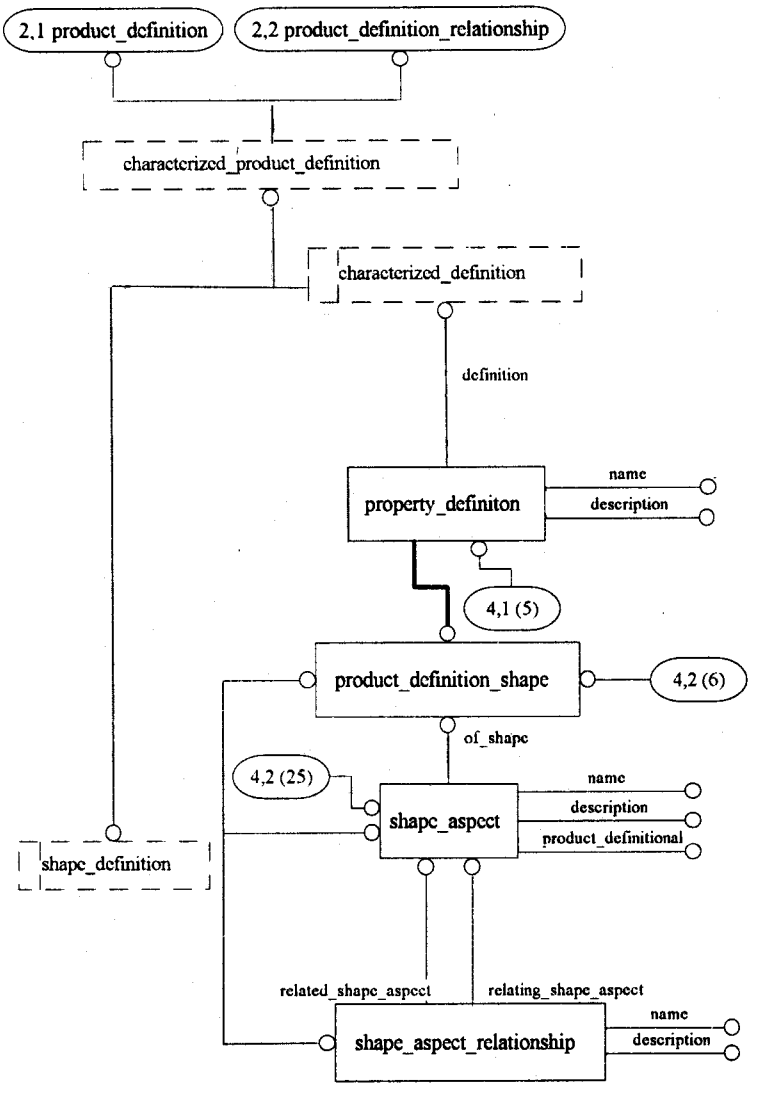


图 H4　参数定义-应用解释模型(AIM)EXPRESS-G 图 39-4

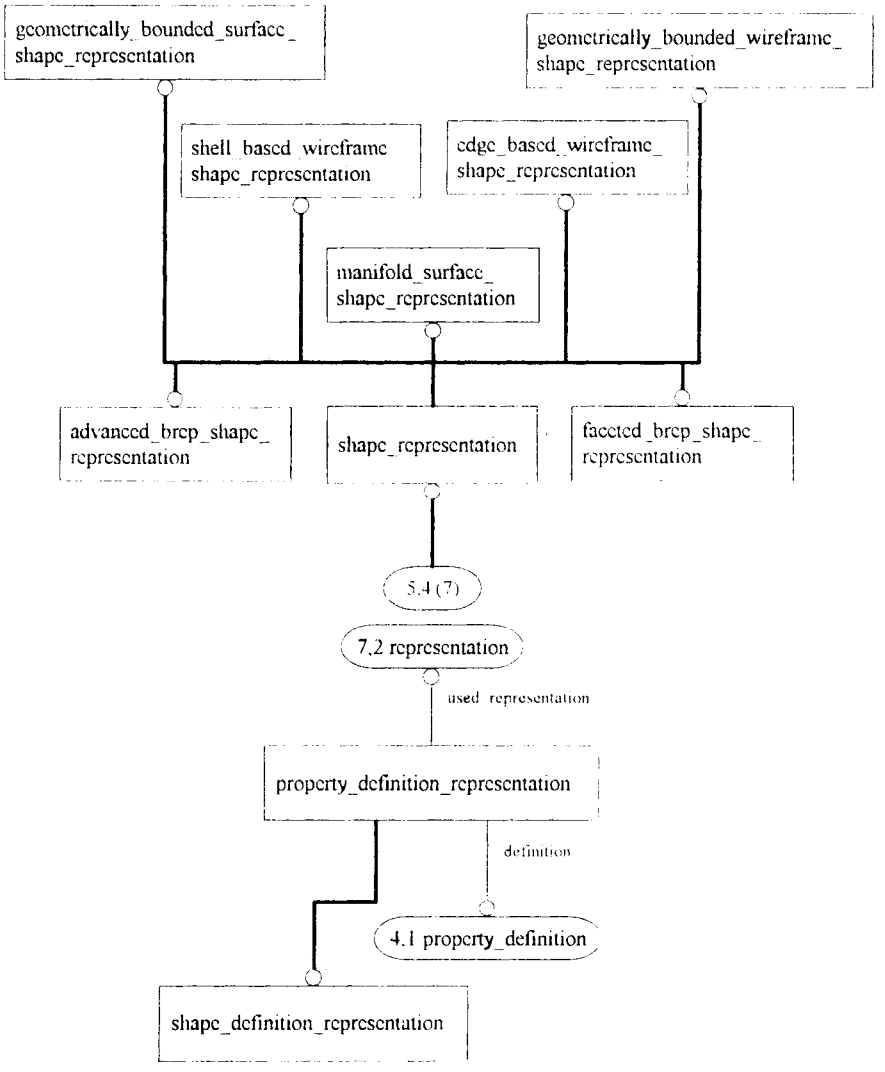


图 H5　参数表达-应用解释模型(AIM)EXPRESS-G 图 39-5

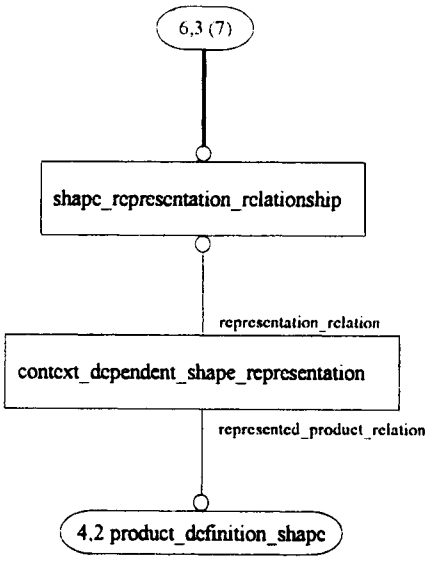


图 H6　形状表示关系-应用解释模型(AIM)EXPRESS-G 图 39-6

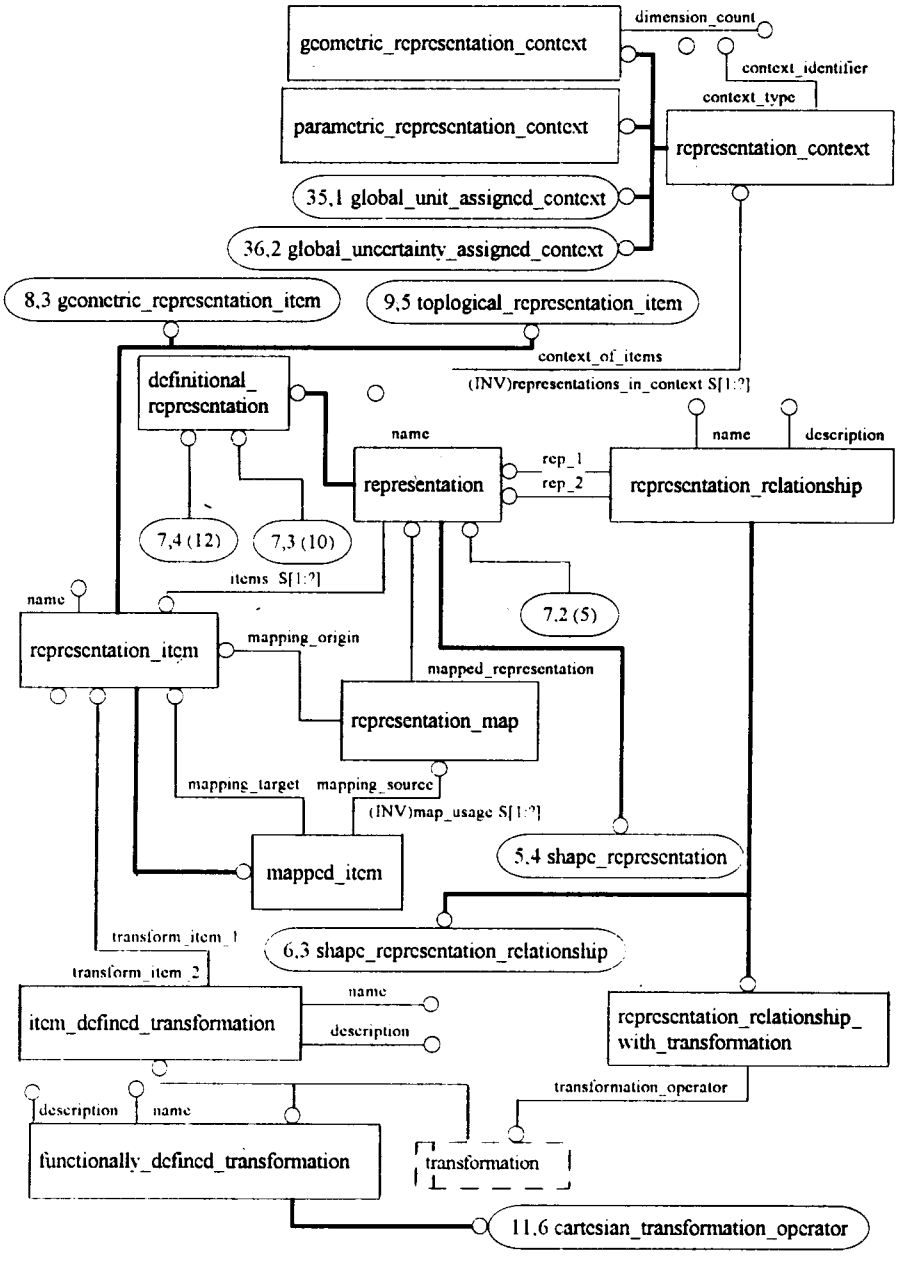


图 H7　表示-应用解释模型(AIM)EXPRESS-G 图 39-7

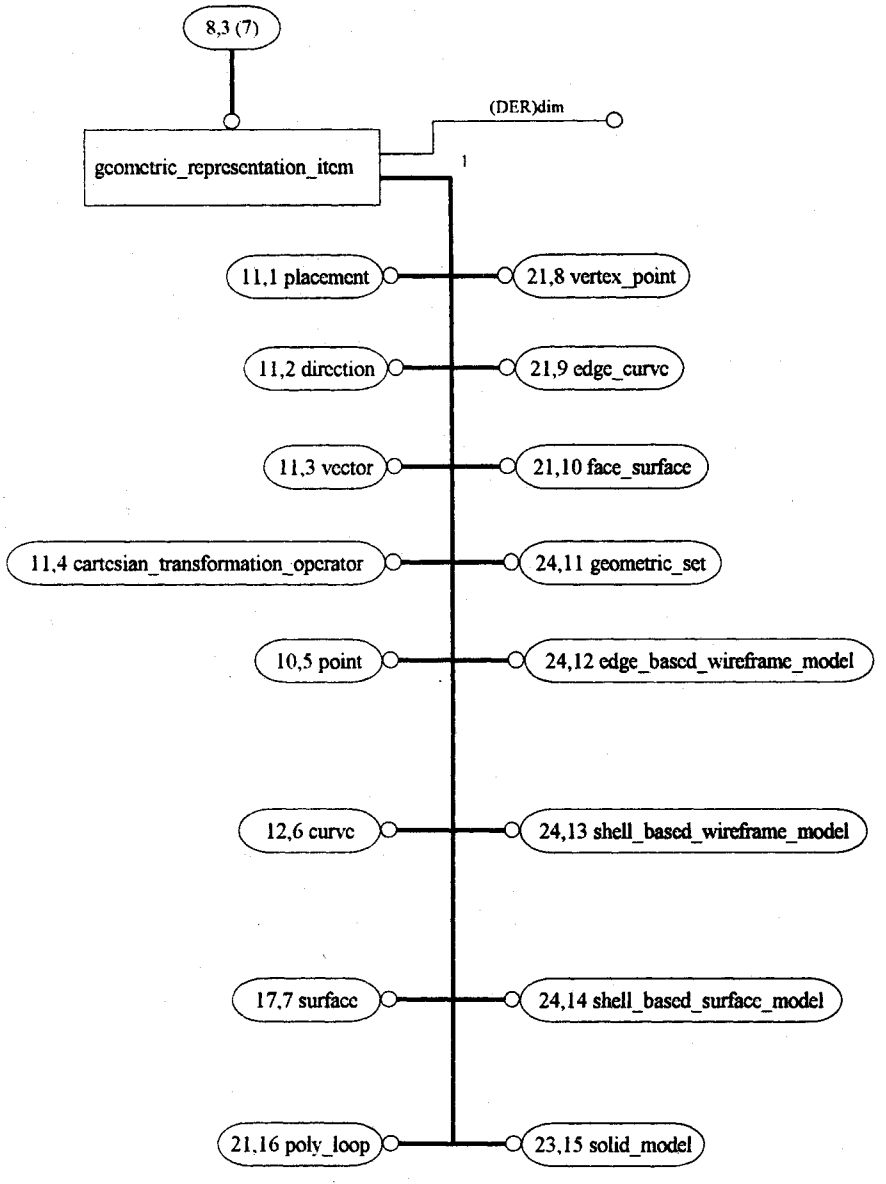
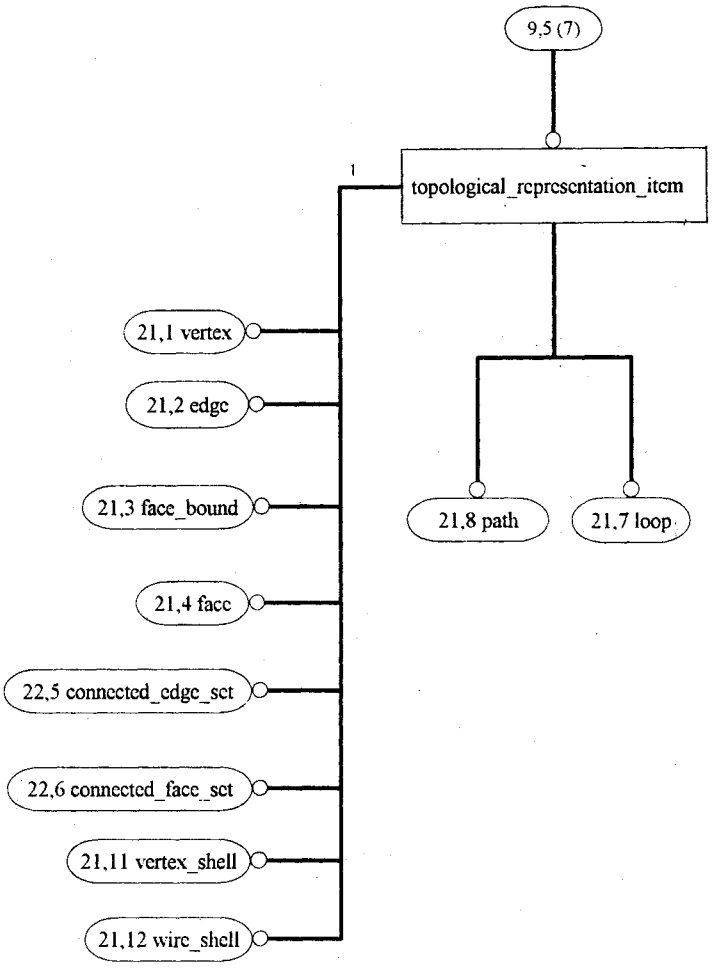


图 H8　几何表示项-应用解释模型(AIM)EXPRESS-G 图 39-8



NOTE - The tree showing loop ANDOR path participates in the ONEOF relationship among the entities within the other tree.

图 H9　拓扑表示项-应用解释模型(AIM)EXPRESS-G 图 39-9

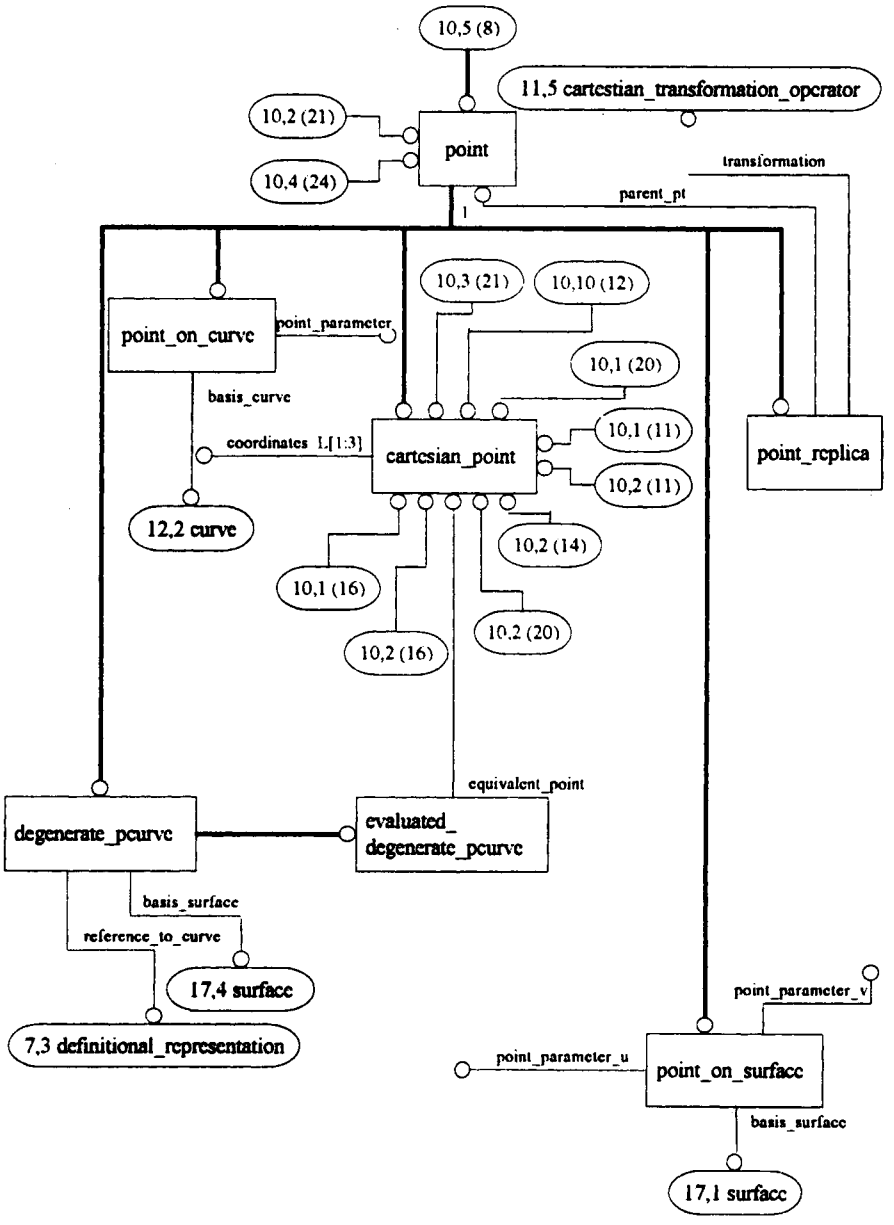


图 H10 点-应用解释模型(AIM)EXPRESS-G 图 39-10

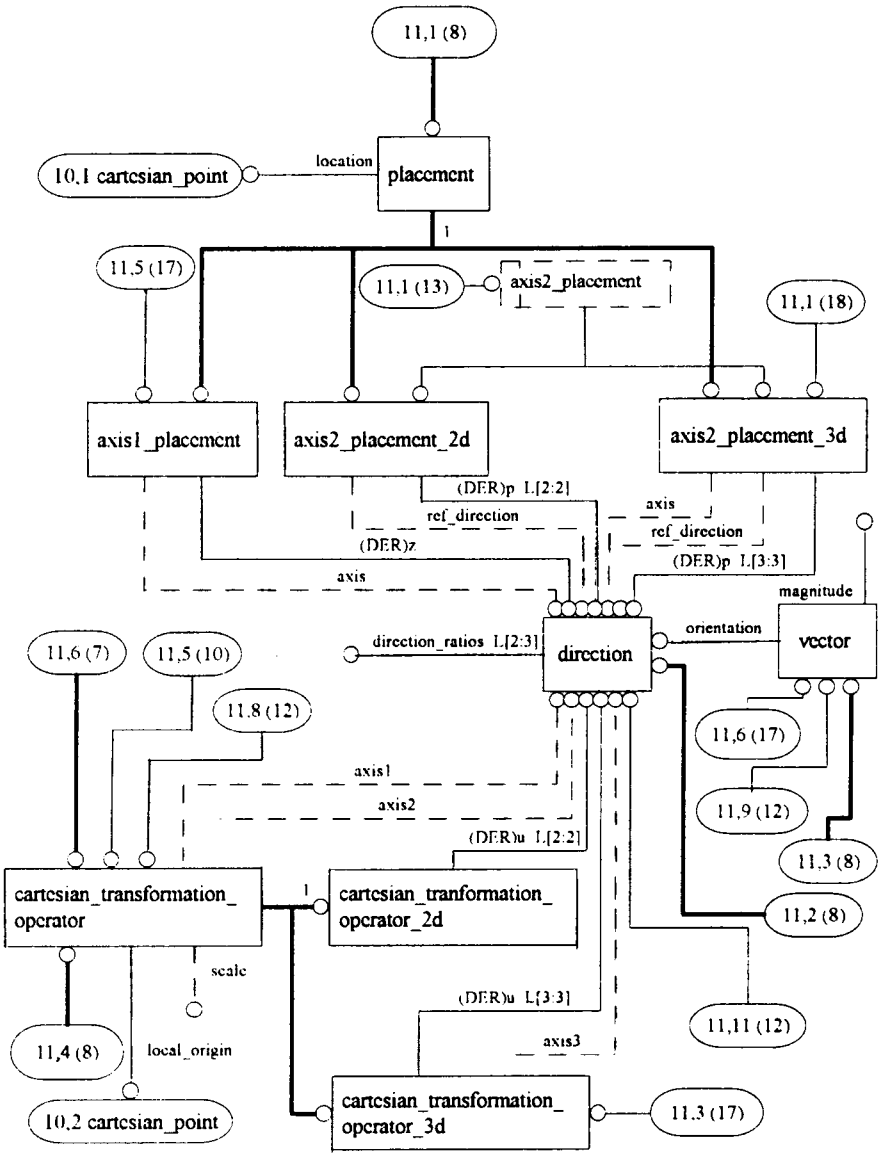


图 H11　几何有向-应用解释模型(AIM)EXPRESS-G 图 39-11

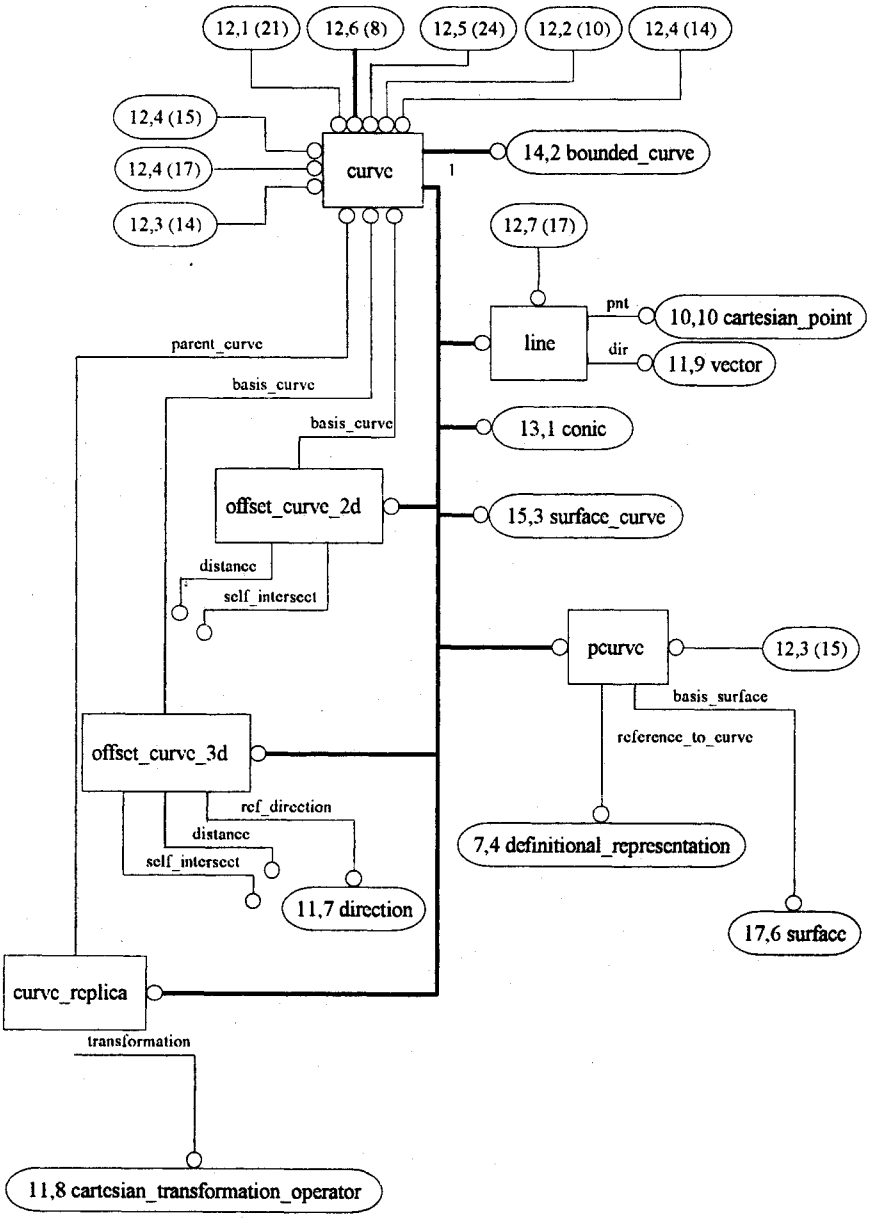


图 H12　曲线-应用解释模型(AIM)EXPRESS-G 图 39-12

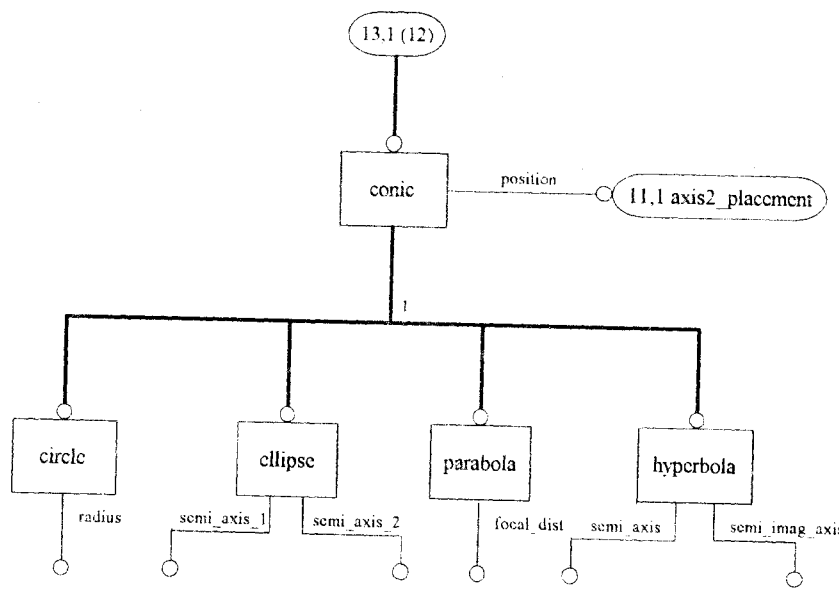


图 H13　圆锥曲线-应用解释模型(AIM)EXPRESS-G 图 39-13

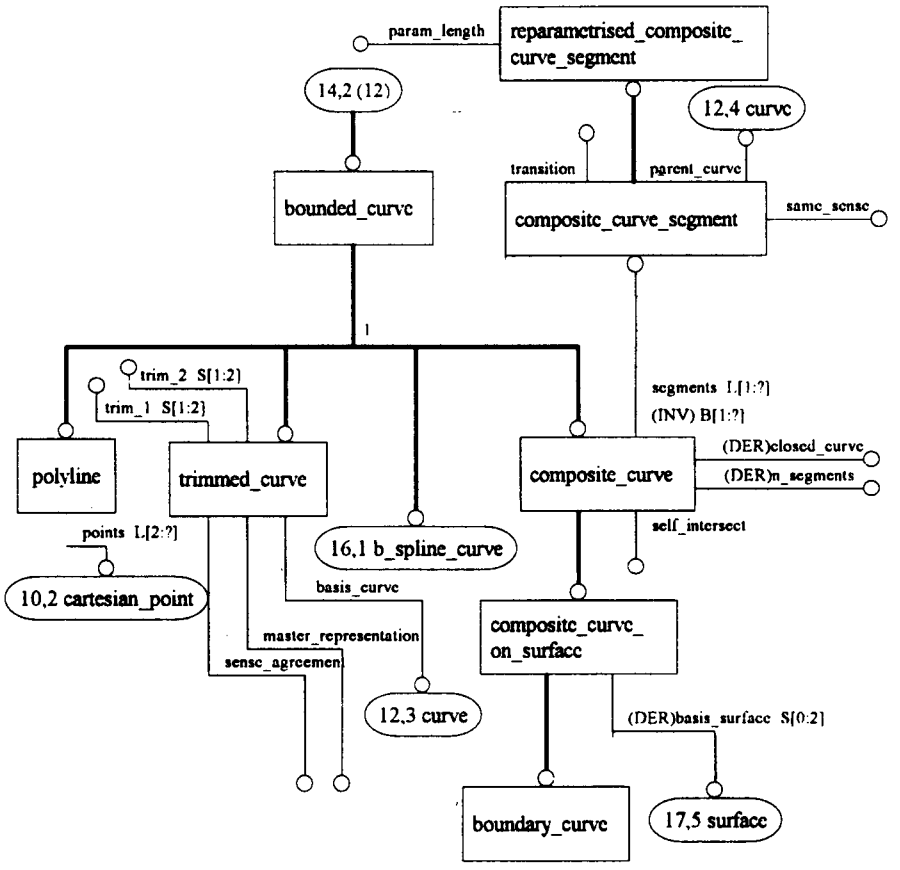


图 H14　有界曲线-应用解释模型(AIM)EXPRESS-G 图 39-14

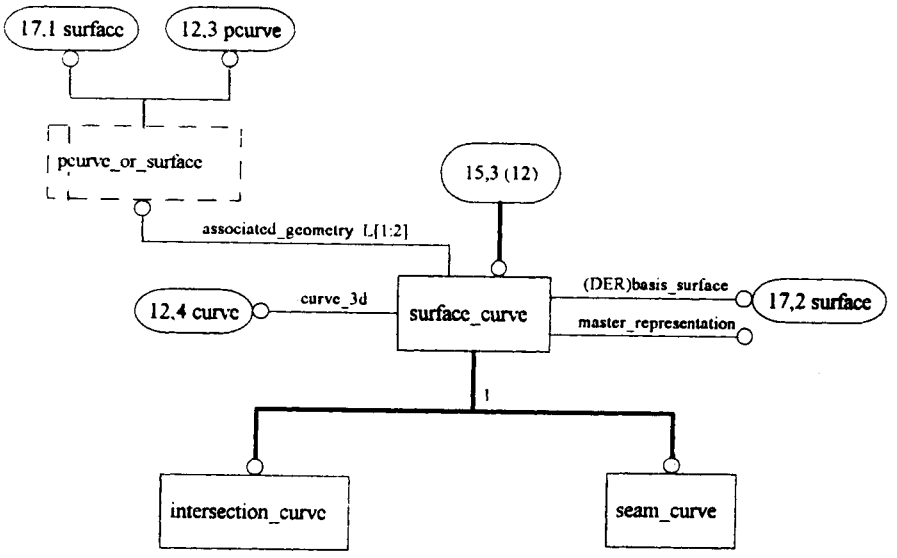


图 H15　曲面曲线-应用解释模型(AIM)EXPRESS-G 图 39-15

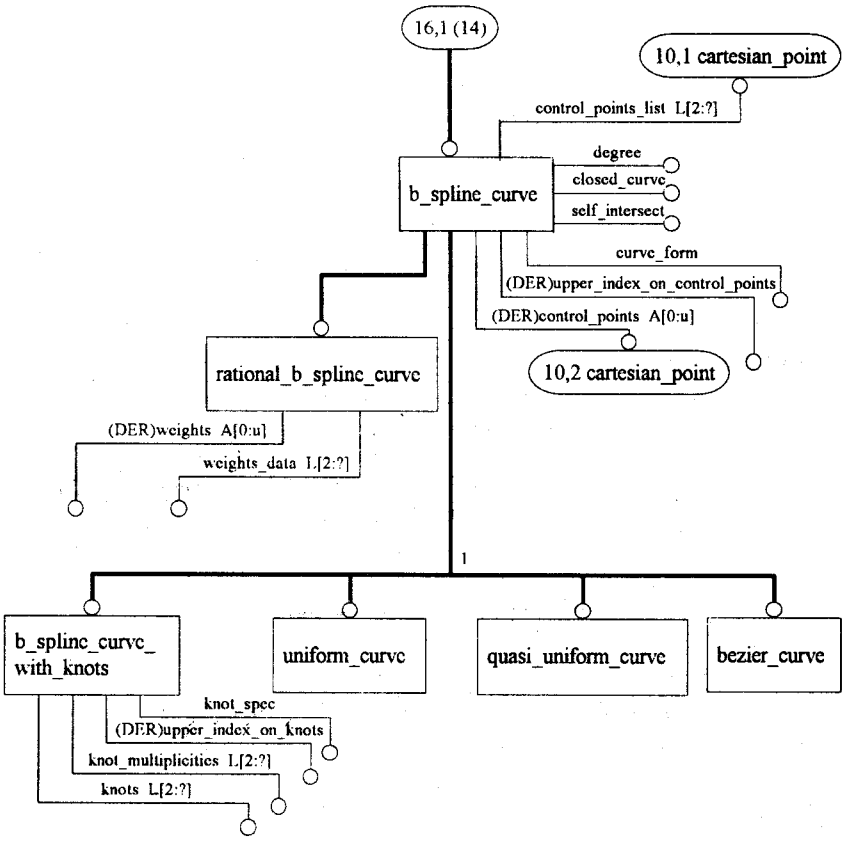


图 H16　B-样条曲线-应用解释模型(AIM)EXPRESS-G 图 39-16

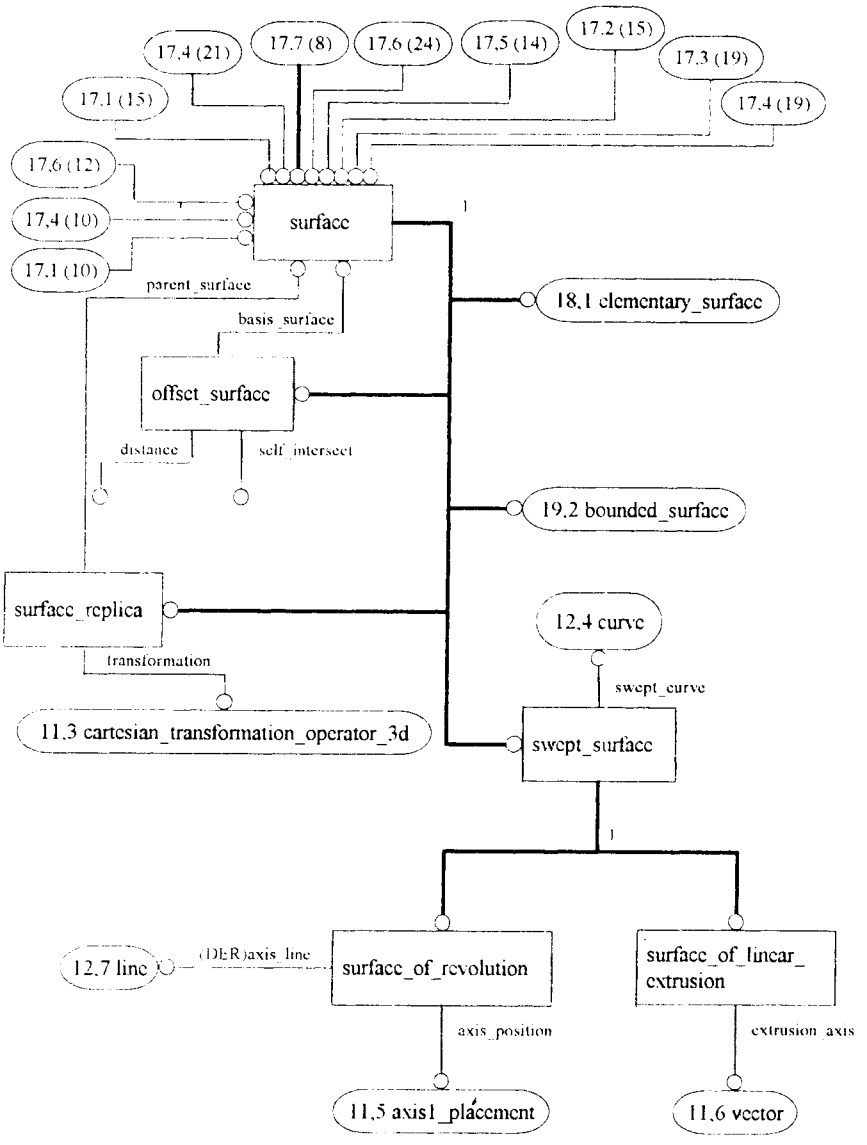


图 H17　曲面-应用解释模型(AIM)EXPRESS-G 图 39-17

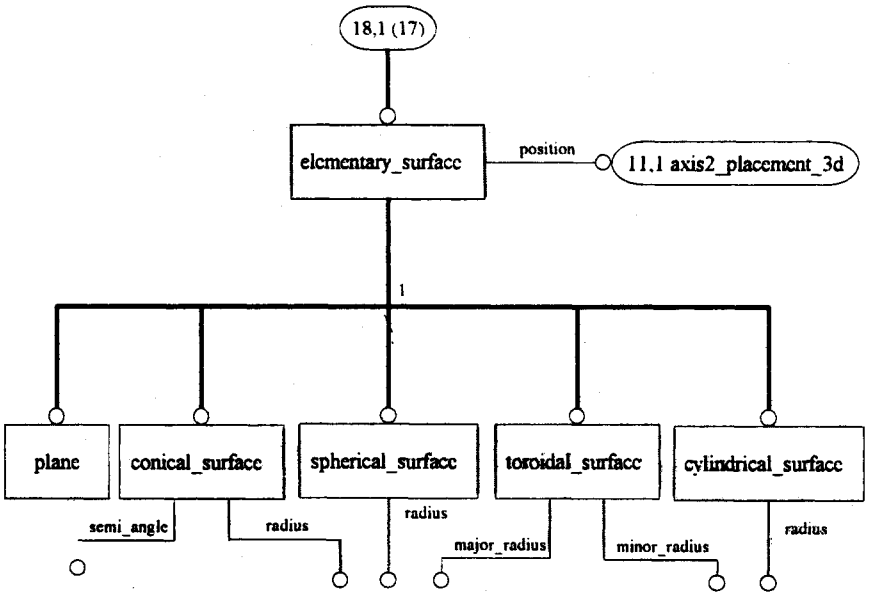


图 H18　基本曲面-应用解释模型(AIM)EXPRESS-G 图 39-18

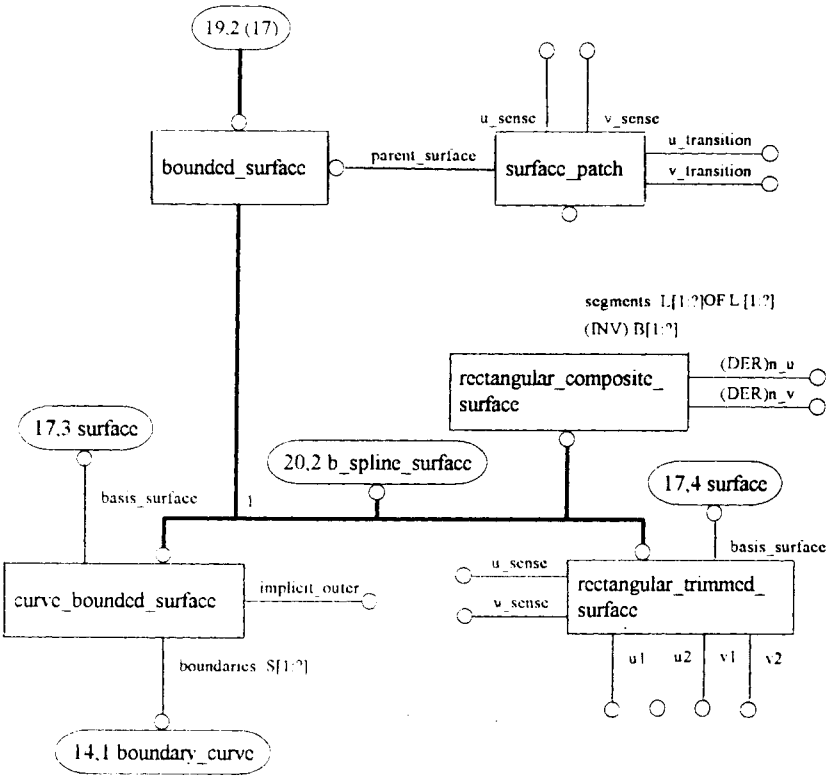


图 H19　有界曲面-应用解释模型(AIM)EXPRESS-G 图 39-19

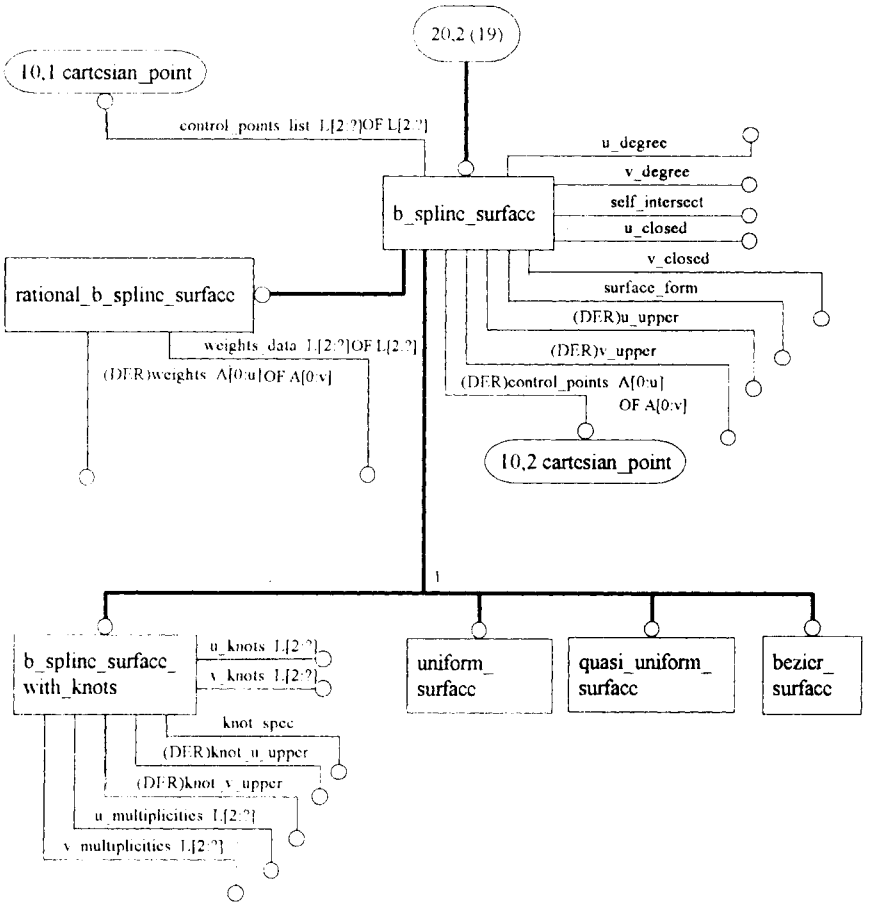


图 H20 B-样条曲面-应用解释模型(AIM)EXPRESS-G 图 39-20

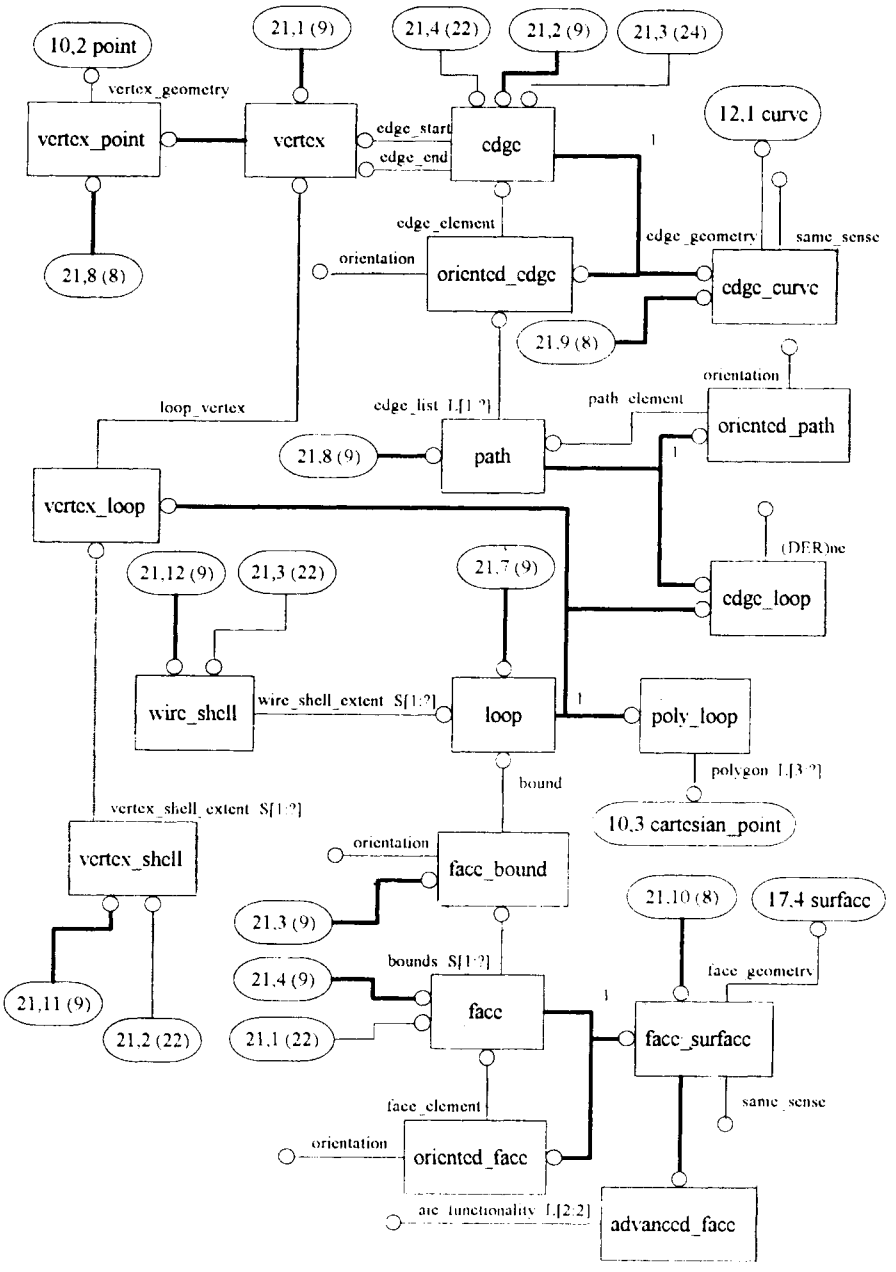


图 H21　拓扑-应用解释模型(AIM)EXPRESS-G 图 39-21

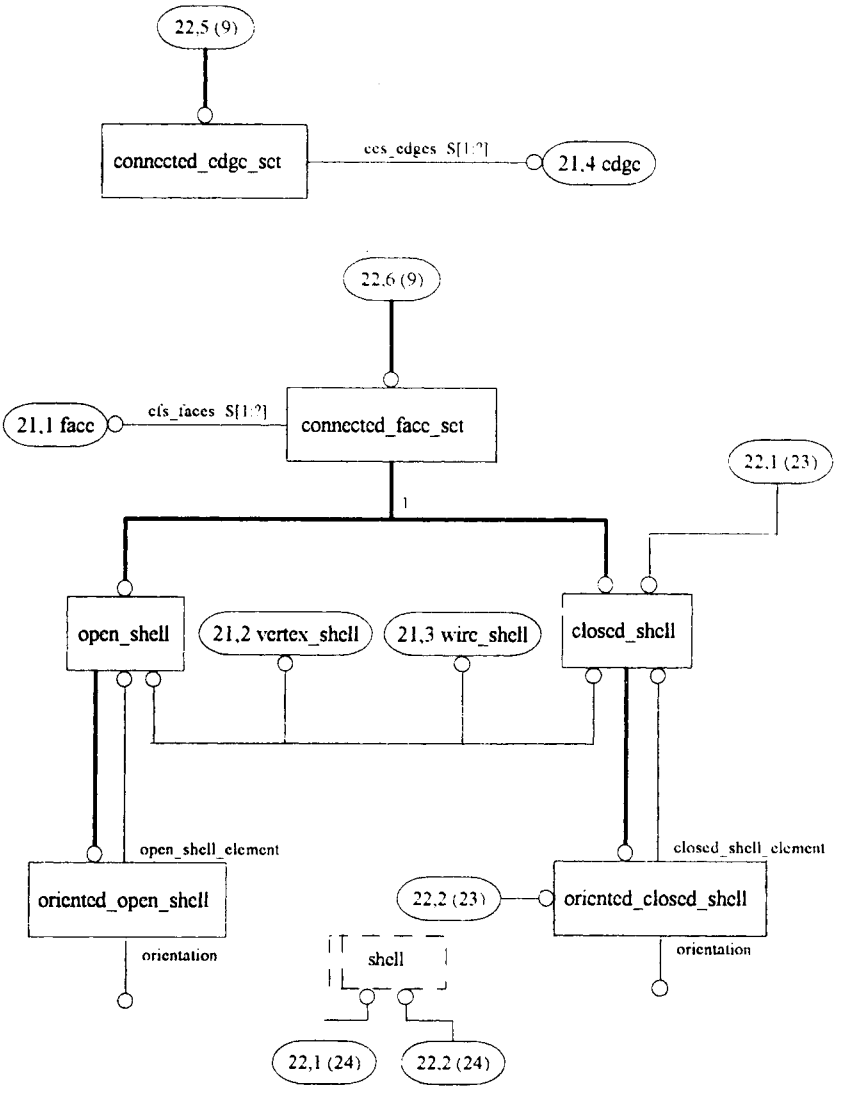


图 H22　壳-应用解释模型(AIM)EXPRESS-G 图 39-22

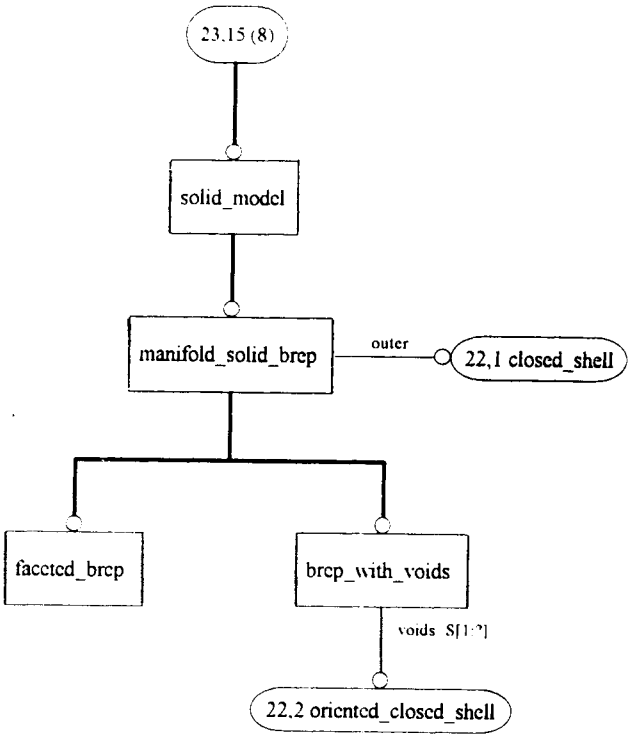


图 H23　实体模型-应用解释模型(AIM)EXPRESS-G 图 39-23

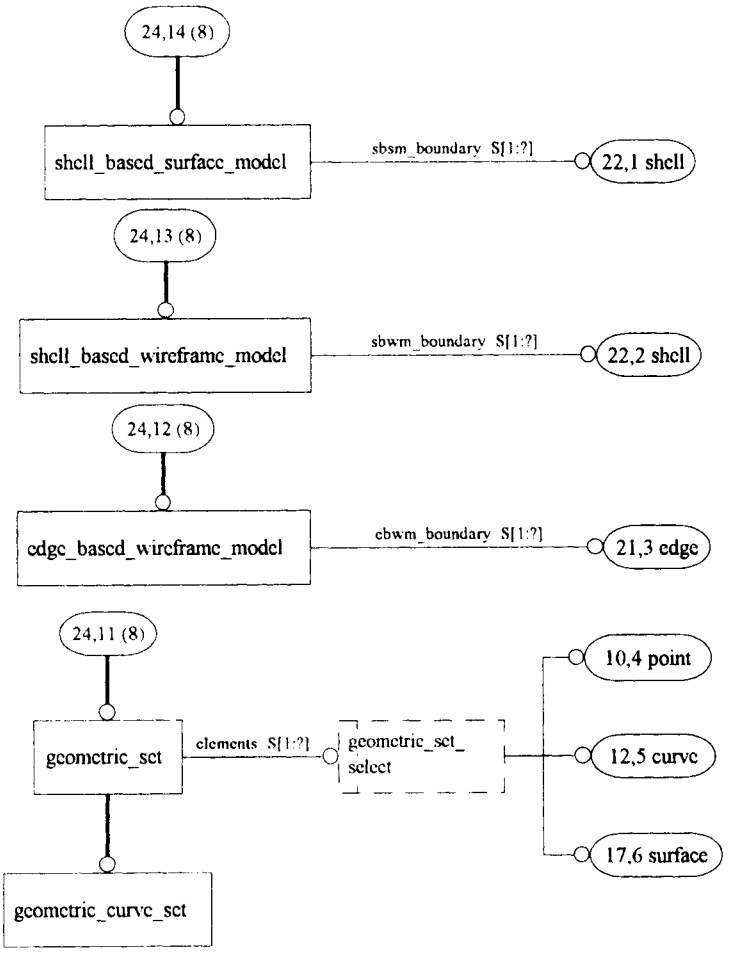


图 H24　曲面和线框模型-应用解释模型(AIM)EXPRESS-G 图 39-24

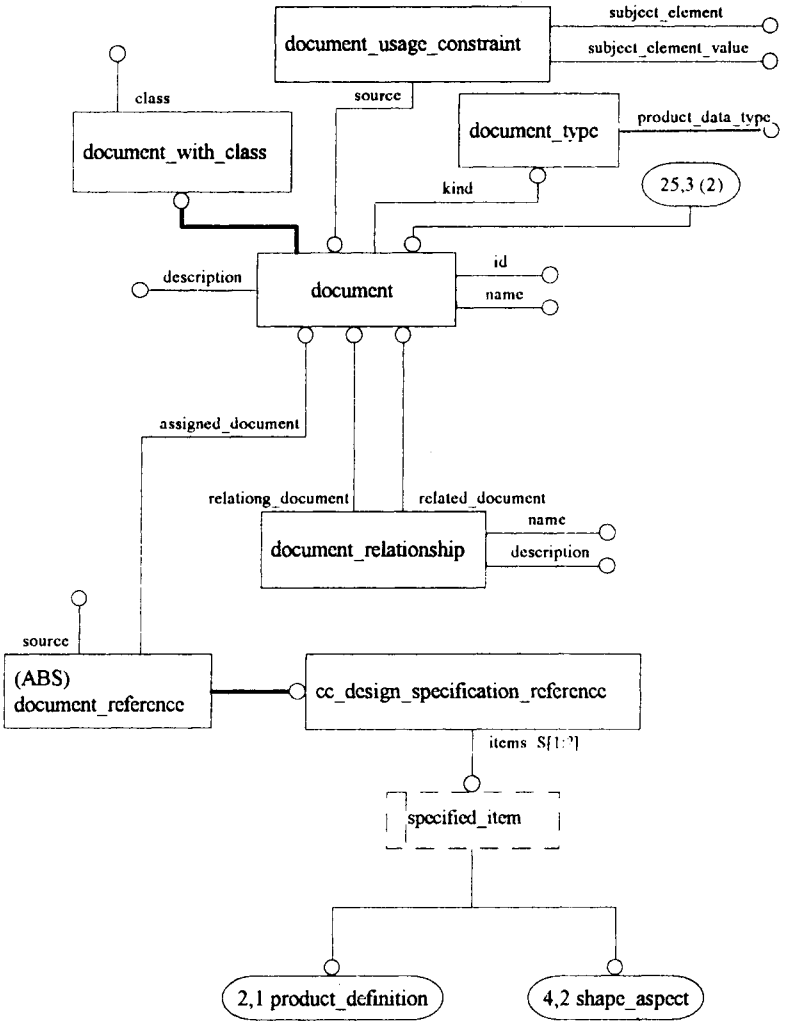


图 H25　文档-应用解释模型(AIM)EXPRESS-G 图 39-25

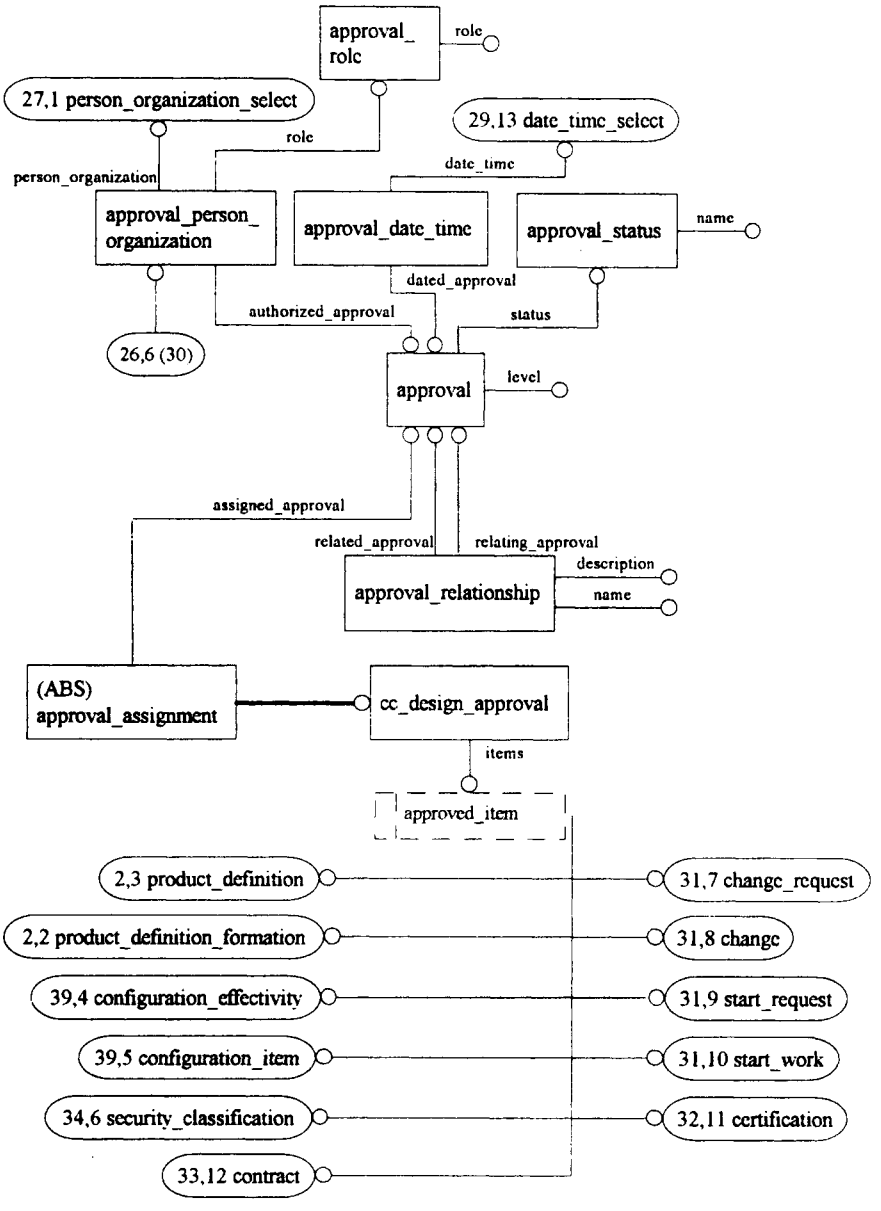


图 H26　批准-应用解释模型(AIM)EXPRESS-G 图 39-26

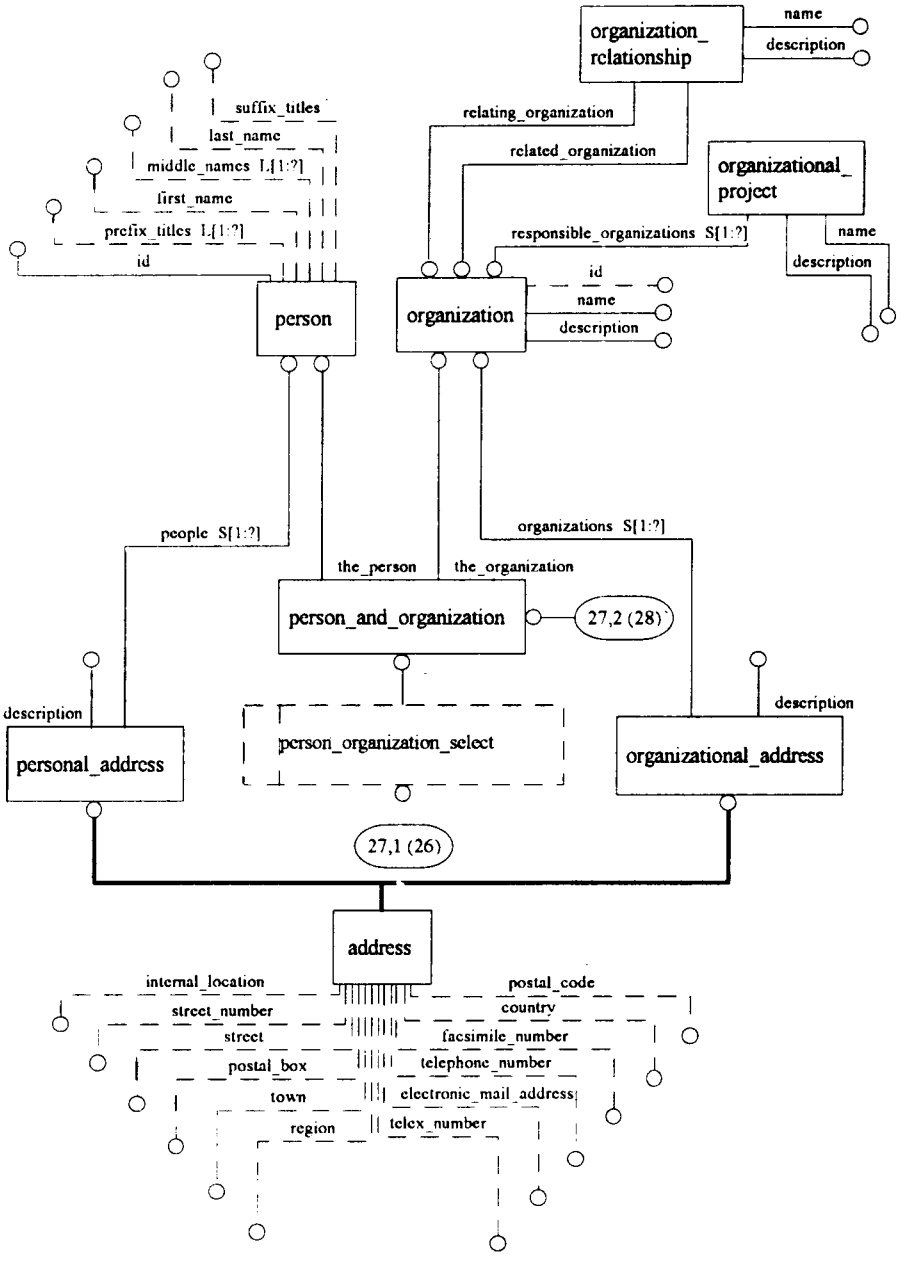


图 H27　人员和组织-应用解释模型(AIM)EXPRESS-G 图 39-27

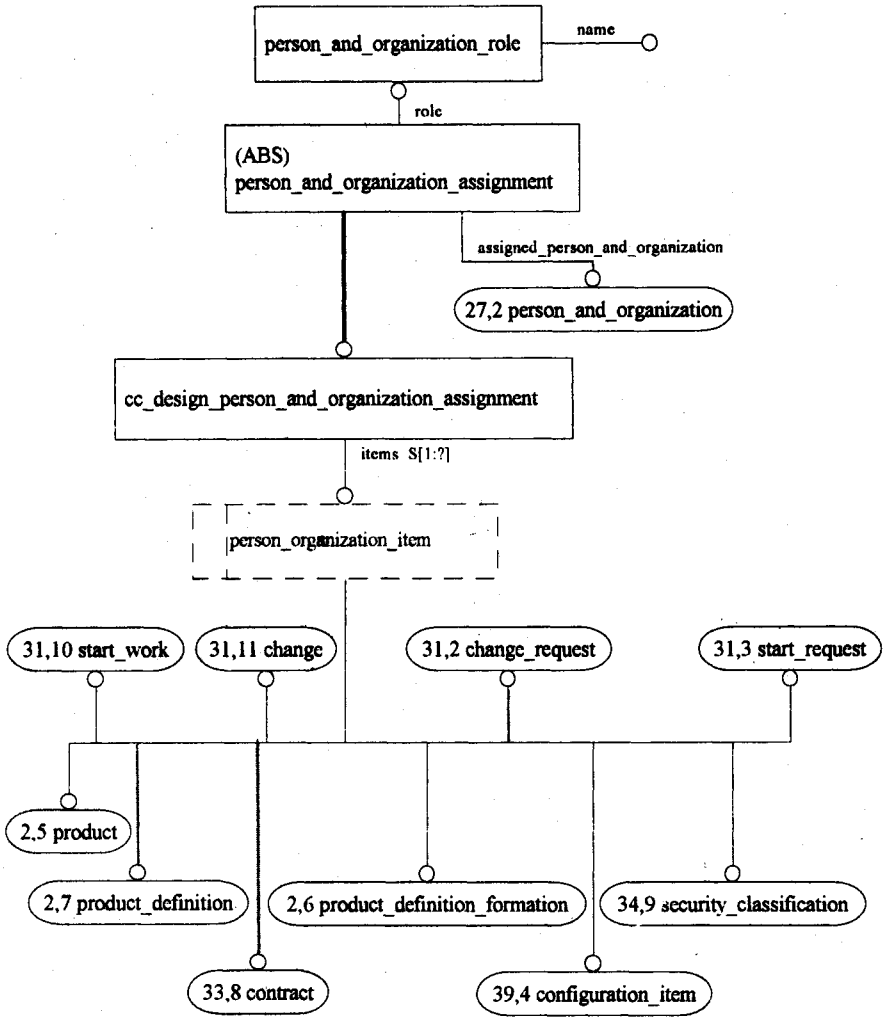


图 H28　人员和组织赋值-应用解释模型(AIM)EXPRESS-G 图 39-28

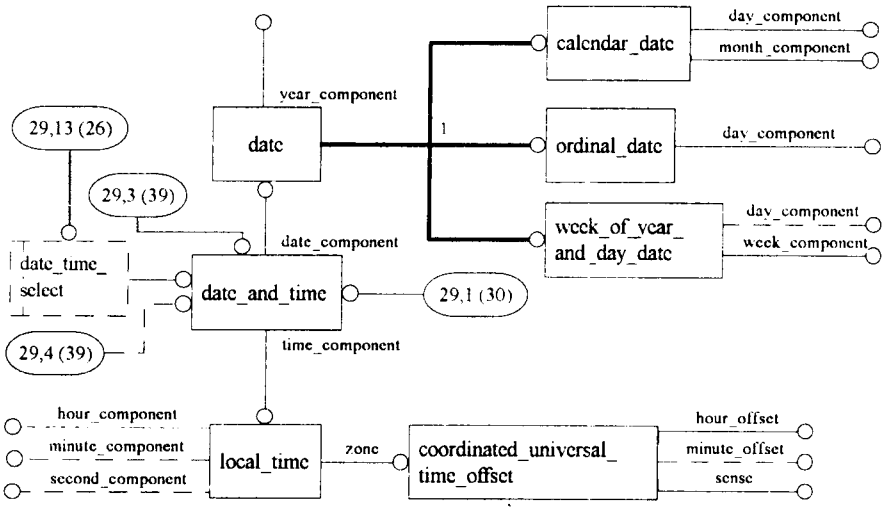


图 H29　日期和时间-应用解释模型(AIM)EXPRESS-G 图 39-29

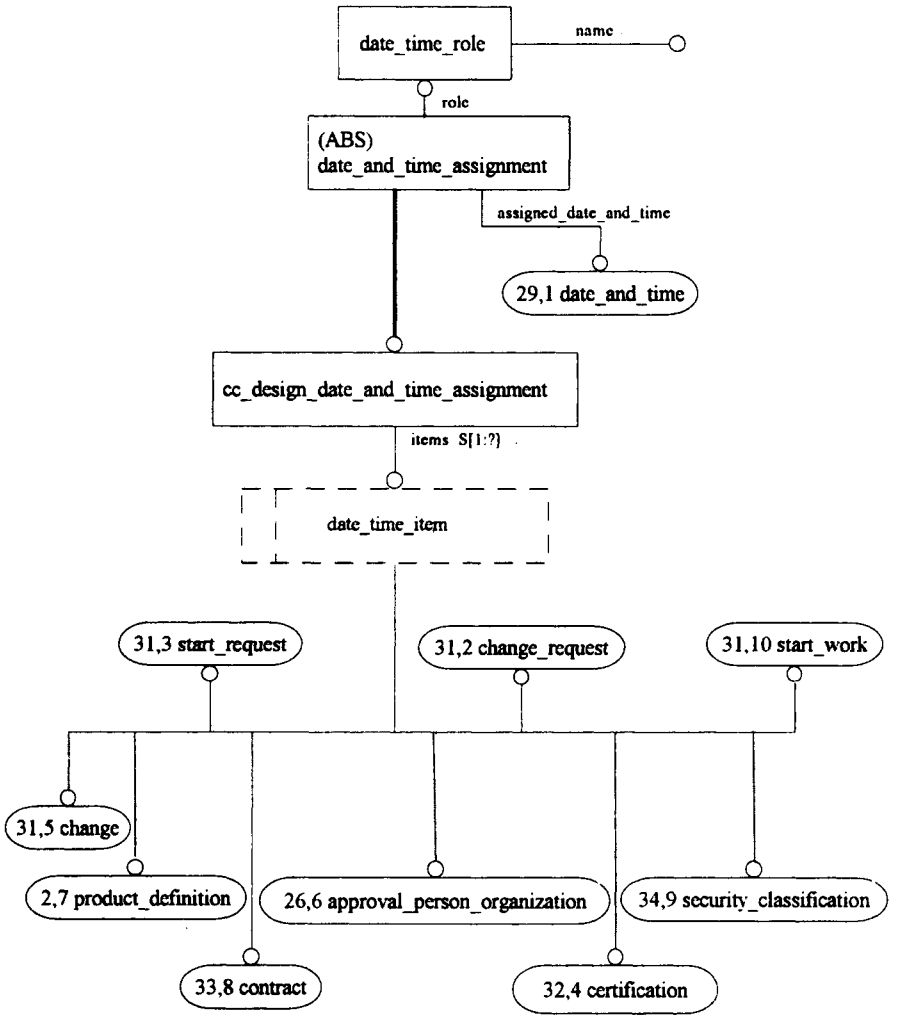


图 H30　日期和时间赋值-应用解释模型(AIM)EXPRESS-G 图 39-30

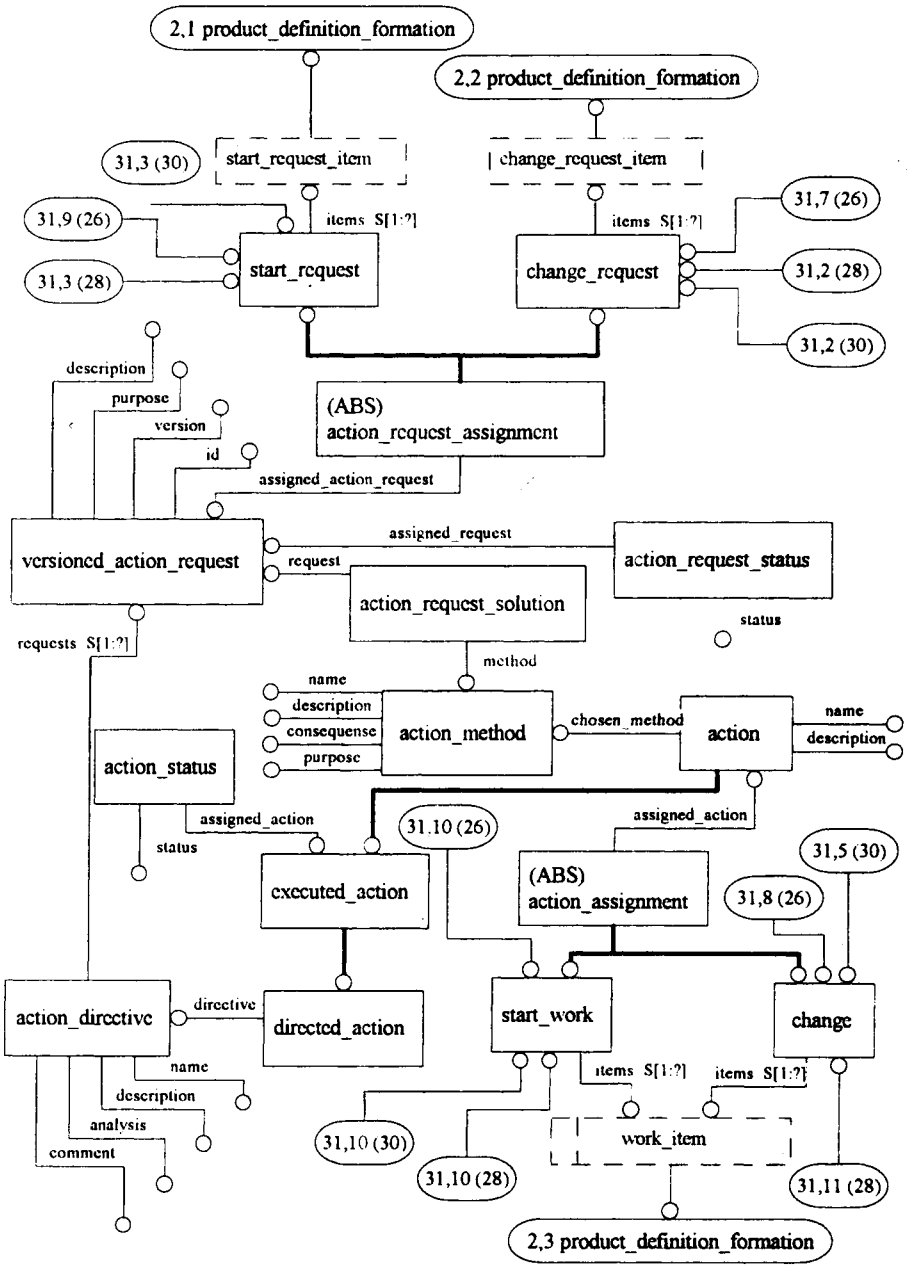


图 H31　工作和更改文档-应用解释模型(AIM)EXPRESS-G 图 39-31

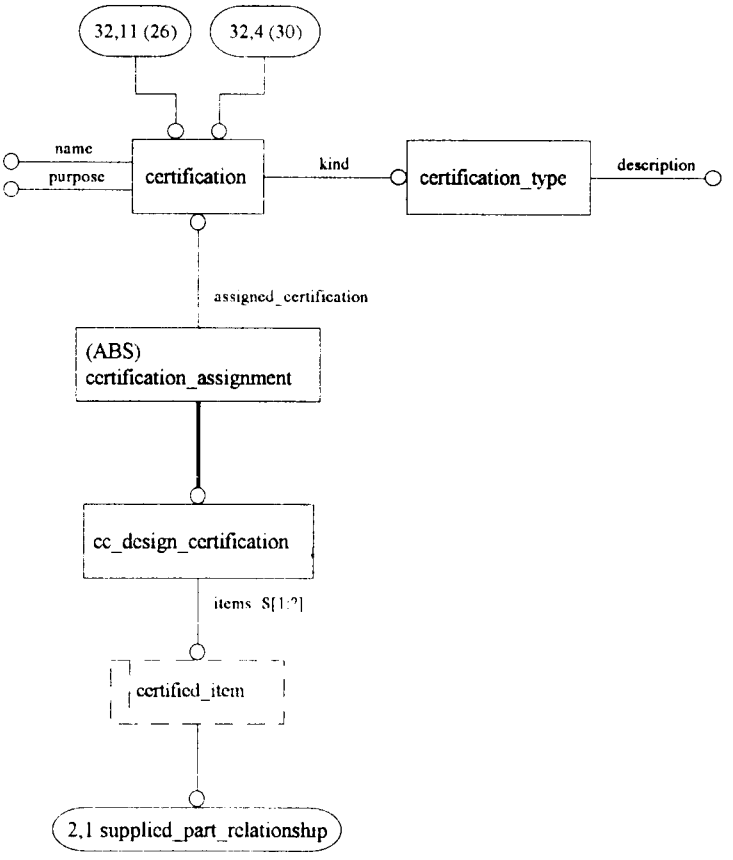


图 H32　认证-应用解释模型(AIM)EXPRESS-G 图 39-32

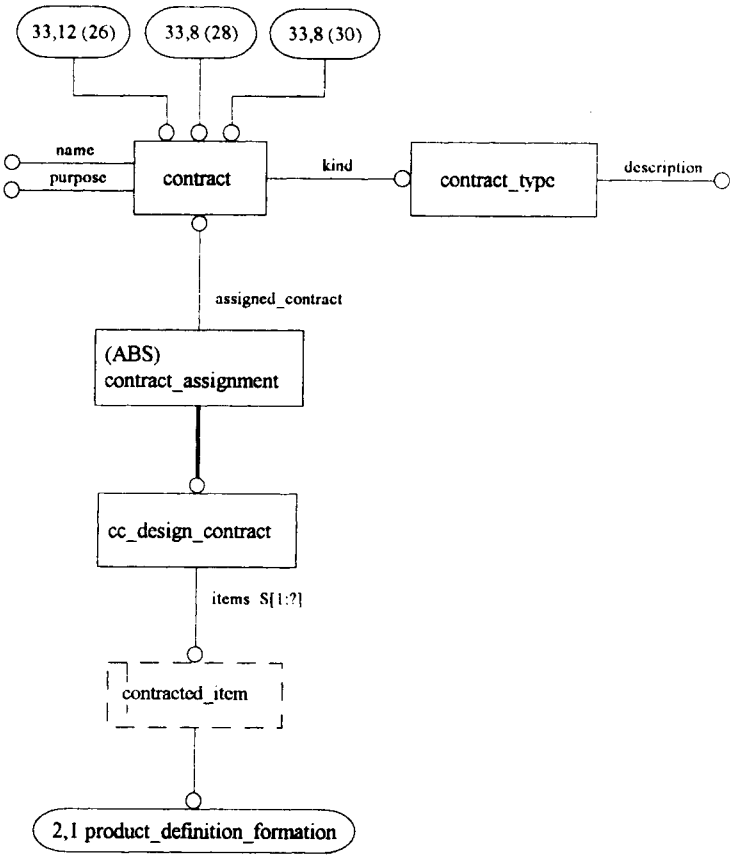


图 H33　合同-应用解释模型(AIM)EXPRESS-G 图 39-33

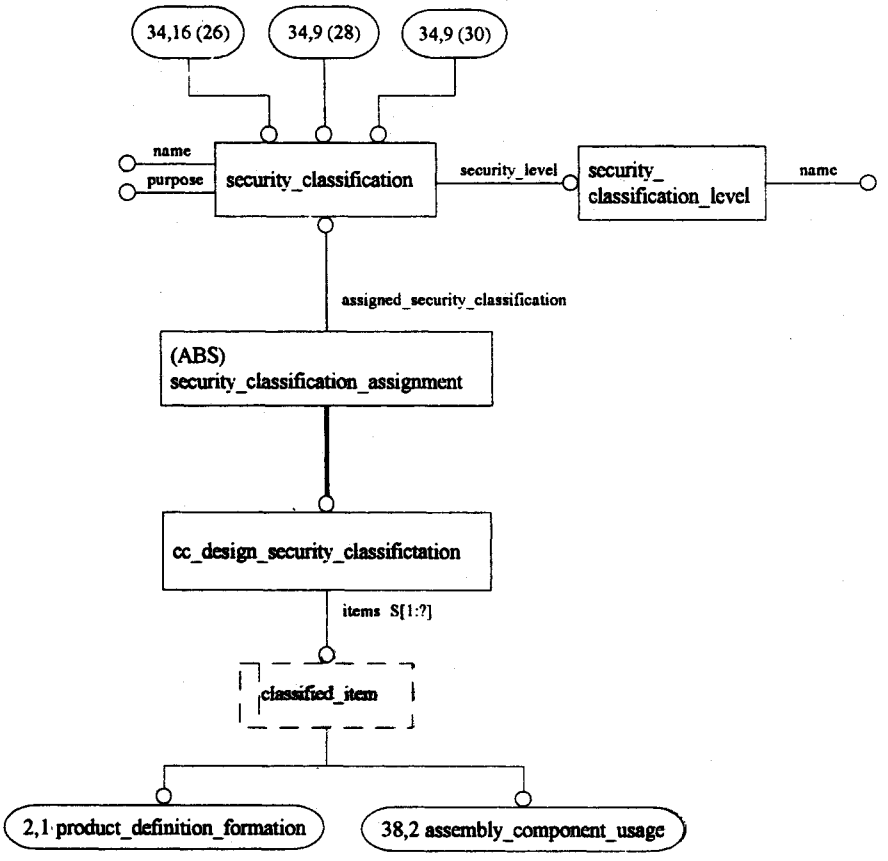


图 H34　安全分类-应用解释模型(AIM)EXPRESS-G 图 39-34

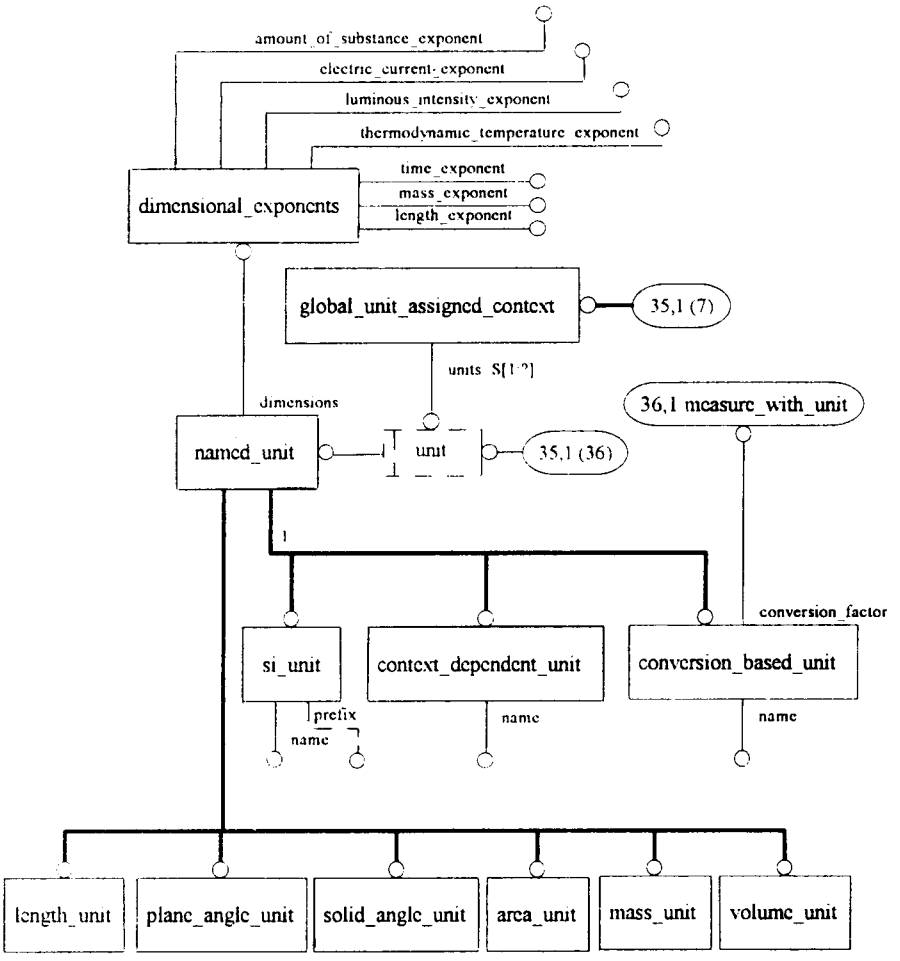


图 H35　单位-应用解释模型(AIM)EXPRESS-G 图 39-35

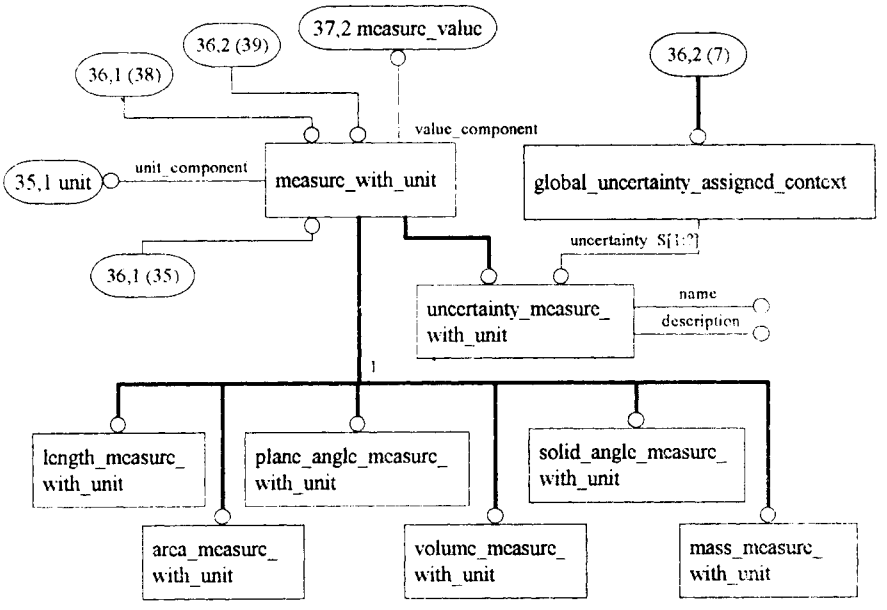


图 H36　有单位度量-应用解释模型(AIM)EXPRESS-G 图 39-36

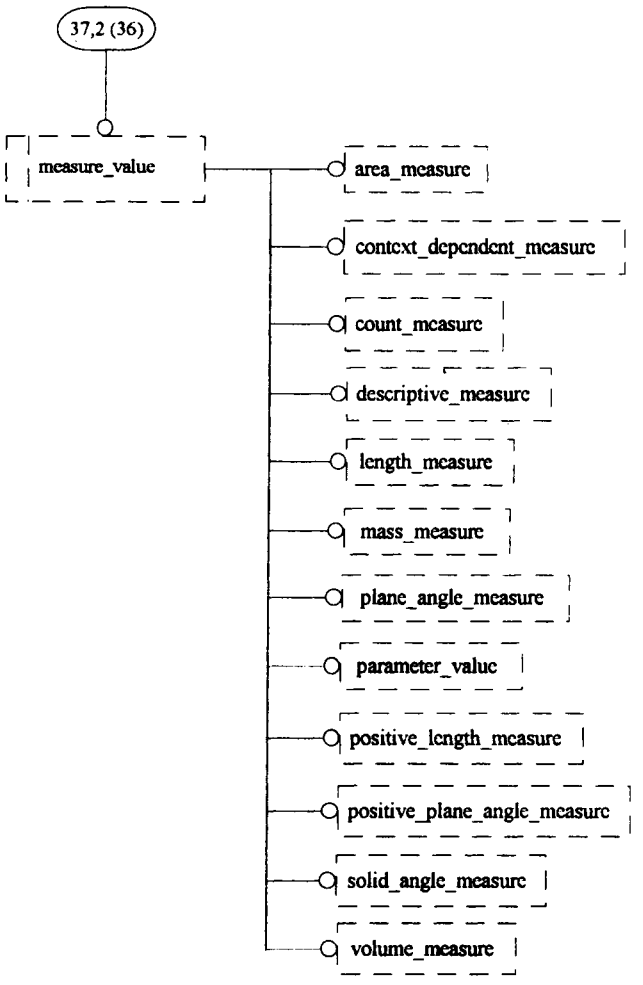


图 H37　度量-应用解释模型(AIM)EXPRESS-G 图 39-37

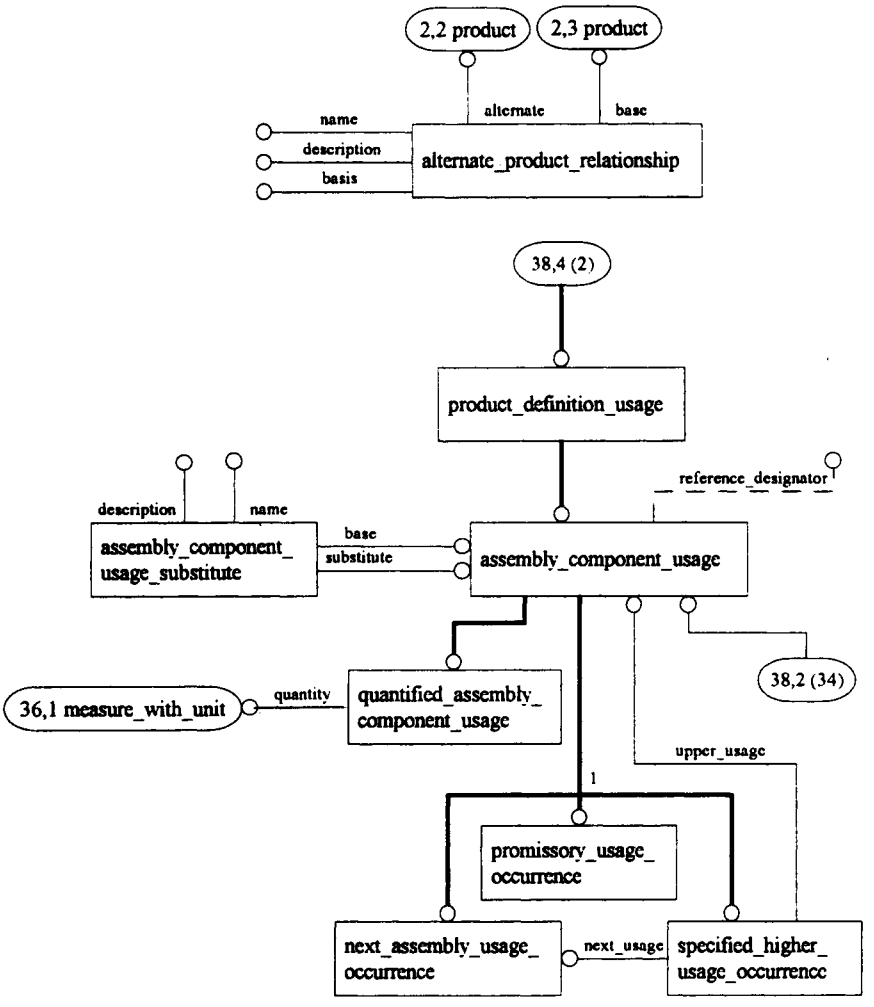


图 H38　产品结构-应用解释模型(AIM)EXPRESS-G 图 39-38

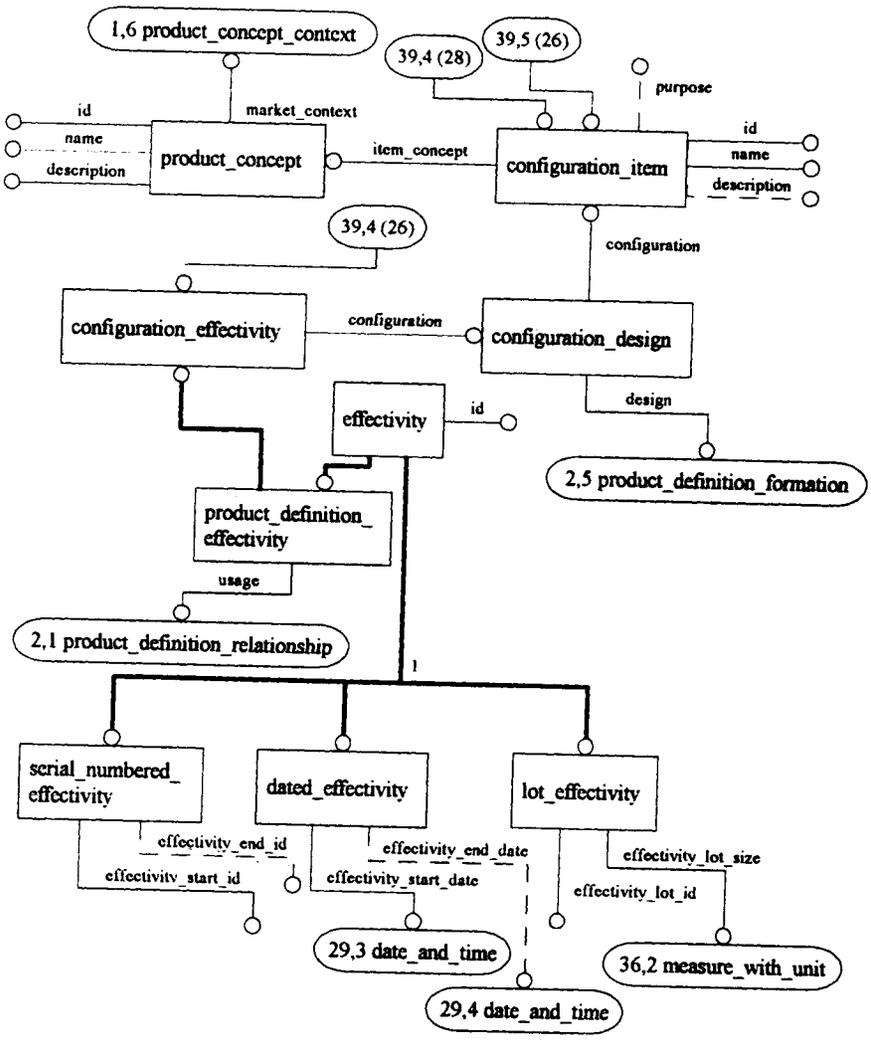


图 H39　配置-应用解释模型(AIM)EXPRESS-G 图 39-39

附 录 J
（提示的附录）
应用解释模型 EXPRESS 列表

本附录提供了本标准中的 EXPRESS 语言列表，其正像附录 A 中提供的那样，没有任何文字定义或其他注释。

本标准还包括一块 1.44MB DOS 格式化的磁盘，该磁盘上以数字形式提供了本附录的内容。该文件名为 ANNEX-J。该文件以 ASCII 码形式存储。

附 录 K
（提示的附录）
应用协议使用指南

K1 使用测试目的

下列条款提供了使用/功能测试目的。这些测试目的的意图是测试一实现的功能，该功能是一实现能够对兼有一些实体和属性的各复杂疑问提供正确结果的能力，以便获得对一个实际的大众问题的解答。

K1.1 有效性

下面列出了对有效性功能单元的使用测试目的：

——给定一个 product 标识，确定配置管理职能中各人员的名字和商业地址；

——给定一个 product_concept，确定各配置项，具有一些规定的 end_date 的一个 dated_effectivity 要应用到这些配置项；

——给定一(详细)零件的 product_definition 标识，对每个 configuration_design 确定该零件的量，serial_numbered_effectivity 要应用到它。

K1.2 截止项标识

下面列出了对 end_item_identification 功能单元的使用测试目的：

——对于一个给定的 product_concept，确定其关联的各 configuration_items，其提出的各 change requests 应用；

——对于一个给定的 product_concept，确定各零件是购买的而不是自己制造的；

——对于一个给定的 configuration_item，确定构件 product_definition 具有替代产品。

K1.3 物料清单

下面列出了对 bill_of_material 功能单元的使用测试目的：

——给定一个装配件 product_definition，确定哪个构件零件是标准件；

——给定一个 product 标识，确定最高层装配件等级的契约零件表，规定下一层较高层装配件的零件数量；

——给定一个装配件 product_definition，确定每个构件零件的释放状态；

——给定一个装配件 product_definition，确定每个构件零件的安全分类；

——给定一个 product_definition_formation，确定必须做 product_definition_formation 的合同以及确定合同职能中的相应部门；

——给定一个 product_definition，对于替代和规定的 product_definition 二者同时出现在该替代应用时，确定每个替代的 product_definition_formation 及引用标记。

K1.4 零件标识

下面列出了对 part_identification 功能单元的使用测试目的：

——给定一个 product_definition_formation，确定它的零件级别和可采用的 product_catagories；

——给定一个 product_definition，确定是否已经由其他零件设计了该零件标识及确定其基础零件，如果它是可用的，那么从基础零件导出它；

——给定一个 product_definition，确定在任何下一层装配中它每次使用的安全分类等级。

K1.5 设计信息

下面列出了对 design_information 功能单元的使用测试目的：

——给定一个 product_definition，确定在某些规定的下一层装配中该 product_definition 的表面粗糙度规范和相关的引用标记；

——给定一个 product_definition，确定可采用的材料牌号，材料级别和材料硬度。

K1.6 资源控制

下面列出了对 source_control 功能单元的使用测试目的：

——给定一个零件，确定它在某些 product_definition_formation 条件下是否能买到，并且如果它是可采用的，那么确定每个这种 product_definition_formation 的供应商的地址和标识。

K1.7 设计活动控制

下面列出了对 design_activity_control 功能单元的使用测试目的：

——给定一个 change_request，确定各合同，该各合同要求受了影响的 product_definition_formation；

——给定一个已批准的 change，确定各配置项，该各配置项要求受了影响的 product_definition_conformationd；

——给定一个 configuration_item，确定通过 change_request 受到影响的各构件零件；

——给定一个 change_request，确定请求接受职能中的人员、各请求状态、批准职能中的人员和各批准状态；

——给定一个批准的 change，确定由更改的零件带来冲击的各 effectivity。

K1.8 形状

下面列出了对 shape 功能单元的使用测试目的：

——给定一个详细零件 product_definition，确定在它的 shape_representation 中要求的几何和/或拓扑；

——给定一个装配件 product_definition，确定仅与该装配件零件的 shape_representation 相关的几何和/或拓扑；

——给定一个装配件 product_definition 和一些规定的坐标空间范围，正如由一些 shape_representation 描述的那样，确定在该范围中那个装配件的所有构件零件；

——给定一个 product_definition，确定其初步设计的 shape_representation；

——给定一个 shape_representation，确定所有 product_definition_formation，在它们的描述中它要求该 shape_representation；

——给定一个 product_definition，确定各规范，各规范应用于规定该零件的各 shape_aspect 或各范围；

——确定所有的 geometric_representation_items，其不参与任何 product_definition(部分)的形状表示(否定测试目的)。

K2 示例部分

下面是使用 GB/T 16656.21 中定义的格式来表达的一个示例部分。为了该注释和插图的目的，本

示例不使用来自附录B的各EXPRESS短名，并且具有长串数值的属性已经被断开在不同的线上。本部分遵守的模式是应用解释模型模式，它可在本标准的附录A中找到。

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('THIS IS A SAMPLE AP 203 STEP MODEL'),'1');
FILE_NAME('CONCEPTUAL PART EXAMPLE',
'1994-08-19 T15:30:00',
('LORI BRINDLE'),
('PDES,Inc.'),
'NO VERSION',
'HAND POPULATED',
'APPROVED BY LARRY MCKEE');
FILE_SCHEMA(('CONFIG_CONTROL_DESIGN'));
ENDSEC;
DATA;
#1=DIMENSIONAL_EXPONENTS(0.0,0.0,0.0,0.0,0.0,0.0,0.0);
#13=SI_UNIT(*,.CENTI.,.METRE.);
#14=(NAMED_UNIT(#1)PLANE_ANGLE_UNIT()SI_UNIT($,.RADIAN.));
#15=(NAMED_UNIT(#1)SI_UNIT($,.STERADIAN.)SOLID_ANGLE_UNIT());
#16=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.54),#13);
#17=DIMENSIONAL_EXPONENTS(1.0,0.0,0.0,0.0,0.0,0.0,0.0);
#18=(CONVERSION_BASED_UNIT('INCH',#16)LENGTH_UNIT()NAMED_UNIT(#17));
#19=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(0.000001),#13,'b-
rep precision','the value for tolerance of b-rep connectivity for underlying geometry');
#1000=CARTESIAN_POINT('cp1',(0.00000,0.00000,0.00000));
#1001=DIRECTION('dir1',(1.00000,0.00000.0.00000));
#1002=VECTOR('vec1',#1001,9.00000);
#1003=LINE('line1',#1000,#1002);
#1004=CARTESIAN_POINT('cp2',(9.00000,0.00000,0.00000));
#1005=DIRECTION('dir2',(0.00000,1.00000,0.00000));
#1006=VECTOR('vec2',#1005,6.00000);
#1007=LINE('line2',#1004,#1006);
#1008=CARTESIAN_POINT('cp3',(9.00000,6.00000,0.00000));
#1009=DIRECTION('dir3',(-1.00000,0.00000,0.00000));
#1010=VECTOR('vec3',#1009,9.00000);
#1011=LINE('line3',#1008,#1010);
#1012=CARTESIAN_POINT('cp4',(0.00000,6.00000,0.00000));
#1013=DIRECTION('dir4',(0.00000,-1.00000,0.00000));
#1014=VECTOR('vec4',#1013,6.00000);
#1015=LINE('line4',#1012,#1014);
#1016=CARTESIAN_POINT('cp5',(0.00000.0.00000,-0.25000));
#1017=VECTOR('vec5',#1001,9.00000);
#1018=LINE('line5',#1016,#1017);
```

#1019=CARTESIAN_POINT('cp6',(9.00000,0.00000,-0.25000));
#1020=VECTOR('vec6',#1005,6.00000);
#1021=LINE('line6',#1019,#1020);
#1022=CARTESIAN_POINT('cp7',(9.00000,6.00000,-0.25000));
#1023=VECTOR('vec7',#1009,9.00000);
#1024=LINE('line7',#1022,#1023);
#1025=CARTESIAN_POINT('cp8',(0.00000,6.00000,-0.25000));
#1026=VECTOR('vec8',#1013,6.00000);
#1027=LINE('line8',#1025,#1026);
#1028=DIRECTION('dir8',(0.00000,0.00000,-1.00000));
#1029=VECTOR('vec9',#1028,0.25000);
#1030=LINE('line9',#1000,#1029);
#1031=VECTOR('vec10',#1028,0.25000);
#1032=LINE('line10',#1004,#1031);
#1033=VECTOR('vec11',#1028,0.25000);
#1034=LINE('line12',#1012,#1033);
#1035=VECTOR('vec12',#1028,0.25000);
#1036=LINE('line13',#1008,#1035);
#1037=CARTESIAN_POINT('cp8',(1.62500,6.00000,-0.12500));
#1038=AXIS2_PLACEMENT_3D('ap1',#1037,#1005,#1001);
#1039=CIRCLE('cir1',#1038,0.06250);
#1040=CARTESIAN_POINT('cp9',(1.62500,5.87500,-0.12500));
#1041=AXIS2_PLACEMENT_3D('ap2',#1040,#1005,#1001);
#1042=CIRCLE('cir2',#1041,0.06250);
#1043=CARTESIAN_POINT('cp10',(7.37500,6.00000,-0.12500));
#1044=AXIS2_PLACEMENT_3D('ap2',#1043,#1005,#1001);
#1045=CIRCLE('cir3',#1044,0.06250);
#1046=CARTESIAN_POINT('cp11',(7.37500,5.87500,-0.12500));
#1047=AXIS2_PLACEMENT_3D('ap3',#1046,#1005,#1001);
#1048=CIRCLE('cir4',#1047,0.06250);
#1049=CARTESIAN_POINT('cp12',(0.75000,0.50000,0.00000));
#1050=DIRECTION ('dir10',(0.00000,0.00000,1.00000));
#1051=AXIS2_PLACEMENT_3D('ap4',#1049,#1050,#1001);
#1052=CIRCLE('cir5',#1051,0.25000);
#1053=CARTESIAN_POINT('cp13',(0.75000,0.50000,-0.25000));
#1054=AXIS2_PLACEMENT_3D('ap5',#1053,#1050,#1001);
#1055=CIRCLE('cir6',#1054,0.25000);
#1056=CARTESIAN_POINT('cp14',(8.25000,0.50000,0.00000));
#1057=AXIS2_PLACEMENT_3D('ap6',#1056,#1050,#1001);
#1058=CIRCLE('cir7',#1057,0.25000);
#1059=CARTESIAN_POINT('cp15',(8.25000,0.50000,-0.25000));
#1060=AXIS2_PLACEMENT_3D('ap7',#1059,#1050,#1001);
#1061=CIRCLE('cir8',#1060,0.25000);

```
#1062=CARTESIAN_POINT('cp16',(0.75000,5.50000,0.00000));
#1063=AXIS2_PLACEMENT_3D('ap8',#1062,#1050,#1001);
#1064=CIRCLE('cir9',#1063,0.25000);
#1065=CARTESIAN_POINT('cp17',(0.75000,5.50000,-0.25000));
#1066=AXIS2_PLACEMENT_3D('ap9',#1065,#1050,#1001);
#1067=CIRCLE('cir10',#1066,0.25000);
#1068=CARTESIAN_POINT('cp18',(8.25000,5.50000,0.00000));
#1069=AXIS2_PLACEMENT_3D('ap10',#1068,#1050,#1001);
#1070=CIRCLE('cir11',#1069,0.25000);
#1071=CARTESIAN_POINT('cp19',(8.25000,5.50000,-0.25000));
#1072=AXIS2_PLACEMENT_3D('ap11',#1071,#1050,#1001);
#1073=CIRCLE('cir12',#1072,0.25000);
#1074=AXIS2_PLACEMENT_3D('ap12',#1000,#1050,#1001);
#1078=COORDINATED_UNIVERSAL_TIME_OFFSET(5,$,.BEHIND.);
#1079=LOCAL_TIME(12,0,$,#1078);
#1080=TRIMMED_CURVE('tc1',#1003(0.0),(9.0),.T.,.PARAMETER.);
#1081=TRIMMED_CURVE('tc2',#1007(0.0),(6.0),.T.,.PARAMETER.);
#1082=TRIMMED_CURVE('tc3',#1011(0.0),(9.0),.T.,.PARAMETER.);
#1083=TRIMMED_CURVE('tc4',#1015(0.0),(6.0),.T.,.PARAMETER.);
#1084=TRIMMED_CURVE('tc5',#1018(0.0),(9.0),.T.,.PARAMETER.);
#1085=TRIMMED_CURVE('tc6',#1021(0.0),(6.0),.T.,.PARAMETER.);
#1086=TRIMMED_CURVE('tc7',#1024(0.0),(9.0),.T.,.PARAMETER.);
#1087=TRIMMED_CURVE('tc8',#1027(0.0),(6.0),.T.,.PARAMETER.);
#1088=TRIMMED_CURVE('tc9',#1030(0.0),(0.25),.T.,.PARAMETER.);
#1089=TRIMMED_CURVE('tc10',#1032(0.0),(0.25),.T.,.PARAMETER.);
#1090=TRIMMED_CURVE('tc11',#1034(0.0),(0.25),.T.,.PARAMETER.);
#1091=TRIMMED_CURVE('tc12',#1036(0.0),(0.25),.T.,.PARAMETER.);
#1099=(GEOMETRIC_REPRESENTATION_CONTEXT(3)
      GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT(#19)
      GLOBAL_UNIT_ASSIGNED_CONTEXT((#18,#14,#15))
      REPRESENTATION_CONTEXT('ID1','3D'));
#1100=GEOMETRIC_CURVE_SET('gcs1',(#1080,#1081,#1082,#1083,#1084,#1085,
      #1086,#1087,#1088,#1089,#1090,#1091,#1039,#1042,#1045,#1048,#1052,
      #1055,#1058,#1061,#1064,#1067,#1070,#1073));
#1101=GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION('gbwr1',
      (#1100,#1074),#1099);
#1102=APPLICATION_CONTEXT('CONFIGURATION MANAGEMENT');
#1103=APPLICATION_PROTOCOL_DEFINITION('COMMITTEE_DRAFT',
      'CONFIG_CONTROL_DESIGN',1994,#1102);
#1104=MECHANICAL_CONTEXT('CONFIGURATION MANAGEMENT',#1102,
'mechanical');
#1106=PRODUCT('2865000-1','REAR PANEL','REAR PANEL FOR BOX',(#1104));
#1107=PRODUCT_RELATED_PRODUCT_CATEGORY('detail','DETAIL PART',(#1106));
```

```
#1109=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE('-',
      'INITIAL RELEASE',#1106,.MADE.);
#1110=APPROVAL_STATUS('approved');
#1111=APPROVAL(#1110,'APPROVE PRODUCT VERSION');
#1112=CC_DESIGN_APPROVAL(#1111,(#1109));
#1113=PERSON('1111111','DOE','JOHN',('J'),$,$);
#1114=ORGANIZATION('99999','MYCOMPANY','WE BUILD PARTS');
#1115=PERSON_AND_ORGANIZATION(#1113,#1114);
#1118=APPROVAL_ROLE('VERSION APPROVAL');
#1119=APPROVAL_PERSON_ORGANIZATION(#1115,#1111,#1118);
#1120=CALENDAR_DATE(1994,2,1);
#1122=DATE_AND_TIME(#1120,#1079);
#1125=APPROVAL_DATE_TIME(#1122,#1111);
#1126=PERSON_AND_ORGANIZATION_ROLE('creator');
#1127=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#1115,#1126,
      (#1109));
#1128=SECURITY_CLASSIFICATION_LEVEL('unclassified');
#1129=SECURITY_CLASSIFICATION('CLASSIFICATION','CLASSIFY PRODUCT
      VERSION',
      #1128);
#1130=CC_DESIGN_SECURITY_CLASSIFICATION(#1129,(#1109));
#1131=APPROVAL_STATUS('approved');
#1132=APPROVAL(#1131,'APPROVE SECURITY CLASSIFICATION');
#1133=CC_DESIGN_APPROVAL(#1132,(#1129));
#1134=PERSON('2222222','DOE','JANE',('J'),$,$);
#1135=ORGANIZATION('88888','OURCOMPANY','WE CLASSIFY PARTS');
#1136=PERSON_AND_ORGANIZATION(#1134,#1135);
#1139=APPROVAL_ROLE('APPROVE SECURITY CLASSIFICATION');
#1140=APPROVAL_PERSON_ORGANIZATION(#1136,#1132,#1139);
#1141=CALENDAR_DATE(1994,2,2);
#1143=DATE_AND_TIME(#1141,#1079);
#1146=APPROVAL_DATE_TIME(#1143,#1132);
#1147=PERSON_AND_ORGANIZATION_ROLE('classification_officer');
#1148=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#1136,#1147,(#
1129));
#1149=CALENDAR_DATE(1994,2,1);
#1150=COORDINATED_UNIVERSAL_TIME_OFFSET(5,$,.BEHIND.);
#1151=LOCAL_TIME(12,0,$,#1150);
#1152=DATE_AND_TIME(#1149,#1151);
#1153=DATE_TIME_ROLE('classification_date');
#1154=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#1152,#1153,(#1129));
#1155=DESIGN_CONTEXT('PRODUCTION',#1102,'design');
#1156=PRODUCT_DEFINITION('PDID1','DEFINITION OF 2865000-1',#1109,#1155);
```

```
#1157=PRODUCT_DEFINITION_SHAPE('DESIGN SHAPE','SHAPE FOR 2865000-1',
      #1156);
#1158=SHAPE_DEFINITION_REPRESENTATION(#1157,#1101);
#1159=APPROVAL_STATUS('approved');
#1160=APPROVAL(#1159,'APPROVE PRODUCT DEFINITION');
#1161=CC_DESIGN_APPROVAL(#1160,(#1156));
#1162=PERSON('3333333','DOE','JIM',('J'),$,$);
#1163=ORGANIZATION('77777','YOURCOMPANY','WE DEFINE PARTS');
#1164=PERSON_AND_ORGANIZATION(#1162,#1163);
#1165=APPROVAL_ROLE('APPROVER');
#1166=APPROVAL_PERSON_ORGANIZATION(#1164,#1160,#1165);
#1170=CALENDAR_DATE(1994,2,1);
#1171=DATE_AND_TIME(#1170,#1079);
#1174=APPROVAL_DATE_TIME(#1171,#1160);
#1175=PERSON_AND_ORGANIZATION_ROLE('creator');
#1176=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#1164,#1175,
      (#1156));
#1177=CALENDAR_DATE(1994,2,1);
#1178=COORDINATED_UNIVERSAL_TIME_OFFSET(5,$,.BEHIND.);
#1179=LOCAL_TIME(12,0,$,#1178);
#1180=DATE_AND_TIME(#1177,#1179);
#1181=DATE_TIME_ROLE('creation_date');
#1182=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#1180,#1181,(#1156));
#1190=PERSON_AND_ORGANIZATION_ROLE('design_owner');
#1191=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#1115,#1190,
      (#1106));
#1192=PERSON_AND_ORGANIZATION_ROLE('design_supplier');
#1193=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#1115,#1192,
      (#1109));
ENDSEC;
END-ISO-10303-21;
```

附 录 L
（提示的附录）
参 考 文 献

[1] *Guidelines for the Development and Approval of STEP Application Protocols, Version 1.1*, ISO TC184/SC4/WG4 N66, January 1993.

[2] Marks' Standard Handbook for Mechanical Engineers, edited by Avallone and Baumeister, Ninth Edition, McGraw-Hill, 1987.

[3] *IDEF0(ICAM Definition Language 0)*, Federal Information Processing Standards Publication 183, Integration Definition for Function Modeling(IDEF0), FIPS PUB 183, National Institute of Standards and Technology, December 1993.

[4] *IDEF1X (ICAM Definition Language 1 Extended)*, Federal Information Processing Standards Publication 184, Integration Definition for Information Modeling (IDEF1X), FIPS PUB 184, National Institute of Standards and Technology, December 1993.

实体、属性名中英文对照

AAM:应用活动模型(application activity model)
abstract test suite(抽象测试套件)
action(活动)
action assignment(活动赋值)
action directive(活动命令)
action directive role(活动命令职能)
action execution(活动执行)
action method(活动方法)
action request assignment(活动请求赋值)
action request solution(活动请求方案)
action request status(活动请求状态)
action status(活动状态)
acu requires security classification(装配构件使用需要安全分类)
additional data(附加数据)
Additional design information(附加设计信息)
address(地址)
adopted solution(采用方案)
Advanced B rep(高级的 B rep)
advanced boundary representation(高级的边界表示)
advanced brep shape representation(高级的边界形状表示)
advanced face(高级的表面)
AIM:应用解释模型(application interpreted model)
Alternate part(备用零件)
alternate product relationship(备用产品关系)
analysis data(分析数据)
ancestor node(前辈节点)
AP:应用协议(application protocol)
application(应用)
application activity model(应用活动模型)
application context(应用相关环境)
application interpreted model(应用解释模型)
application object(应用对象)
application protocol(应用协议)
application reference model(应用参考模型)
application context(应用相关环境)
application context element(应用相关环境元素)
application context requires ap definition(应用相关环境需要应用协议定义)
application protocol definition(应用协议定义)

Approval(批准)
approval_assignment(批准赋值)
approval_date_time(批准日期时间)
approval_item(批准项)
approval_person_organization(批准人员组织)
approval_relationship(批准关系)
approval_requires_approval_date_time(批准需要批准日期时间)
approval_requires_approval_person_organization(批准需要批准人员组织)
approval_role(批准职能)
approval_status(批准状态)
approvals_are_assigned(批准被赋值)
approved(已批准)
arcwise connected(弧连通的)
area_measure_with_unit(面积度量)
area_unit(面积单位)
ARM:应用参考模型(application reference model)
as_required(按照要求)
as_required(按所需)
as_required_quantity(按所需数量)
assembly(装配件)
assembly(装配)
assembly_component_usage(装配构件使用)
assembly_component_usage_substitute(装配构件使用代用)
assembly_shape_is_defined(被定义的装配件形状)
assembly_shape_is_defined(装配形状被定义)
assigned_approval(赋值批准)
associated_geometrys(缔合几和)
authorization(授权)
authorized_approval(授权批准)
axis 1_placement(轴 1 方位)
axis 2_placement_2d(二维空间轴 2 方位)
axis 2_placement_3d(三维空间轴 2 方位)
axisymmetric(轴对称)
b_spline_curve(B 样条曲线)
b_spline_curve_with_knots(有结点的 B 样条曲线)
b_spline_surface(B 样条曲面)
b_spline_surface_with_knots(有结点的 B 样条曲面)
bag_to_set(袋-集合-转换)
basis_curve_check(基本曲线检查)
basis_surface_check(基本曲面检查)
bezier_curve(Bezier 曲线)
bezier_suface(Bezier 曲面)

bill_of_material(物料清单)
billofmaterial structure(物料清单结构)
BOM:物料清单(billofmaterial)
boundary(边界)
boundary representation model(边界表示模型)
boundary representation solid model(边界表示实体模型)
boundary_curve(边界曲线)
bounded_curve(有界曲线)
bounded_surface(有界曲面)
bounds(界限)
brep_with_voids(有空洞的边界表示)
CAD:计算机辅助设计(computer aided design)
CAD_filename(CAD 文件名)
calendar_date(日历日期)
cartesian_point(笛卡尔点)
cartesian_transformation_operator(笛卡尔变换运算器)
cartesian_transformation_operator_2d(二维笛卡尔变换运算器)
cartesian_transformation_operator_3d(三维笛卡尔变换运算器)
cast(铸造)
cc_design_approval(配置控制设计批准)
cc_design_certification(配置控制设计认证)
cc_design_contract(配置控制设计合同)
cc_design_date_and_time_assignment(配置控制设计日期和时间赋值)
cc_design_date_time_correlation(配置控制设计日期和时间相互关系)
cc_design_person_and_organization_assignment(配置控制设计人员和组织赋值)
cc_design_person_and_organization_correlation(配置控制设计人员和组织相互关系)
cc_design_security_classification(配置控制设计安全分类)
cc_design_specification_reference(配置控制设计规范引用)
cc_design_approval(配置控制设计批准)
cc_design_certification(配置控制设计认证)
cc_design_contract(配置控制设计合同)
cc_design_date_and_time_assignment(配置控制设计日期和时间赋值)
cc_design_person_and_organization_assignment(配置控制设计人员和组织赋值)
cc_design_security_classification(配置控制设计安全分类)
cc_design_specification_reference(配置控制设计规范引用)
certification(认证)
certification_assignment(认证赋值)
certification_date(认证日期)
certification_required(所需认证)
certification_requires_approval(认证需要批准)
certification_requires_date_time(认证需要日期时间)
certification_type(认证类型)

certified_item(认证项)
change(更改)
change_date(更改日期)
Change_order(更改单)
Change_request(更改请求)
change_request_items(更改请求项)
change_request_requires_approval(更改请求需要批准)
change_request_requires_date_time(更改请求需要日期时间)
change_request_requires_person_organization(更改请求需要人员组织)
change_requires_approval(更改需要批准)
change_requires_date_time(更改需要日期时间)
child node(子节点)
circle(圆)
classification_date(分类日期)
classification_officer(分类员)
classified(有级别)
classified_item(分类项)
closed curve(闭曲线)
closed surface(闭曲面)
closed_shell(封闭壳)
CM:配置控制 (configuration management)
coined (高压铸造)
component (构件)
component_assembly_position (构件装配位置)
component_quantity (构件数量)
component_quantity (构件数量)
composite_curve (复合曲线)
composite_curve_on_surface (曲面上的复合曲线)
composite_curve_segment (复合曲线段)
confidential (机密)
configuration_effectivity (配置有效性)
configuration_design (配置设计)
configuration_effectivity (配置有效性)
configuration_item (配置项)
configuration_item_requires_approval (配置项需要批准)
configuration_item_requires_person_organization (配置项需要人员组织)
configuration_manager (配置管理员)
conformance class (一致性级别)
conformance requirement (一致性需求)
conformance testing (一致性测试)
conic (圆锥曲线)
conical_surface (圆锥面)

connected（连通的）
connected component（连通部分）
connected edge set（连接边集）
connected face set（连接面集合）
consequence（结果）
context dependent shape representation（相关环境依赖的形状表示）
context dependent unit（相关环境依赖单位）
contract（合同）
contract assignment（合同赋值）
contract date（合同日期）
contract number（合同号）
contract requires approval（合同需要批准）
contract requires person organization（合同需要人员组织）
contract type（合同类型）
contracted item（合同项）
contractor（合同员）
conversion based unit（换算单位）
coordinate space（坐标空间）
coordinated assembly and shape（协调的装配和形装）
coordinated universal time offset（协调的世界时差）
cost plus(成本加价)
er eator（创建者）
creation date（开发日期）
creation date（创建日期）
creator（创建者）
curve（曲线）
curve 3d（三维曲线）
curve bounded surface（曲线界定的曲面）
curve replica（曲线复制）
customer furnished equipment（配备设备的用户）
cycle（循环）
cylindrical surface（圆柱面）
design owner（设计拥有者）
DAG：有向非循环图（directed acyclic graph）
data（数据）
data exchange（数据交换）
Date（日期）
date and time（日期和时间）
date and time assignment（日期和时间赋值）
date and time assignment（日期和时间赋值）
date and time role（日期时间职能）
date time item(日期时间项)

date_time_item(日期时间项)
date_time_role(日期时间职能)
dated_approval (日期批准)
dated_effectivity (日期有效性)
declassification_date (取消分类日期)
definition (定义)
definitional_representation (定义表达)
degenerate_pcurve (退化参数域曲线)
dependent_instantiable_action_directive (依赖可实例化活动命令)
dependent_instantiable_approval_status (依赖可实例化批准状态)
dependent_instantiable_certification_type (依赖可实例化认证类型)
dependent_instantiable_contract_type (依赖可实例化合同类型)
dependent_instantiable_date (依赖可实例化日期)
dependent_instantiable_date_time_role (依赖可实例化日期时间职能)
dependent_instantiable_document_type (依赖可实例化文档类型)
dependent_instantiable_named_unit (依赖可实例化命名单位)
dependent_instantiable_parametric_representation_context (依赖可实例化参数表达相关环境)
dependent_instantiable_person_and_organization_role (依赖可实例化人员和组织职能)
dependent_instantiable_representation_item (依赖可实例化表示项)
dependent_instantiable_security_classification_level (依赖可实例化安全分类等级)
dependent_instantiable_shape_representation (依赖可实例化形状表示)
descendent node (后辈节点)
description (描述)
descriptive_measure (描述度量)
design phase (设计阶段)
design_activity_control (设计活动控制)
design_context (设计相关环境)
design_context_for_property (设计相关环境目的)
design_context_for_property (性能设计相关环境)
Design_discipline_product_definition (设计科目产品定义)
design_information (设计信息)
design_make_from_relationship (设计参照关系)
design_owner (设计拥有者)
design_specification (设计规范)
design_supplier (设计提供者)
detail (细目)
dimension_of (维数)
dimensional_exponents (量纲指数)
dimensionality (维数)
directed acyclic graph (指定非循环图)
directed_action (指定活动)
direction (方向)

disapproved（不批准）
discipline_id（科目标识）
dmanifold with boundary（有边界的 d 维流形）
document（文档）
document_reference（文档引用）
document_relationship（文档关系）
document_to_product_definition（文档与产品定义）
document_type（文档类型）
document_usage_constraint（文档使用约束）
document_with_class（有级别文档）
domain（域）
drawn（拉伸）
edge（边）
edge_based_wireframe_model（基于边的线框模型）
edge_based_wireframe_representation（基于边的线框表示）
edge_based_wireframe_shape_representation（基于边的线框形状表示）
edge_curve（边曲线）
edge_loop（边环）
effectivity（有效性）
effectivity_requires_approval（有效性需要批准）
element（元素）
elementary_surface（初等曲面）
ellipse（椭圆）
end_date（截止日期）
end_item_identification（截止项标识）
Engineering_assembly（工程装配）
Engineering_make_from（工程参照件）
Engineering_next_higher_assembly（工程下一层高级装配）
Engineering_promissory_usage（工程约定使用）
evaluated_degenerate_pcurve（赋值的退化参数域曲线）
executed_action（完成的活动）
extent（量度）
extruded（挤压）
face（表面）
face_bound（面界）
face_outer_bound（表面外部界限）
face_surface（表面曲面）
faceted_brep（小平面边界表示）
faceted_brep_shape_representation（小平面边界表示形状表示）
Facetted_B_rep（小平面 Brep）
facetted_boundary_representation（小平面边界表示）
FALSE（假）

finite（有限的）
fixed_price（固定价格）
forged（锻造）
form，fit and function of a product（产品的形状，配合与功能）
formed（锻压）
from_effectivity_id（有效性起始标识）
functionally_defined_transformation（功能上定义的变换）
gbsf_check_curve（几何有界曲面检查曲线）
gbsf_check_point（几何有界曲面检查点）
gbsf_check_surface（几何有界曲面检查曲面）
geometric coordinate system（几何坐标系）
geometric_curve_set（几何曲线集合）
Geometric_model_representation（几何模型表示）
geometric_representation_context（几何表示相关环境）
geometric_representation_item（几何表示项）
geometric_representation_item_3d（三维几何表示项）
geometric_set（几何集合）
geometrically founded（基于几何的）
geometrically related（几何相关的）
geometrically_bounded_surface_shape_representation（几何有界的曲面形状表示）
geometrically_bounded_wireframe_shape_representation（几何有界的线框形状表示）
global_uncertainty_assigned_context（全局带不确定性的相关环境）
global_unit_assigned_context（全局单位赋值相关环境）
global_unit_assigned_context（全局单位相关环境）
global_unit_assignment（全局单位赋值）
graph（图）
handle（句柄）
hold（扣留）
homeomorphic（同胚）
hyperbola（双曲线）
ICAM：集成计算机辅助制造（integrated computeraided manufacturing）
ID：标识（identification）
IDEF0：集成计算机辅助制造定义语言 0(ICAM definition language 0)
IDEF1X：集成计算机辅助制造定义语言 1 扩展(ICAM definition language lextended)
implementation method（实现方法）
in_work（在工作中）
information（信息）
initiator（发起人）
inseparable_assembly（不可拆装配）
integrated resource（集成资源）
interpretation（解释）
intersection_curve（交线）

issued（发布）
item_concept（项概念）
item_defined_transformation（项定义变换）
item_id（标识项）
items（各项）
leaf node（叶节点）
length_measure_with_unit（长度度量）
length_unit（长度单位）
line（直线）
link（链）
list（表）
local_time（当地时间）
loop（环）
lot（批）
lot_effectivity（批量有效性）
lot_number（批量号）
lot_size（批量大小）
lot_size_unit_of_measure（批量单位）
machanical part（机械零件）
machined（机加工）
make_or_buy_code（制造或购买代码）
manifold_solid_brep（流形实体边界表示）
manifold_surface_shape_representation（流形曲面形状表示）
manifold_surface_with_topology（拓扑流形曲面）
mapped_item（映射项）
mass_measure_with_unit（质量度量）
mass_unit（质量单位）
Material_specification（材料规范）
measure_with_unit（度量）
mechanical_context（机械相关环境）
model_name（模型名）
molded（模塑）
msb_shells（各流形实体表示壳）
name（名）
named_unit（命名单位）
next_assembly_usage_occurrence（下一层装配活动）
no_shape_for_make_from（无参照形状）
no_shape_for_supplied_part（无供应零件形状）
node（节点）
Non_topological_surface_and_wireframe（无拓扑曲面与线框）
not_yet_approved（尚未批准）
Of_product（产品的）

offset_curve_2d(二维偏置曲线)
offset_curve_3d(三维偏置曲线)
offset_surface (偏置曲面)
open_curve (开曲线)
open_surface (开曲面)
ordered_action (指令活动)
ordinal_date (顺序日期)
organization (组织)
organization_project (组织项目)
organization_relationship (组织关系)
organizational_address (组织的地址)
orientable (可定向的)
oriented_closed_shell (有向闭式壳)
oriented_edge (有向边)
oriented_face (有向表面)
oriented_open_shell (有向开式壳)
oriented_path (有向路径)
overlap (重叠)
parabola (抛物线)
parameter_range (参数范围)
parameter_space (参数域)
parametric_representation_context (参数表示相关环境)
parent_node (父节点)
parent_child (父子)
parent_curve (父曲线)
parent_surface (父曲面)
Part (零件)
part_classification (零件分类)
part_identification (零件标识)
part_nomenclature (零件名称)
part_number (零件号)
part_supplier (零件供应商)
part_type (零件类型)
Part_version (零件版本)
parts list strucutre (零件表结构)
path (路径)
pcurve (参数域曲线)
person (人员)
person_and_organization (人员和组织)
person_and_organization_assignment (人员和组织赋值)
person_and_organization_role (人员和组织的职能)
Person_organization (人员组织)

person_organization_id（人员组织标识）
person_organization_item（人员组织项）
personal_address（人员的地址）
phase_of_product（产品阶段）
PICS proforma（协议实现一致性声明问卷）
PICS：协议资料和一致性声明（Protocol information and conformance statement）
placement coordinate system（定位坐标系）
placement（定位）
plane（平面）
plane_angle_measure_with_unit（平面角度度量）
plane_angle_unit（平面角度单位）
Planned_date_effectivity（计划日期有效性）
Planned_effectivity（计划有效性）
Planned_lot_effectivity（计划批量有效性）
Planned_sequence_effectivity（计划工序有效性）
point（点）
point_on_curve（曲线上的点）
point_on_surface（曲面上的点）
point_replica（点复制）
poly_loop（多环）
polyline（折线）
postprocessor（后置处理程序）
preprocessor（前置处理程序）
Process_specification（工艺规范）
product（产品）
product data（产品数据）
product_category（产品目录）
product_category_relationship（产品目录关系）
product_concept（产品概念）
product_concept_context（产品概念相关环境）
product_concept_requires_configuration_item（产品概念需要配置项）
Product_configuration（产品配置）
product_context（产品相关环境）
product_data_type（产品数据类型）
product_definition（产品定义）
product_definition_context（产品定义相关环境）
product_definition_effectivity（产品定义有效性）
product_definition_formation（产品定义形成）
product_definition_formation_with_specified_source（有规定源的产品定义形成）
product_definition_relationship（产品定义关系）
product_definition_requires_approval（产品定义需要批准）
product_definition_requires_date_time（产品定义需要日期时间）

product_definition_requires_person_organization（产品定义需要人员组织）
product_definition_shape（产品定义形状）
product_definition_usage（产品定义使用）
product_definition_with_associated_documents（有相关文档的产品定义）
Product_model（产品模型）
product_related_product_category（有关产品的产品目录）
product_requires_person_organization（产品需要人员组织）
product_requires_product_category（产品需要产品目录）
product_requires_version（产品需要版本）
product_version_requires_approval（产品版本需要批准）
product_version_requires_person_organization（产品版本需要人员组织）
product_version_requires_security_classification（产品版本需要安全分类）
promissory use（约定用法）
promissory_usage_occurance（约定的活动）
property_definition（性能定义）
proposed（提出）
proprietary（专有的）
protocol implementation conformance statement（协议实现一致性声明）
Purpose（目的）
quantified_assembly_component_usage（合格的装配构件使用）
quantity_unit_of_measure（数量度量单位）
quasi_uniform_curve（准均匀曲线）
quasi_uniform_surface（准均匀曲面）
rational_b_spline_curve（有理 B 样条曲线）
rational_b_spline_surface（有理 B 样条曲面）
reason（理由）
recommended_solution（推荐方案）
rectangular_composite_surface（矩形复合曲面）
rectangular_trimmed_surface（矩形剪裁曲面）
reference_designator（引用标志）
reference_to_curve（引用曲线）
related_product_definition（关联的产品定义）
relating_document（关联文档）
release_date（发放日期）
release_status（发放状态）
reparameterized_composite_curve_segment（重新参数化的复合曲线段）
representation（表示）
representation_context（表示相关环境）
representation_item（表示项）
representation_map（表示映射）
representation_relationship（表示关系）
representation_relationship_with_transformation（有变换的表示关系）

request_date（请求日期）
request_recipient（请求接收者）
requested_action（请求活动）
restrict_action_request_status（限制活动请求状态）
restrict_approval_status（限制批准状态）
restrict_certification_type（限制认证类型）
restrict_contract_type（限制合同类型）
restrict_date_time_role（限制日期时间职能）
restrict_document_type（限制文档类型）
restrict_person_organization_role（限制人员组织职能）
restrict_product_category_value（限制产品目录值）
restrict_security_classification_level（限制安全分类等级）
revision_letter（修改文字）
role（职能）
rolled（轧制）
root node（根节点）
seam_curve（接合面曲线）
secret（秘密）
security_classification（安全分类）
security_classification_assignment（安全分类赋值）
security_classification_level（安全分类等级）
security_classification_optional_date_time（安全分类可选的日期时间）
security_classification_requires_approval（安全分类需要批准）
security_classification_requires_date_time（安全分类需要日期时间）
security_classification_requires_person_organization（安全分类需要人员组织）
security_code（安全代码）
SELF_definition_relation.related_product_definition（自定义关系。有关的产品定义）
SELF_definition_relation.relating_product_definition（自定义关系。关联产品定义）
self_intersect（自交）
selfintersect（自交）
selfloop（封闭环）
serial_numbered_effectivity（顺序有效性）
serial_numbered_effectivity（连续编号有效性）
set（集合）
Shape（形状）
Shape_aspect（形状特征）
shape_aspect_relationship（形状特征关系）
shape_definition_representation（形状定义表示）
shape_representation（形状表示）
shape_representation_relationship（形状表示关系）
sheared（剪切）
shell_based_surface_model（基于壳的曲面模型）

shell_based_wireframe_model（基于壳的线框模型）
shell_based_wireframe_representation（基于壳的线框表示）
shell_based_wireframe_shape_representation（基于壳的线框形状表示）
si_unit（公制单位）：
sign_off_date（签发日期）
solid_model（实体模型）
solid_angle_measure_with_unit（实体角度度量）
solid_angle_unit（实体角度单位）
solid_model（实体模型）
source_control（资源控制）
space dimensionality（空间维数）
Specification（规范）
specification_code（规范代码）
specification_reference（规范引用）
specification_source（规范源）
specified_higher_usage_occurrence（规定的更高层活动）
specified_item（规定项）
spherical_surface（球形曲面）
standard_part_indicator（标准件标志）
start_date（开始日期）
Start_order（开始单）
start_request（开始请求）
start_request_item（开始请求项）
start_request_requires_approval（开始请求需要批准）
start_request_requires_date_time（开始请求需要日期时间）
start_request_requires_person_organization（开始请求需要人员组织）
start_work（开始工作）
start_work_requires_approval（开始工作需要批准）
start_work_requires_date_time（开始工作需要日期时间）
status（状态）
STRING（串）
structure（结构）
subassembly（子装配件）
Substitute_part（代用零件）
subtype_mandatory_action（子类型强制的活动）
subtype_mandatory_effectivity（子类型强制的有效性）
subtype_mandatory_product_context（子类型强制的产品相关环境）
subtype_mandatory_product_definition_formation（子类型强制的产品定义功能）
subtype_mandatory_product_definition_formation（子类型强制的产品定义形成）
subtype_mandatory_product_definition_usage（子类型强制的产品定义使用）
subtype_mandatory_representation（子类型强制的表示）
subtype_mandatory_representation_context（子类型强制的表示相关环境）

subtype_mandatory_shape_representation（子类型强制的形状表示）
Supplied_part_version（外购件版本）
supplied_part_relationship（供应零件关系）
supplied_part_relationship（外购零件关系）
Supplied_part_version（供应零件版本）
Supplier（供应商）
supplier_id（供应商标识）
supplier_part_number（外购零件号）
surface（曲面）
surface_curve（曲面曲线）
surface_finish_specification（表面粗糙度规范）
surface_of_linear_extrusion（线性延拓曲面）
surface_of_revolution（旋转曲面）
surface_patch（曲面片）
surface_replica（曲面复制）
swept_surface（扫成曲面）
the_bag（该袋）
the_role（该职能）
thru_effectivity_id（有效性结束标识）
top_secret（绝密）
topological_sense（拓扑方向）
topological_representation_item（拓扑表示项）
toroidal_surface（圆环曲面）
transformation（变换）
trimmed_curve（剪裁曲线）
TRUE（真）
uncertainty_measure_with_unit（误差度量）
unclassified（无级别）
uniform_curve（均匀曲线）
uniform_surface（均匀曲面）
unique_version_change_order（单个版本更改指令）
unique_version_change_order_rule（单个版本更改指令规则）
unit of functionality（功能单元）
unit_of_measure（度量单位）
units（单位）
UoF：功能单元（Units of Functionality）
Usage_constraint（使用约束）
using_representations（利用表示）
valid_geometrically_bounded_wf_curve（合法的几何有界线框曲线）
valid_geometrically_bounded_wf_point（合法的几何有界线框点）
valid_wireframe_edge_curve（合法的线框边曲线）
valid_wireframe_vertex_point（合法的线框顶点点）

value（值）
vector（矢量）
version（版本）
versioned_action_request（该版活动请求）
versioned_action_request_requires_solution（该版活动请求需要方案）
versioned_action_request_requires_status（该版活动请求需要状态）
vertex（顶）
vertex_loop（顶环）
vertex_point（顶点）
vertex_shell（顶壳）
volume_measure_with_unit（体积度量）
volume_unit（体积单位）
week_of_year_and_day_date（年的星期和日期）
wire_shell（线壳）
wireframe model（线框模型）
Wireframe_with_topology（拓扑线框）
withdrawn（撤销）
work_item（工作项）
Work_order（工作单）
work_order_id（工作单标识）
Work_request（工作请求）
work_request_id（工作请求标识）

中 华 人 民 共 和 国
国 家 标 准
工业自动化系统与集成
产品数据的表达与交换
第 203 部分：应用协议：配置控制设计
GB/T 16656.203—1997
*
中 国 标 准 出 版 社 出 版
北京复兴门外三里河北街 16 号
邮政编码：100045
电　　话：68522112
中国标准出版社秦皇岛印刷厂印刷
新华书店北京发行所发行　各地新华书店经售
版权专有　不得翻印
*
开本 880×1230 1/16　印张 25¾　字数 822 千字
1998 年 6 月第一版　1998 年 6 月第一次印刷
印数 1—2 000
*
书号：155066·1-14890　定价 114 元
*
标 目 340—24

GB/T 16656.203-1997