ГОСТ Р ИСО 10303-203—2003

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы автоматизации производства и их интеграция

ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ

Часть 203

Прикладной протокол Проекты с управляемой конфигурацией

Издание официальное

БЗ 5—2003/74

ГОССТАНДАРТ РОССИИ
Москва

Предисловие

1 РАЗРАБОТАН Научно-исследовательским центром (НИЦ) CALS-технологий «Прикладная логистика» и Всероссийским научно-исследовательским институтом стандартизации (ВНИИстандарт) Госстандарта России

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 431 «CALS-технологии»

3 ПРИНЯТ И ВВЕДЕН В ДЕЙСТВИЕ Постановлением Госстандарта России от 15 октября 2003 г. № 295-ст

4 Настоящий стандарт представляет собой полный аутентичный текст международного стандарта ИСО 10303-203—94 «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 203. Прикладной протокол. Проекты с управляемой конфигурацией» с учетом Поправок № 1 (1996 г.), № 2 (1998 г.) и Изменения № 1 (2000 г.)

5 ВВЕДЕН ВПЕРВЫЕ

Содержание

Введение

Стандарты серии ГОСТ Р ИСО 10303 распространяются на машинно-ориентированное представление данных об изделии и обмен этими данными. Целью является создание механизма, позволяющего описывать данные об изделии на протяжении всего его жизненного цикла независимо от конкретной системы. Характер такого описания делает его пригодным не только для обмена инвариантными файлами, но также и для создания баз данных об изделиях, коллективного пользования этими базами и архивирования соответствующих данных.

Стандарты серии ГОСТ Р ИСО 10303 представляют собой набор отдельно издаваемых стандартов (частей). Части данной серии стандартов относятся к одной из следующих тематических групп: методы описания, интегрированные ресурсы, прикладные протоколы, комплекты абстрактных тестов, методы реализации и аттестационное тестирование. Группы стандартов данной серии описаны в ГОСТ Р ИСО 10303-1. Настоящий стандарт входит в группу прикладных протоколов.

Стандарт устанавливает прикладной протокол (ПП) использования данных об изделии в определенном контексте, удовлетворяющем промышленным потребностям обмена конфигурационно-управляемыми данными об изделии в рамках трехмерных конструкций механических деталей и сборочных единиц (узлов). Для определения компоновки изделий в организациях используют различные автоматизированные системы. Общее описание изделия определяется его формой, конфигурациями изделия и возможностями применения ряда определений изделия для конкретной конфигурации. Эти данные могут быть размещены в базах данных одной или нескольких прикладных систем внутри организации. Объединение данных о форме изделия с данными о его конфигурации обеспечивает возможность для данной организации описать выпускаемые ею изделия без привлечения избыточной информации, хранящейся в несвязанных прикладных системах. Для взаимообмена проектной информацией об изделиях организация должна обеспечить представление соответствующих данных об изделии субподрядчикам, поставщикам и заказчикам.

Настоящий прикладной протокол определяет обмен описаниями изделий, представляемыми в трехмерной форме, и данными, определяющими конфигурацию этих изделий, и данными, управляющими этой конфигурацией. Этот протокол связан только с фазой проектирования в жизненном цикле изделия. Используя спецификацию данного протокола можно обмениваться только сведениями о конструкциях механических деталей и сборочных единиц (узлов). Определение трехмерной формы механической детали или сборочной единицы в настоящем протоколе может быть задано посредством любого из пяти различных типов геометрических представлений.

Однако форма изделия не является главной в данном протоколе. Основное внимание в спецификации протокола уделено данным, определяющим сопровождение и управление изделием. Этими данными являются:

- обозначение изделия для заказчиков и связь данного обозначения с компонентами изделия;
- документация по официальным изменениям и вариантам конструкции изделия;
- сведения (предыстория) о разработке изделия от его идеи до исполнения (выпуска);
- структура взаимосвязи каждого компонента с изделием в целом;
- дополнительная информация о материалах, процессах, отделке и других требованиях к изделию;
- обозначение официальных поставщиков изделия или его проекта.

Настоящий протокол определяет контекст, область применения и информационные требования к обмену данными о конфигурационно-управляемых трехмерных конструкциях механических деталей и сборочных единиц, а также интегрированные ресурсы, необходимые для удовлетворения этим требованиям.

Прикладные протоколы обеспечивают основу для разработки реализаций стандартов серии ГОСТ Р ИСО 10303 и комплектов абстрактных тестов для аттестационного тестирования реализаций ПП.

Примечания

1 Настоящий стандарт дополнен приложениями, содержащими ряд поясняющих и справочных материалов к тексту основной части стандарта.

2 В тексте настоящего стандарта объекты и конструкции на языке EXPRESS в ряде случаев выделены полужирным шрифтом (например, **release_status**).

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы автоматизации производства и их интеграция

ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ

Часть 203

Прикладной протокол. Проекты с управляемой конфигурацией

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

Дата введения 2004-07-01

1 Область применения

Настоящий стандарт определяет интегрированные ресурсы, необходимые для описания области обмена данными между прикладными системами и информационных требований для трехмерных конструкций (проектов) механических деталей и сборочных единиц. Конфигурация в этом контексте охватывает только данные и процессы, которые управляют трехмерными данными о конструкции изделия. Понятие обмена используется с целью распространения области применения стандарта только на данные, используемые как часть трехмерного определения изделия. Организации, обменивающиеся данными в соответствии с настоящим стандартом, могут быть связаны договорными отношениями, которые в стандарте не рассмотрены.

Примечание — Прикладная функциональная модель (ПФМ), приведенная в приложении Е, определяет графическое представление процессов и информационных потоков в соответствии с областью применения настоящего стандарта.

Область применения настоящего стандарта охватывает:

a) изделия, состоящие из механических деталей и сборочных единиц;

b) данные, определяющие изделие и управляющие его конфигурацией, относящиеся к стадии проектирования изделия;

c) изменение проекта (конструкции) и данные, связанные с документированием процесса внесения изменений;

d) пять типов представлений формы детали, которые включают каркасное и поверхностное представления без топологии, каркасную геометрию с топологией, разнородные поверхности с топологией, фасетное граничное представление и граничное представление;

e) альтернативные представления данных по различным правилам (дисциплинам) на стадии проектирования в жизненном цикле изделия;

f) обозначение государственных, отраслевых, фирменных или других спецификаций для проекта (конструкции), процесса, обработки поверхности и материалов, которые определены проектировщиком для конструируемого изделия;

g) государственное, отраслевое, фирменное или прочее обозначение стандартных частей с целью включения их в конструкцию (проект) изделия;

h) данные, необходимые для контроля за ходом проекта;

i) данные, необходимые для контроля за утверждением проекта, отдельных аспектов проекта или управления конфигурацией изделия;

j) данные, указывающие поставщика изделия или его проекта и, при необходимости, определенную информацию о поставщике;

k) обозначение контракта и ссылка на него, если деталь разрабатывается согласно контракту;

Издание официальное

l) обозначение уровня классификации защиты (конфиденциальности) отдельной детали или детали, являющейся компонентом сборочной единицы;

m) данные, применяемые при анализе проекта, или результаты его проверки, используемые для обоснования изменений, вносимых в проект.

Область применения настоящего стандарта не охватывает:

a) данные, применяемые при анализе проекта, или результаты его проверки, не используемые для обоснования изменений, вносимых в проект;

b) данные об изменениях в проекте по результатам исходного анализа до окончания данного проекта;

c) данные, определяющие изделие и управление его конфигурацией, относящиеся к любым стадиям жизненного цикла создания изделия помимо его проектирования;

d) коммерческие данные для управления проектированием конструкции;

e) альтернативные представления данных по различным правилам (дисциплинам), кроме стадии проектирования (например, на стадии производства);

f) использование трехмерной булевой геометрии для представления предметов проектирования;

g) данные, относящиеся к визуальному представлению любой формы изделия или управлению его конфигурацией.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

ГОСТ Р ИСО/МЭК 8824-1—2001 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации

ГОСТ Р ИСО 10303-1—99 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1. Общие представления и основополагающие принципы

ГОСТ Р ИСО 10303-11—2000 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 11. Методы описания. Справочное руководство по языку EXPRESS

ГОСТ Р ИСО 10303-21—2002 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 21. Методы реализации. Кодирование открытым текстом структуры обмена

ГОСТ Р ИСО 10303-31—2002 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 31. Методология и основы аттестационного тестирования. Общие положения

ГОСТ Р ИСО 10303-41—99 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 41. Интегрированные обобщенные ресурсы. Основы описания и поддержки изделий

ГОСТ Р ИСО 10303-43—2002 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 43. Интегрированные обобщенные ресурсы. Структуры представлений

ГОСТ Р ИСО 10303-44—2002 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 44. Интегрированные обобщенные ресурсы. Конфигурация структуры изделия

ИСО 31—92* Физические величины и единицы их измерения

ИСО 1000—92* Единицы измерения физических величин в системе Си и рекомендации по применению единиц, кратных им, и некоторых других единиц

ИСО 10303-42—94* Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 42. Интегрированные обобщенные ресурсы. Геометрическое и топологическое представления

ИСО 10303-501—2000* Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 501. Прикладная интерпретированная конструкция. Плоский контур

---

* Оригинал международного стандарта ИСО — во ВНИИКИ Госстандарта России.

ИСО 10303-502—2000* Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 502. Прикладная интерпретированная конструкция. Объемный контур

ИСО 10303-507—2001* Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 507. Прикладная интерпретированная конструкция. Геометрически ограниченная поверхность

ИСО 10303-509—2001* Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 509. Прикладная интерпретированная конструкция. Копируемая поверхность

ИСО 10303-510—2000* Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 510. Прикладная интерпретированная конструкция. Геометрически ограниченный контур

ИСО 10303-511—2001* Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 511. Прикладная интерпретированная конструкция. Топологически ограниченный контур

ИСО 10303-512—99* Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 512. Прикладная интерпретированная конструкция. Представление многогранного контура

ИСО 10303-514—99* Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 514. Прикладная интерпретированная конструкция. Представление сложной границы

3 Определения и сокращения

3.1 Термины, определенные в ГОСТ Р ИСО 10303-1

В настоящем стандарте использованы следующие термины:

- **комплект стандартных тестов;**
- **приложение;**
- **прикладная функциональная модель;**
- **прикладной контекст;**
- **прикладная интерпретированная модель;**
- **прикладной предмет (объект);**
- **прикладной протокол;**
- **прикладная эталонная модель;**
- **сборочная единица (узел);**
- **комплектующее (компонент);**
- **класс соответствия;**
- **требование соответствия;**
- **данные;**
- **обмен данными;**
- **метод реализации;**
- **информация;**
- **интегрированный ресурс;**
- **интерпретация;**
- **форма ЗСРП;**
- **изделие;**
- **данные об изделии;**
- **заявка о соответствии реализации протоколу (ЗСРП);**
- **структура;**
- **функциональная единица.**

3.2 Термины, определенные в ГОСТ Р ИСО 10303-31

В настоящем стандарте использованы следующие термины:

- **аттестационное тестирование;**
- **препроцессор;**
- **постпроцессор.**

---

* Оригинал международного стандарта ИСО — во ВНИИКИ Госстандарта России.

3.3 Термины, определенные в ИСО 10303-42

В настоящем стандарте использованы следующие термины:

- **дугообразное соединение;**
- **ось симметрии;**
- **ограничения;**
- **граница;**
- **трехмерная модель с граничным представлением;**
- **замкнутая кривая;**
- **замкнутая поверхность;**
- **соединение;**
- **соединенный компонент;**
- **кривая;**
- **цикл;**
- **размерность;**
- **область значений;**
- **пространство (оболочка);**
- **конечный;**
- **геометрическая система координат;**
- **граф;**
- **манипулятор;**
- **гомоморфный;**
- **список;**
- **ограниченное *d*-мерное пространство;**
- **незамкнутая кривая;**
- **незамкнутая поверхность;**
- **ориентированный;**
- **перекрытие;**
- **диапазон параметров;**
- **пространство параметров;**
- **координатная система размещения;**
- **самопересекающийся;**
- **самозацикленный;**
- **множество;**
- **размерность пространства;**
- **поверхность;**
- **топологический смысл.**

3.4 Термины, определенные в ГОСТ Р ИСО 10303-43

В настоящем стандарте использованы следующие термины:

- **координатное пространство;**
- **геометрически ограниченный;**
- **геометрически связанный.**

3.5 Термины, определенные в ГОСТ Р ИСО 10303-44

В настоящем стандарте использованы следующие термины:

- **узел-предок;**
- **структура спецификации;**
- **подчиненный узел;**
- **ориентированный ациклический граф;**
- **узел-потомок;**
- **форма, монтаж и функциональное назначение (изделия);**
- **краевой узел;**
- **связь;**
- **партия;**
- **узел (вершина);**
- **узел-родитель;**
- **структура списка частей (деталей);**
- **целевое назначение;**
- **корневой узел (вершина).**

3.6 Другие определения

В настоящем стандарте использованы следующие термины с соответствующими определениями:

3.6.1 **стадия проектирования** (design phase): Период, в течение которого изменяют техническое представление изделия.

3.6.2 **механическая деталь** (mechanical part): Физический объект заданной формы, изготовленный из соответствующего материала.

3.6.3 **трехмерная модель** (solid model): Трехмерный объект, внутреннее и внешнее описания которого разделены двумерной границей.

3.6.4 **подузел** (sub-assembly): Составная часть сборочной единицы, рассматриваемая как единое целое.

3.6.5 **каркасная модель** (wireframe model): Модель, описанная точками, отрезками и кривыми, контуры которой образуют определенную форму.

3.7 Сокращения

В настоящем стандарте использованы следующие сокращения:

ПФМ (AAM) — прикладная функциональная модель;
ПИМ (AIM) — прикладная интерпретированная модель;
ПП (AP) — прикладной протокол;
ПЭМ (ARM) — прикладная эталонная модель;
СП (BOM) — спецификация (перечень изделий и материалов);
САПР (CAD) — система автоматизированного проектирования;
УК (CM) — управление конфигурацией;
ОАГ (DAG) — ориентированный ациклический граф;
ИАП (ICAM) — интегрированное автоматизированное производство;
ОБ (ID) — обозначение (идентификация);
IDEF0 — язык описания ИАП уровня 0;
IDEF1X — язык описания ИАП уровня 1 — расширенный;
ЗСРП (PICS) — заявка о соответствии реализации протоколу;
ФЕ (UoF) — функциональная единица.

4 Информационные требования

В настоящем разделе определены информационные требования для обмена информацией о конструкциях (проектах) трехмерных механических деталей и сборочных единиц с управляемой конфигурацией.

Информационные требования определены как набор функциональных единиц, прикладных объектов и утверждений. Эти утверждения относятся к отдельным прикладным объектам и отношениям между ними. Информационные требования определены с использованием терминологии из области применения настоящего прикладного протокола.

Примечания

1 Графическое представление информационных требований приведено в приложении G.

2 В приложении F показано, как информационные требования соотносятся с процессами, охваченными областью применения настоящего прикладного протокола.

3 В таблице преобразований (см. 5.1) показано, как информационные требования могут быть выполнены с использованием интегрированных ресурсов настоящего стандарта. Использование интегрированных ресурсов определяет дополнительные требования, которые являются общими для прикладных протоколов в целом.

4.1 Функциональные единицы

В настоящем подразделе определены функциональные единицы (ФЕ) для прикладного протокола управления конфигурацией конструкций (проектов) трехмерных механических деталей и сборочных единиц. Настоящий стандарт определяет следующие функциональные единицы:

- advanced_boundary_representation;
- authorization;
- bill_of_material;
- design_activity_control;
- design_information;
- effectivity;

- end_item_identification;
- facetted_boundary_representation;
- manifold_surface_with_topology;
- non_topological_surface_and_wireframe;
- part_identification;
- shape;
- source_control;
- wireframe_with_topology.

Ниже рассмотрены конкретные функциональные единицы и описаны выполняемые ими функции. Прикладные объекты, входящие в ФЕ, определены в подразделе 4.2.

4.1.1 Функциональная единица advanced_boundary_representation

Функциональная единица advanced_boundary_representation содержит представление детали на основе трехмерных моделей с граничным представлением. Это представление позволяет определить кривые, поверхности и топологию, ограничивающую эти элементы. Границы определяются только соответствующей топологией. Вся геометрия, определяющая форму изделия, должна быть связана с топологией.

В данной ФЕ использован прикладной объект Advanced_B_rep.

4.1.2 Функциональная единица authorization

Функциональная единица authorization содержит конструктивы, описывающие часть данных об изделии, принятых на определенном уровне группой лиц, обладающих соответствующими полномочиями.

В данной ФЕ использованы следующие прикладные объекты:

- Approval;
- Person_organization.

4.1.3 Функциональная единица bill_of_material

Функциональная единица bill_of_material содержит предметы, необходимые для представления структурированного список материалов или компонентов, требуемых для изготовления детали.

В данной ФЕ использованы следующие прикладные объекты:

- Alternate_part;
- Component_assembly_position;
- Engineering_assembly;
- Engineering_make_from;
- Engineering_next_higher_assembly;
- Engineering_promissory_usage;
- Substitute_part.

4.1.4 Функциональная единица design_activity_control

Функциональная единица design_activity_control содержит информацию, описывающую предысторию вариантов детали. В ней определены исходные требования к детали, а также требования по внесению изменений в пересмотренные варианты детали. Данная ФЕ является основанием для проведения разработки на основе исходной или измененных вариантов детали.

В данной ФЕ использованы следующие прикладные объекты:

- Change_order;
- Change_request;
- Start_order;
- Start_request;
- Work_order;
- Work_request.

4.1.5 Функциональная единица design_information

Функциональная единица design_information содержит набор информации о детали, отдельно по обозначению и форме данной детали, привязанный к окончательно спроектированной детали. Эта информация содержит спецификации, связанные с конструкцией (проектом) и изготовлением детали, и ограничения по применению этих спецификаций.

В данной ФЕ использованы следующие прикладные объекты:

- Additional_design_information;
- Design_specification;
- Material_specification;
- Process_specification;

- Specification;
- Surface_finish_specification;
- Usage_constraint.

4.1.6 Функциональная единица effectivity

Функциональная единица effectivity содержит информацию относительно запланированного использования компонентов в модели изделия.

В данной ФЕ использованы следующие прикладные объекты:

- Planned_date_effectivity;
- Planned_effectivity;
- Planned_lot_effectivity;
- Planned_sequence_effectivity.

4.1.7 Функциональная единица end_item_identification

Функциональная единица end_item_identification содержит информацию относительно различных изделий, поставляемых организацией заказчикам. Данная ФЕ определяет информационные структуры организации для управления конфигурацией этих изделий.

В данной ФЕ использованы следующие прикладные объекты:

- Product_configuration;
- Product_model.

4.1.8 Функциональная единица faceted_boundary_representation

Функциональная единица faceted_boundary_representation определяет представление формы детали, когда плоские поверхности ограничивают трехмерную модель. В данном представлении используют только точки и плоские многоугольники, а вся топологическая информация присутствует в данном представлении неявно.

В данной ФЕ использован прикладной объект Faceted_B_rep.

4.1.9 Функциональная единица manifold_surface_with_topology

Функциональная единица manifold_surface_with_topology содержит представление формы детали с использованием множества топологически определенных поверхностей. Внешняя граница детали определяется трехмерными кривыми, поверхностями и их топологией.

В данной ФЕ использован прикладной объект Manifold_surface_with_topology.

4.1.10 Функциональная единица non_topological_surface_and_wireframe

Функциональная единица non_topological_surface_and_wireframe содержит представление формы детали с использованием поверхностной или каркасной геометрии без топологии. При формировании этого представления используются только точки, кривые и поверхности. Границы кривых явно задают точками на кривых и прямыми связями между этими точками и кривыми, которые они ограничивают. Границы поверхностей задают кривыми на этих поверхностях и прямыми связями между кривыми и ограничиваемыми ими поверхностями. Если поверхности и кривые не замкнуты, то они должны быть четко ограничены (обрезаны).

В данной ФЕ использован прикладной объект Non_topological_surface_and_wireframe.

4.1.11 Функциональная единица part_identification

Функциональная единица part_identification содержит структуру, посредством которой могут быть определены детали, их варианты (версии) и представления этих деталей с точки зрения различных дисциплин.

В данной ФЕ использованы следующие прикладные объекты:

- Design_discipline_product_definition;
- Part;
- Part_version.

4.1.12 Функциональная единица shape

Функциональная единица shape содержит геометрическое и топологическое определения детали. Данная ФЕ позволяет определять различные типы геометрического представления каждой части формы детали. Эти части, рассматриваемые вместе, образуют форму детали.

В данной ФЕ использованы следующие прикладные объекты:

- Geometric_model_representation;
- Shape;
- Shape_aspect.

4.1.13 Функциональная единица source_control

Функциональная единица source_control содержит информацию об организации, аттестованной на производство конкретной детали.

В данной ФЕ использованы следующие прикладные объекты:

- Supplier;
- Supplied_part_version.

4.1.14 Функциональная единица wireframe_with_topology

Функциональная единица wireframe_with_topology содержит каркасное представление формы детали, определенное топологией границ. Оно охватывает трехмерные кривые и соответствующую топологию.

В данной ФЕ использован прикладной объект Wireframe_with_topology.

4.2 Объекты предметной области

В настоящем подразделе определены прикладные объекты для прикладного протокола управления конфигурацией конструкций (проектов) трехмерных механических деталей и сборочных единиц. Каждый прикладной объект является отдельным элементом, реализующим уникальную прикладную концепцию и содержащим атрибуты, определяющие элементы данных этого объекта. Конкретные прикладные объекты и их определения приведены в последующих пунктах настоящего подраздела.

4.2.1 Прикладной объект Additional_design_information

Прикладной объект Additional_design_information определяет набор спецификаций, связанных с конструкцией (проектом) детали.

Пример 1 — Концепции проницаемости (прозрачности), поглощающей и отражающей способностей являются дополнительной проектной информацией в форме требований к изделию для последующих процессов (например, производства и испытания), чтобы знать, какие ограничительные требования относятся к детали.

4.2.2 Прикладной объект Advanced_B_rep

Прикладной объект Advanced_B_rep является типом объекта Geometric_model_representation (см. 4.2.15), представляющим форму или аспект формы детали на основе трехмерной модели с граничным представлением. Это представление позволяет определить кривые, поверхности и топологию их ограничения. Границы явно определены только топологией. Вся геометрия, определяющая форму детали, должна быть связана с топологией.

4.2.3 Прикладной объект Alternate_part

Прикладным объектом Alternate_part является деталь, взаимозаменяемая с другой деталью по форме, типоразмерам и функциональному назначению.

Примечание — Использование взаимозаменяемой детали не входит в область интересов организации, изготавливающей собственную деталь, и поэтому не отслеживается этой организацией.

Пример 2 — Проектом требуются винты определенного размера для листового металла. Такие винты выпускают различные предприятия, и они эквивалентны с точки зрения формы, типоразмера и функционального назначения для заданного размера. Винты имеют различные обозначения, присвоенные им изготовителем. Проектирующая организация не отслеживает, винты какого изготовителя используются в реальной сборочной единице. Эти винты — взаимозаменяемые детали.

4.2.4 Прикладной объект Approval

Прикладной объект Approval указывает на утверждение или не утверждение части данных об изделиях в рамках данной организации. С этим объектом связаны следующие данные:

- date;
- purpose;
- status.

4.2.4.1 *Данные date (дата)*

Эти данные определяют конкретную или потенциальную дату утверждения.

4.2.4.2 *Данные purpose (цель)*

Эти данные определяют основания для рассмотрения предмета утверждения.

4.2.4.3 *Данные status (статус)*

Эти данные определяют состояние утверждения части данных об изделии или отношений между частями этих данных. Допустимыми значениями для статуса являются: “утверждено (approved)” и “не утверждено (not approved)”. “Утверждено” означает, что удовлетворены необходимые условия, “не утверждено” — эти условия не удовлетворены.

4.2.5 Прикладной объект Change_order

Прикладной объект Change_order является типом объекта Work_order (см. 4.2.40), санкциони-

рующий разработку измененной конструкции (проекта) детали, которая заканчивается созданием нового варианта детали. С этим объектом связаны следующие данные:

- adopted_solution;
- change_date.

4.2.5.1 *Данные adopted_solution*

Эти данные определяют принятое решение, отобранное из набора рекомендуемых решений для Change_request (см. 4.2.6).

4.2.5.2 *Данные change_date*

Эти данные определяют дату начала реализации условий, заданных в Change_order.

4.2.6 Прикладной объект Change_request

Прикладной объект Change_request является типом объекта Work_request (см. 4.2.40), определяющим состав работ, которые должны быть выполнены при внесении изменений в конструкцию (проект) детали. С этим объектом связаны следующие данные:

- consequence;
- recommended_solution;
- version.

4.2.6.1 *Данные consequence*

Эти данные определяют влияние выполненной работы на качественные показатели, функциональные возможности или вид конкретного рекомендуемого воздействия на вариант детали. В одном объекте Change_request может быть задано несколько данных вида consequence.

Пример 3 — Изменением в проекте может быть утолщение подкоса крыла самолета. Последствием этого является увеличение нагрузки на крыло на величину ***x*** и, следовательно, увеличение грузоподъемности самолета на величину ***y***.

4.2.6.2 *Данные recommended_solution*

Эти данные определяют возможное решение, удовлетворяющее требованиям, описанным в заявке на внесение изменения в объекте Work_request (см. 4.2.41). В одном объекте Change_request может быть задано несколько данных вида recommended_solution.

4.2.6.3 *Данные version*

Эти данные определяют индивидуальное обозначение (идентификатор) для каждой итерации заявки на внесение изменения.

4.2.7 Прикладной объект Component_assembly_position

Прикладной объект Component_assembly_position определяет положение конкретного компонента в сборочной единице. С этим объектом связаны данные вида transformation.

4.2.7.1 *Данные transformation*

Эти данные определяют размещение и ориентацию компонента в геометрической системе координат сборочной единицы.

4.2.8 Прикладной объект Design_discipline_product_definition

Прикладным объектом Design_discipline_product_definition является одно из организационных определений или видов объекта Part_version (см. 4.2.20). С этим объектом связаны следующие данные:

- CAD_filename;
- creation_date;
- description;
- discipline_id.

Примечание — Данный объект может быть использован, чтобы отразить определение конкретного Part_version на любой промежуточной стадии проектирования, когда определение Part_version еще формально не отслеживается организацией. Объект может использоваться для отражения различных стадий в цикле определения изделия.

Пример 4 — Стадии в цикле определения изделия могут быть представлены в виде спецификации, модели конечных элементов, функциональной системы или с производственной точки зрения.

4.2.8.1 *Данные CAD_filename*

Эти данные определяют имя файла, содержащего геометрическое описание детали в системе автоматизированного проектирования (САПР). Данные не являются обязательными для конкретного Design_discipline_product_definition. Если имя файла задано, то предполагается, что файл

содержит внешнее определение формы детали, действующее в рамках организации, которая эту деталь разработала.

4.2.8.2 *Данные creation_date*

Эти данные определяют дату и время первоначального создания конкретного Design_discipline_product_definition.

4.2.8.3 *Данные description*

Эти данные определяют назначение конкретного определения изделия.

4.2.8.4 *Данные discipline_id*

Эти данные определяют вид или стадию, для которой дано определение изделия.

4.2.9 Прикладной объект Design_specification

Прикладной объект Design_specification является типом объекта Specification (см. 4.2.31), устанавливающим проектные требования к деталям. Эти требования не определяются другим проектом, конструктивными особенностями или ссылочными данными.

Пример 5 — В наборе требований, сформулированных в Design_specification, могут быть отражены ограничения по массе, габаритам, отражающей способности, цвету, прозрачности и внешнему виду.

4.2.10 Прикладной объект Engineering_assembly

Прикладной объект Engineering_assembly определяет отношения соподчиненности между сборочной единицей и деталью или подузлом. Подтипами Engineering_assembly могут быть прикладные объекты Engineering_next_higher_assembly (см. 4.2.12) или Engineering_promissory_usage (см. 4.2.13). С рассматриваемым объектом связаны данные вида security_code.

Примечание — Область применения настоящего стандарта ограничена обменом проектной спецификацией изделия. Эта область не охватывает все спецификации, необходимые данной организации.

4.2.10.1 *Данные security_code*

Эти данные определяют классификацию ограничения доступа к сведениям о компоненте сборочной единицы.

Примечание — Эти данные отличаются от атрибута security_code, связанного с Part_version (см. 4.2.20), в части привязки компонента к сборочной единице, изменяющей классификацию доступа к сведениям о детали, являющейся компонентом сборочной единицы с заданным уровнем доступа к сведениям об этой единице. Конкретная деталь может иметь свои ограничения по доступу к сведениям о ней независимо от ее использования в какой-либо сборочной единице. Код ограничения доступа, описанный в настоящем пункте, определяет ограничения по доступу к сведениям о детали в строго заданном контексте.

Пример 6 — Колесо может быть использовано в производстве ряда автомобилей и классифицироваться с точки зрения ограничения доступа как несекретное. То же самое колесо может быть использовано в сборочной единице, входящей в экспериментальный новый автомобиль, находящийся в стадии разработки. Хотя колесо непосредственно несекретно, сборочная единица, входящая в новый автомобиль, классифицируется как совершенно секретная и данное колесо, входящее в состав этой единицы, также должно классифицироваться как совершенно секретное. В этом случае security_code для колеса, входящего в сборочную единицу нового автомобиля, должен быть задан как совершенно секретный.

4.2.11 Прикладной объект Engineering_make_from

Прикладной объект Engineering_make_from определяет отношение между двумя деталями, при котором одна деталь используется как основа для проектирования другой. Это отношение устанавливают в проекте, чтобы оно могло быть прослежено в течение жизненного цикла изделия.

Пример 7 — Компания *A* проектирует сборочную единицу, используя держатель манжеты, спроектированный компанией *B*, но добавляет два дополнительных установочных отверстия к изделию компании *B*. В этом случае компания *A* обозначает держатель манжеты, разработанный компанией *B*, как деталь, определенную в Engineering_make_from.

4.2.12 Прикладной объект Engineering_next_higher_assembly

Прикладной объект Engineering_next_higher_assembly является типом объекта Engineering_assembly (см. 4.2.10). Он определяет отношение детали к непосредственному родителю в пределах иерархии сборочных единиц. С этим объектом связаны следующие данные:

- as_required;
- component_quantity;
- reference_designator;
- unit_of_measure.

4.2.12.1 *Данные as_required*

Эти данные определяют явно количество конкретного компонента или в зависимости от специфического применения.

Пример 8 — Если должно быть задано количество листового металла, используемого для изготовления детали, можно указать один рулон, где один — количество, а рулон — единица измерения. Рулоны металла могут быть также определены с указанием потребности в них. В этом случае указанная потребность — количество, а рулон — единица измерения.

4.2.12.2 *Данные component_quantity*

Эти данные определяют количество комплектующей детали, полученное на основе анализа структуры сборочной единицы, если оно не задано в as_required.

4.2.12.3 *Данные reference_designator*

Эти данные определяют индивидуальное обозначение (идентификатор), выделяющее конкретный экземпляр комплектующей детали в сборочной единице, когда в ее состав входят несколько одинаковых деталей.

Пример 9 — Если при сборке автомобиля используются четыре одинаковых колеса, тогда reference_designator выделяет, например, левое переднее колесо из других.

4.2.12.4 *Данные unit_of_measure*

Эти данные определяют единицу измерения, в терминах которой выражено количество в component_quantity.

4.2.13 Прикладной объект Engineering_promissory_usage

Прикладной объект Engineering_promissory_usage является типом объекта Engineering_assembly (см. 4.2.10). Он определяет отношение применимости детали в сборочной единице более высокого уровня, когда ее применимость в сборочной единице следующего за ней уровня не определена.

Пример 10 — Для крепления стрингеров в фюзеляже самолета необходимо большое количество кронштейнов. Проектировщик разрабатывает один из кронштейнов и должен определить его использование в структуре изделия. В этот момент проектировщик еще не знает, какая сборочная единица более высокого уровня, начиная со стрингера, к которому крепится кронштейн, определена в структуре изделия. Проектировщик, однако, знает конструкцию фюзеляжа, в котором этот кронштейн применяется. Путем использования данного объекта проектировщик может определить, что кронштейн входит в подузел фюзеляжа.

4.2.14 Прикладной объект Faceted_B_rep

Прикладной объект Faceted_B_rep является объектом типа Geometric_model_representation (см. 4.2.15), служащим для представления формы детали или части ее формы в виде, когда ограничивающими поверхностями для трехмерной модели с граничным представлением являются плоские поверхности. Это представление учитывает определения форм, представленных плоскими поверхностями в качестве ограничивающих поверхностей. В данном представлении используются только точки и плоские многоугольники, а большая часть топологической информация присутствует неявно. Оболочки состоят из граней, ограниченных исключительно многоугольниками.

4.2.15 Прикладной объект Geometric_model_representation

Прикладной объект Geometric_model_representation определяет форму или часть формы детали. Каждый из объектов Geometric_model_representation может включать один из следующих объектов: Advanced_B_Rep (см. 4.2.2), Faceted_B_rep (см. 4.2.14), Non_topological_surface_and_wireframe (см. 4.2.18), Manifold_-surface_with_topology (см. 4.2.16) или Wireframe_with_topology (см. 4.2.39).

4.2.16 Прикладной объект Manifold_surface_with_topology

Прикладной объект Manifold_surface_with_topology является объектом типа Geometric_model_representation (см. 4.2.15), представляющим форму или часть формы детали с использованием множества топологических поверхностей. Внешние границы детали определяют трехмерными кривыми, поверхностями и топологией.

4.2.17 Прикладной объект Material_specification

Прикладной объект Material_specification является объектом типа Specification (см. 4.2.31), определяющим свойства сырья, смесей или полуфабрикатов, используемых при изготовлении изделия.

4.2.18 Прикладной объект Non_topological_surface_and_wireframe

Прикладной объект Non_topological_surface_and_wireframe является типом объекта Geometric_model_representation (см. 4.2.15), представляющим форму или части формы детали с использованием геометрии поверхности или каркаса без топологии. Это представление формируют только на основе точек, кривых и поверхностей. Границы кривых явно задают точками на них и явными

связями между точками и ограничиваемыми ими кривыми. Границы поверхностей задают кривыми на них и явными связями между кривыми и ограничиваемыми ими поверхностями. Если поверхности и кривые являются не замкнутыми, они должны быть явно ограничены (обрезаны).

4.2.19 Прикладной объект Part

Прикладным объектом Part является деталь, изготовляемая или используемая в прозводственном процессе. С этим объектом связаны следующие данные:

- part_classification;
- part_nomenclature;
- part_number;
- part_type;
- standard_part_indicator.

4.2.19.1 *Данные part_classification*

Эти данные определяют семейство деталей, изготавливаемых по общим производственным процессам. Для конкретной детали (Part) данные вида part_classification могут быть не заданы.

Пример 11 — Механически обрабатываемые, штампованные или обтачиваемые детали могут быть категорированы по отдельным классам.

4.2.19.2 *Данные part_nomenclature*

Эти данные определяют наименование детали в пределах организации.

Примечание — Это наименование также может использоваться вне рамок данной организации. Например, в пределах государства или отрасли может быть принято соглашение по обозначению различных видов деталей общего назначения.

4.2.19.3 *Данные part_number*

Эти данные определяют индивидуальное обозначение детали в пределах организации.

4.2.19.4 *Данные part_type*

Эти данные определяют один из типов деталей. В настоящем стандарте определены следующие типы: деталь (detail), сборочная единица (assembly), покупной материал (customer supplied material) или неразборная сборочная единица (inseparable assembly). Изделием типа detail является деталь, указанная на самом низком уровне спецификации продукции. Изделием типа assembly является сборочная единица, состоящая из набора других компонентов, которые в собранном виде предназначены для выполнения определенной функции. Изделием типа government (государственное) для покупного материала является компонент, поставляемый из государственного источника. Изделием типа inseparable assembly является сборочная единица, которая после сборки не может быть разобрана без причинения физических повреждений по крайней мере одному из ее компонентов.

4.2.19.5 *Данные standard_part_indicator*

Эти данные определяют, описана ли конструкция детали (компонента, изделия) внешним образом по отношению спецификации состава сборочной единицы.

Примечание — Деталь (изделие) может быть стандартной в пределах компании, отрасли или другой организации.

4.2.20 Прикладной объект Part_version

Прикладной объект Part_version предназначен для обозначения представления детали (компонента) после того, как ее проект был официально выпущен или изменен. В новый экземпляр данного объекта вносят только те изменения, которые официально прослежены организацией, отвечающей за данную деталь (компонент). Прочие изменения в новый экземпляр рассматриваемого объекта не включают, но отслеживают при помощи объекта Design_discipline_product_definition (4.2.8). С этим объектом связаны следующие данные:

- contract_number;
- make_or_buy_code;
- release_status;
- revision_letter;
- security_code.

4.2.20.1 *Данные contract_number*

Эти данные предназначены для обозначения контракта, согласно которому была разработана деталь (компонент). Эти данные не задают для частного объекта Part_version. Если проект был

разработан согласно контракту, рассматриваемый атрибут должен определять индивидуальное обозначение этого контракта.

4.2.20.2 *Данные make_or_buy_code*

Эти данные определяют планируемый организацией-разработчиком способ получения детали. Этот атрибут может иметь всего два значения: изготавливаемое (make) или покупное (buy). Если значением является make, организация планирует изготовление детали собственными силами, если buy, то организация планирует закупку детали у поставщика.

4.2.20.3 *Данные release_status*

Эти данные определяют статус версии детали в части выдачи проектной информации. Данный атрибут может принимать только два значения: выпущенный (released) или не выпущенный (unreleased). Выпущенными считают те версии, которые были рассмотрены и одобрены для дальнейшего использования. Не выпущенными считают те версии, которые либо были не рассмотрены, либо не одобрены для дальнейшего использования.

4.2.20.4 *Данные revision_letter*

Эти данные определяют индивидуальное обозначение конкретной версии детали (компонента).

4.2.20.5 *Данные security_code*

Эти данные определяют классификацию ограничения доступа к конкретной версии детали (компонента).

4.2.21 Прикладной объект Person_organization

Прикладной объект Person_organization предназначен для обозначения лица и конкретной организации, в пределах которой лицо выполняет определенную функцию (играет роль). С этим объектом связаны следующие данные:

- address;
- organization;
- person;
- person_organization_id.

4.2.21.1 *Данные address*

Эти данные предназначены для указания адресата на бумажных документах и/или в электронной почте, а в некоторых случаях определяют физическое местоположение лица и организации (предприятия). Для конкретного объекта Person_organization адрес не является обязательным атрибутом.

4.2.21.2 *Данные organization*

Эти данные определяют коллектив людей, объединенных для достижения одной или нескольких общих целей.

4.2.21.3 *Данные person*

Эти данные определяют конкретного человека (отдельное лицо).

4.2.21.4 *Данные person_organization_id*

Эти данные определяют индивидуальное обозначение объекта Person_organization.

4.2.22 Прикладной объект Planned_date_effectivity

Прикладной объект Planned_date_effectivity является типом объекта Planned_effectivity (см. 4.2.23). Он предназначен для указания предполагаемого использования проектирующей организацией данной детали (компонента) в объекте Product_configuration. Этот объект определяет предполагаемое использование компонента в пределах конфигурации изделия одной или двумя связанными датами. С этим объектом связаны следующие данные:

- end_date;
- start date.

4.2.22.1 *Данные end_date*

Эти данные определяют окончательную дату использования проектирующей организацией конкретного компонента в составе изделия, заданного объектом Product_configuration. Для конкретного Planned_date_effectivity задание end_date не обязательно. Отсутствие при обмене информацией значения для end_date указывает на неопределенность этого значения.

4.2.22.2 *Данные start_date*

Эти данные определяют исходную дату начала использования данного компонента в составе изделия, заданного объектом Product_configuration.

4.2.23 Прикладной объект Planned_effectivity

Прикладной объект Planned_effectivity определяет предполагаемое использование компонента в конкретной конфигурации изделия. Каждый объект Planned_effectivity может быть представлен

объектом типа Planned_date_effectivity (4.2.22), Planned_lot_effectivity (4.2.24) или Planned_sequence_effectivity (4.2.25).

4.2.24 Прикладной объект Planned_lot_effectivity

Прикладной объект Planned_lot_effectivity является типом объекта Planned_effectivity (см. 4.2.23). Этот объект определяет использование детали (компонента) в составе изделия, задаваемого объектом Product_configuration, когда данная деталь изготавливается партиями. Такой подход применяют, когда детали изготовляют периодическими партиями и/или если основные характеристики детали могут варьироваться в зависимости от условий изготовления. С этим объектом связаны следующие данные:

- lot_number;
- lot_size;
- lot_size_unit_of_measure.

4.2.24.1 *Данные lot_number*

Эти данные определяют группу изделий, составляющих партию.

4.2.24.2 *Данные lot_size*

Эти данные определяют количество деталей в партии.

4.2.24.3 *Данные lot_size_unit_of_measure*

Эти данные определяют единицу измерения, в которой выражено количество изделий в партии (lot_size).

4.2.25 Прикладной объект Planned_sequence_effectivity

Прикладной объект Planned_sequence_effectivity является типом объекта Planned_effectivity (см. 4.2.23). Он предназначен для определения использования проектной организацией детали (компонента) в составе изделия, идентифицируемого объектом Product_configuration, когда данный компонент изготавливается партиями. С этим объектом связаны следующие данные:

- component_quantity;
- from_effectivity_id;
- quantity_unit_of_measure;
- thru_effectivity_id.

4.2.25.1 *Данные component_quantity*

Эти данные определяют количество деталей (компонентов), применяемых в конкретной конфигурации изделия.

4.2.25.2 *Данные from_effectivity_id*

Эти данные определяют изначально планируемые серийные номера диапазона деталей, используемых в объекте Planned_sequence_effectivity.

4.2.25.3 *Данные quantity_unit_of_measure*

Эти данные определяют единицу измерения для описания component_quantity.

Пример 12 — Значением quantity_unit_of_measure могут быть рулоны, листы, прутки проката и другие внесистемные единицы.

4.2.25.4 *Данные thru_effectivity_id*

Эти данные определяют окончательные серийные номера диапазона деталей, используемых в Planned_sequence_effectivity. В конкретном объекте Planned_sequence_effectivity значение thru_effectivity_id определять не обязательно.

4.2.26 Прикладной объект Process_specification

Прикладной объект Process_specification является типом объекта Specification (см. 4.2.31), используемым для обозначения процесса обработки изделия или материала.

Пример 13 — Примерами подобных типовых процессов являются: термообработка, сварка, плакирование, упаковка и маркировка.

4.2.27 Прикладной объект Product_configuration

Прикладной объект Product_configuration является разновидностью объекта Product_model (см. 4.2.28, 4.2.28.1). На основе этого объекта организуют управление конфигурацией. С этим объектом связаны следующие данные:

- item_id;
- phase_of_product.

Пример 14 — Для конкретного истребителя F14 объект Product_configuration задает конфигурацию D. Организация — разработчик истребителя F14 определила четыре конфигурации: A, B, C и D. В данном примере

задана конфигурация D. Отдельная выпускаемая модель истребителя может иметь ряд различных конфигураций, позволяющих заказчику выбрать одну из них.

4.2.27.1 *Данные item_id*

Эти данные определяют обозначение варианта исходного объекта Product_model.

4.2.27.2 *Данные phase_of_product*

Эти данные определяют стадию жизненного цикла изделия, на которой производится обмен проектной информацией о конкретной конфигурации.

Пример 15 — Область деятельности организации может охватывать четыре стадии жизненного цикла изделия: разработка технологии и концепции, разработка и доводка конструкции, выпуск опытных образцов и испытания сборочной единицы, производство.

4.2.28 Прикладной объект Product_model

Прикладным объектом Product_model является изделие, которое организация поставляет клиентам. Этот объект определяют с целью планирования на стадии проектирования изделия. С этим объектом связаны данные model_name.

4.2.28.1 *Данные model_name*

Эти данные определяют индивидуальное обозначение, присвоенное организацией-разработчиком изделию, поставляемому заказчикам (клиентам).

Пример 16 — Идентификатор "F14" является обозначением в объекте Product_model, определяющим военный самолет (истребитель).

4.2.29 Прикладной объект Shape

Прикладной объект Shape является математическим представлением формы детали (компонента).

4.2.30 Прикладной объект Shape_aspect

Прикладным объектом Shape_aspect является отдельная часть формы детали (компонента). Форма компонента состоит из одного или нескольких объектов Shape_aspect.

Примечание — При проектировании часто используют множество различных типов геометрических моделей, определяющих форму детали в зависимости от сложности ее формы на разных физических уровнях. Каждый из этих физических уровней может быть определен в виде объекта Shape_aspect.

4.2.31 Прикладной объект Specification

Прикладным объектом Specification является документ, содержащий определения, процессы или правила, связанные с уникальными качествами, которыми должны обладать применяемый процесс или законченная деталь (компонент). Объектом Specification может быть один из следующих объектов: Design_specification (см. 4.2.9), Material_specification (см. 4.2.17), Process_specification (см. 4.2.26), или Surface_finish_specification (см. 4.2.37). С этим объектом связаны следующие данные:

- specification_code;
- specification_source.

Пример 17 — Требования к материалу, качеству обработки поверхности и процессу являются примерами описаний, которые могут быть заданы неявно посредством ссылки на другие документы, взамен их явного определения при каждом использовании.

4.2.31.1 *Данные specification_code*

Эти данные определяют индивидуальное обозначение объекта Specification. Данное обозначение является уникальным в пределах данной организации.

4.2.31.2 *Данные specification_source*

Эти данные определяют организацию, отвечающую за объект Specification.

4.2.32 Прикладной объект Start_order

Прикладной объект Start_order является типом объекта Work_order (см. 4.2.40), предназначенным для фиксации факта начала работ по проектированию детали, заканчивающихся созданием исходной версии детали.

4.2.33 Прикладной объект Start_request

Прикладной объект Start_request является типом объекта Work_request (см. 4.2.41), определяющим работу, которую требуется выполнить при разработке исходной версии изделия.

4.2.34 Прикладной объект Substitute_part

Прикладным объектом Substitute_part является деталь (компонент) сборочной единицы, форма, размеры и функциональное назначение которой могут отличаться от аналогичных свойств заменяе-

мой детали, но данная деталь должна полностью удовлетворять требованиям, предъявляемым к заменяемой ею детали в контексте сборочной единицы.

Примечания

1 Форма и количество заменяющих деталей не обязательно должны соответствовать заменяемой детали.

2 Данный прикладной объект также может быть использован для устранения каскадного порождения версий, то есть переобозначения всех сборочных единиц более высоких уровней при создании новой версии компонента нижележащего уровня.

3 Данный прикладной объект позволяет определить только одностороннюю замену. Если в заданном контексте компонент А заменяет компонент В, то В не обязательно заменяет А, если это явно не определено в другом экземпляре данного объекта. Если А заменяет В в контексте С, то в конечном счете в области контекста С в состав сборочной единицы может входить либо А, либо В, но не оба одновременно.

Пример 18 — Диаметр (120±12) см может быть заменен диаметром (125,00±6,25) см, обеспечивающим те же самые измеряемые размеры выпускаемой продукции.

4.2.35 Прикладной объект Supplied_part_version

Прикладным объектом Supplied_part_version является деталь (компонент), определяемая своим обозначением (номером) и поставщиком. С этим объектом связаны следующие данные:

- certification_required;
- supplier_part_numher.

4.2.35.1 *Данные certification_required*

Эти данные указывают на необходимость проверки (сертификации) поставщика до приобретения у него соответствующих деталей (компонентов).

4.2.35.2 *Данные supplier_part_number*

Эти данные определяют номер детали, используемый организацией-изготовителем. В конкретном объекте Supplied_part_version задавать значение supplier_part_number не обязательно.

Пример 19 — Компания А использует деталь 1234567-1 в составе сборочной единицы. Эта деталь фактически закуплена в компании В, но каждая деталь подвергнута дополнительному испытанию для подтверждения ее соответствия требованиям к сборочной единице. Компания В идентифицирует это изделие как AABBCCD-E. Компания А определяет компанию В как поставщика номер 52088. В компании А деталь идентифицирована номером 1234567-1. Поставщиком данной детали является компания В. Кодом поставщика данной детали является 52088, а номером поставляемой детали — AABBCCD-E.

4.2.36 Прикладной объект Supplier

Прикладной объект Supplier определяет организацию, изготавливающую или спроектировавшую деталь (компонент). С этим объектом связаны данные supplier_id.

Эти данные определяют индивидуальное обозначение организации — изготовителя детали. Данным обозначением может быть идентификатор, принятый организацией самостоятельно и присвоенный ей в соответствии с отраслевым или международным кодом.

4.2.37 Прикладной объект Surface_finish_specification

Прикладной объект Surface_finish_specification является типом объекта Specification (см. 4.2.31), описывающим необходимые свойства текстур поверхностей или защитных покрытий детали в процессе ее изготовления или выпуска как законченного изделия.

4.2.38 Прикладной объект Usage_constraint

Прикладной объект Usage_constraint служит для задания ограничения по применению объекта Specification (см. 4.2.31). Он связывает раздел требований с определенной информацией или текстом, который нужно использовать в конкретном случае. С этим объектом связаны следующие данные:

- element;
- value.

4.2.38.1 *Данные element*

Эти данные определяют конкретный раздел или тему в спецификации, подлежащие ограничению или пояснению.

4.2.38.2 *Данные value*

Эти данные определяют ограничения или специфические данные, которые накладываются на раздел или тему в элементе, ограничивающем их использование.

Пример 20 — Изделие должно быть окрашено в соответствии со стандартом предприятия. Тогда требования выглядят как Specification_code = ABCD-1; Specification_source = XYZ Company; а в задаваемых парах ограничения по использованию Usage_constraint могут быть: element = colour; value = green; element = coats; value = 3.

4.2.39 Прикладной объект Wireframe_with_topology

Прикладной объект Wireframe_with_topology является типом объекта Geometric_model_representation (см. 4.2.15), представляющим форму или части формы детали с использованием каркасной геометрии для неявного задания объема тела или топологии ограничивающих его ребер. Это представление формируют при помощи трехмерных кривых и топологии, определяющей граф векторов и ребер.

4.2.40 Прикладной объект Work_order

Прикладной объект Work_order определяет документ, санкционирующий начало работ по разработке или модификации детали. Каждый Work_order является объектом типа Start_order (см. 4.2.32) или Change_order (см. 4.2.5). Он является результатом реализации одного или нескольких объектов Work_request (см. 4.2.41). С этим объектом связаны следующие данные:

- additional_data;
- analysis_data;
- work_order_id.

4.2.40.1 *Данные additional_data*

Эти данные определяют дополнительную информацию об объекте проектирования, возникающую в результате проработки соответствующего множества предлагаемых Work_request. Наличие additional_data в конкретном Work_order необязательно.

4.2.40.2 *Данные analysis_data*

Эти данные определяют оценочные результаты проработок выполняемости и обоснованности условий, изложенных в соответствующих Work_request в части проектирования или модификации детали (компонента). Наличие analysis_data в конкретном Work_order необязательно.

4.2.40.3 *Данные Work_order_id*

Эти данные определяют индивидуальное обозначение (идентификатор) работы, санкционированной Work_order.

4.2.41 Прикладной объект Work_request

Прикладной объект Work_request определяет документ, санкционирующий начало или возобновление определенной работы при разработке детали (компонента). Каждый Work_request является объектом типа Start_request (см. 4.2.32) или Change_order (см. 4.2.5). С этим объектом связаны следующие данные:

- description;
- reason;
- request_date;
- status;
- work_request_id.

4.2.41.1 *Данные description*

Эти данные определяют содержание работы, выполняемой при разработке или уточнении проекта детали (компонента).

4.2.41.2 *Данные reason*

Эти данные определяют причину реализации Work_request.

4.2.41.3 *Данные request_date*

Эти данные определяют дату создания Work_request.

4.2.41.4 *Данные status*

Эти данные определяют текущий уровень завершения Work_request. Допустимыми значениями статусов для Work_request являются: proposed (предложенный), in-work (в работе), released (выпущенный) и hold (отложенный).

4.2.41.5 *Данные work_request_id*

Эти данные определяют индивидуальное обозначение предложенной работы по проектированию детали.

4.3 Прикладные утверждения

В настоящем подразделе определены прикладные утверждения, принятые для прикладного протокола управления конфигурацией конструкций (проектов) трехмерных механических деталей и сборочных единиц. Эти утверждения определяют все отношения между прикладными объектами (далее до конца настоящего подраздела — объект), количество элементов отношений и правила, необходимые для обеспечения целостности и достоверности прикладных объектов прикладной и функциональной единицы (ФЕ). Сами прикладные утверждения и их определения приведены ниже.

4.3.1 Объекты Additional_design_information и Specification

Каждый Additional_design_information представляет собой набор из одного или нескольких объектов Specification. Каждый объект Specification идентифицируют только одним Additional_design_information.

4.3.2 Объекты Approval и Person_organization

Каждый Approval санкционируется одним или несколькими объектами Person_organization. Каждый Person_organization может санкционировать произвольное количество объектов Approval (включая нулевое и единичное).

4.3.3 Объекты Design_discipline_product_definition и Additional_design_information

Каждый Design_discipline_product_definition включает произвольное количество объектов Additional_design_information (включая нулевое и единичное). Каждый Additional_design_information может быть применен к одному или нескольким объектам Design_discipline_product_definition.

4.3.4 Объекты Design_discipline_product_definition и Approval

Каждый объект Design_discipline_product_definition санкционируется только одним Approval. Каждый Approval санкционирует произвольное количество объектов Design_discipline_product_definition (включая нулевое и единичное).

4.3.5 Объекты Design_discipline_product_definition и Engineering_assembly

Каждый Design_discipline_product_definition используют как компонент в произвольном количестве объектов Engineering_assembly (включая нулевое и единичное). Каждый компонент Engineering_assembly определен только одним Design_discipline_product_definition.

Каждый Design_discipline_product_definition является сборочной единицей для произвольного количества объектов Engineering_assembly (включая нулевое и единичное). Каждая сборочная единица Engineering_assembly определена только одним Design_discipline_product_definition.

4.3.6 Объекты Design_discipline_product_definition и Engineering_make_from

Каждый Design_discipline_product_definition является базовой конструкцией для произвольного количества объектов Engineering_make_from (включая нулевое и единичное). Базовая конструкция каждого Engineering_make_from определяется только одним объектом Design_discipline_product_definition.

Каждый Design_discipline_product_definition является исходным в произвольном количестве объектов Engineering_make_from (включая нулевое и единичное). Каждый Engineering_make_from использует в качестве исходного только один Design_discipline_product_definition.

4.3.7 Объекты Engineering_assembly и Planned_effectivity

Каждый Engineering_assembly применим в произвольном количестве объектов Planned_effectivity (включая нулевое и единичное). Каждый Planned_effectivity определяет применяемость только одного объекта Engineering_assembly.

4.3.8 Объекты Engineering_assembly и Substitute_part

Каждый Engineering_assembly имеет заменяемые компоненты, определяемые произвольным количеством объектов Substitute_part (включая нулевое и единичное). Каждый Substitute_part может быть заменяющим только для одного компонента (детали) в объекте Engineering_assembly.

4.3.9 Объекты Engineering_next_higher_assembly и Component_assembly_position

Расположение каждого Engineering_next_higher_assembly либо не определяется, либо определяется одним объектом Component_assembly_position. Каждый Component_assembly_position определяет положение только одного объекта Engineering_next_higher_assembly.

4.3.10 Объекты Geometric_model_representation и Component_assembly_position

Каждый Geometric_model_representation определяет положения компонентов, задаваемых произвольным количеством объектов Component_assembly_position (включая нулевое и единичное). Каждый Component_assembly_position определяет компонент только в одном Geometric_model_representation.

Каждый Geometric_model_representation представляет сборочную единицу, положение которой задается произвольным количеством объектов Component_assembly_position (включая нулевое и единичное). Каждый Component_assembly_position представляет сборочную единицу только в одном Geometric_model_representation.

4.3.11 Объекты Part и Alternate_part

Каждый Part может быть взаимозаменяем произвольным количеством объектов Alternate_part. Каждый Alternate_part может быть взаимозаменяем с одним или несколькими объектами Part.

4.3.12 Объекты Part и Part_version

Каждый Part связан с одним или несколькими объектами Part_version. Каждый Part_version должен определять разновидность только одного объекта Part.

4.3.13 Объекты Part и Substitute_part

Каждый Part связан с произвольным количеством замещающих его в заданных условиях объектов Substitute_part (включая нулевое и единичное). Каждый Substitute_part является объектом типа Part и может в заданных условиях замещать один или несколько объектов Part.

4.3.14 Объекты Part_version и Approval

Каждый Part_version санкционируется только одним объектом Approval. Каждый Approval санкционирует один или несколько объектов Part_version.

4.3.15 Объекты Part_version и Design_discipline_product_definition

Каждый объект Part_version определен одним или несколькими объектами Design_discipline_product_definition. Каждый объект Design_discipline_product-_definition определяет только один объект Part_version.

4.3.16 Объекты Part_version и Supplied_part_version

Каждый Part_version идентифицирован посредством одного или нескольких объектов Supplied_part_version. Каждый Supplied_part_version соответствует только одному объекту Part_version.

4.3.17 Объекты Person_organization и Design_discipline_product_definition

Каждый Person_organization определяет разработчика для произвольного количества объектов Design_discipline_product_definition (включая нулевое и единичное). Каждый объект Design_discipline_product_definition создан только одним объектом Person_organization.

4.3.18 Объекты Person_organization и Part

Каждый Person_organization определяет владельца произвольного количества объектов Part (включая нулевое и единичное). Каждый объект Part принадлежит только одному объекту Person_organization.

4.3.19 Объекты Person_organization и Part_version

Каждый Person_organization определяет разработчика произвольного количества объектов Part_version (включая нулевое и единичное). Каждый объект Part_version создан только одним Person_organization.

4.3.20 Объекты Person_organization и Supplier

Каждый Person_organization обозначает произвольное количество объектов Supplier (включая нулевое и единичное). Каждый объект Supplier обозначен только одним объектом Person_organization.

4.3.21 Объекты Planned_effectivity и Approval

Каждый Planned_effectivity санкционирован только одним объектом Approval. Каждый Approval санкционирует один или несколько объектов Planned_effectivity.

4.3.22 Объекты Product_configuration и Approval

Каждый Product_configuration санкционирован только одним объектом Approval. Каждый Approval санкционирует один или несколько объектов Product_configuration.

4.3.23 Объекты Product_configuration и Part

Каждый Product_configuration соответствует произвольному количеству объектов Part (включая нулевое и единичное). Конфигурация каждого Part определяется произвольным количеством объектов Product_configuration (включая нулевое и единичное).

4.3.24 Объекты Product_configuration и Planned_effectivity

Каждый Product_configuration связан с произвольным количеством объектов Planned_effectivity (включая нулевое и единичное). Каждый Planned_effectivity идентифицирует применяемость только одного объекта Product_configuration.

4.3.25 Объекты Product_model и Product_configuration

С каждым объектом Product_model связаны один или несколько объектов Product_configuration. Каждый Product_configuration определяет конфигурацию только одного объекта Product_model.

4.3.26 Объекты Shape и Design_discipline_product_definition

Каждый Shape определяет геометрические характеристики одного или нескольких объектов Design_discipline_product_definition. Геометрические характеристики каждого Design_discipline_product_definition либо не определяются, либо определяются только одним объектом Shape.

4.3.27 Объекты Shape и Geometric_model_representation

Каждый Shape представляет произвольное количество объектов Geometric_model_representation (включая нулевое и единичное). Каждый Geometric_model_representation является представлением произвольного количества объектов Shape (включая нулевое и единичное).

4.3.28 Объекты Shape и Shape_aspect

Каждый Shape состоит из произвольного количества объектов Shape_aspect (включая нулевое и единичное). Каждый Shape_aspect входит в состав только одного объекта Shape.

4.3.29 Объекты Shape_aspect и Geometric_model_representation

Каждый Shape_aspect представлен одним или несколькими объектами Geometric_model_representation. Каждый Geometric_model_representation представляет только один объект Shape_aspect.

4.3.30 Объекты Shape_aspect и Specification

Характеристики каждого Shape_aspect описаны произвольным количеством объектов Specification (включая нулевое и единичное). Каждый Specification описывает характеристики произвольного количества объектов Shape_aspect (включая нулевое и единичное).

4.3.31 Объекты Specification и Usage_constraint

Каждый Specification ограничен произвольным количеством объектов Usage_constraint (включая нулевое и единичное). Каждый Usage_constraint относится только к одному объекту Specification.

4.3.32 Объекты Supplied_part_version и Approval

Каждый Supplied_part_version санкционирован только одним объектом Approval. Каждый Approval санкционирует произвольное количество объектов Supplied_part_version (включая нулевое и единичное).

4.3.33 Объекты Supplier и Supplied_part_version

Каждый Supplier производит произвольное количество объектов Supplied_part_version (включая нулевое и единичное). Каждый Supplied_part_version генерируется только одним объектом Supplier.

4.3.34 Объекты Work_order и Approval

Каждый Work_order санкционируется только одним объектом Approval. Каждый Approval санкционирует один или несколько объектов Work_order.

4.3.35 Объекты Work_order и Part_version

Каждый Work_order относится к одному или нескольким объектам Part_version. Каждый Part_version появляется в результате применения произвольного количества объектов Work_order (включая нулевое и единичное).

4.3.36 Объекты Work_order и Work_request

Каждый Work_order объединяет один или несколько объектов Work_request. Каждый Work_request либо не входит в состав ни одного, либо только одного объекта Work_order.

4.3.37 Объекты Work_request и Approval

Каждый Work_request санкционирован только одним объектом Approval. Каждый Approval санкционирует один или несколько объектов Work_request.

4.3.38 Объекты Work_request и Part_version

Каждый Work_request связан с одним или несколькими объектами Part_version. Каждый Part_version может ссылаться на произвольное количество объектов Work_request (включая нулевое и единичное).

Если Work_request задает начало работ по нескольким объектам типа Part_version, то эти объекты должны относиться к различным объектам Part.

4.3.39 Объекты Work_request и Person_organization

Каждый Work_request адресуется одному или нескольким объектам Person_organization. Каждый Person_organization извещает о произвольном количестве объектов Work_request (включая нулевое и единичное).

5 Прикладная интерпретированная модель

5.1 Таблицы отображения

Данный раздел содержит таблицы отображения 1—14, показывающие, как каждая ФЕ и прикладной объект настоящего стандарта (см. раздел 4) отображаются в одну или несколько структур ресурсов ПИМ (см. 5.2).

Таблица отображения состоит из пяти граф. Ниже поясняется их содержание.

Графа "Прикладной элемент" содержит наименование прикладного элемента, указанное в определении прикладного объекта из 4.2. Наименования прикладных объектов выделены прописными буквами. Наименования атрибутов (данных) указаны после соответствующего объекта и набраны строчными буквами.

Графа "Элемент ПИМ" содержит наименование элемента ПИМ, указанное в ПИМ (5.2), термин 'IDENTICAL MAPPING (ИДЕНТИЧНОЕ ОТОБРАЖЕНИЕ)' или термин 'PATH (ПУТЬ ДОСТУПА)'. Объекты ПИМ набраны строчными буквами. Наименования атрибутов объектов ПИМ записаны как <наименование объекта>.<наименование атрибута>. Отображение прикладного элемента может быть представлено несколькими связанными элементами ПИМ. Каждый из этих элементов ПИМ представлен в таблице отдельной строкой. Термин 'IDENTICAL MAPPING' указывает, что оба элемента из прикладного указания отображаются в тот же самый элемент ПИМ. Термин 'PATH' указывает, что конкретное прикладное утверждение отображается в полный ссылочный путь.

Графа "Источник". Для элементов ПИМ, интерпретируемых из интегрированных ресурсов, в этой графе указывают номер соответствующего стандарта (части) серии ГОСТ Р ИСО 10303. Для элементов ПИМ, введенных в настоящий стандарт, — номер настоящей части (203) в серии ГОСТ Р ИСО 10303.

Графа "Правила". Здесь могут быть приведены один или несколько номеров, указывающих правила, относящиеся к текущему элементу ПИМ. Для правил, полученных из отношений между прикладными объектами, то же правило определяет вхождения отображений для всех элементов ПИМ, используемых в правиле. Расширенные названия правил перечислены в списке, помещенном после таблиц отображения.

Графа "Ссылочный путь". Чтобы полностью описать отображение элемента ПИМ, может быть необходимым определение последовательности ссылок через несколько связанных элементов ПИМ. Каждый элемент ПИМ размещают в отдельной строке данной графы с символом, определяющим его отношение к элементу ПИМ, размещенному на следующей строке. Графа ссылочного пути доступа, таким образом, определяет роль данного элемента ПИМ относительно элемента ПИМ, приведенного в следующей строке. В случае существования ссылочного пути два или более таких связанных элемента ПИМ определяют интерпретацию интегрированных ресурсов, удовлетворяющих требованиям, указанным этим элементом ПЭМ.

Для каждого элемента ПИМ, созданного для использования в настоящем стандарте, ссылочный путь определен протяженностью от части интегрированного ресурса до супертипа. В случае двунаправленной ссылки от элемента ПИМ с двумя атрибутами, которые порождают ссылочный путь, каждый ссылочный путь заключен в круглые скобки. Элемент ПИМ, являющийся корнем обоих ссылочных путей, записывают либо в наборах круглых скобок, либо перед каждым набором круглых скобок.

Для записи ссылочных путей используют следующие нотационные соглашения:

a) [ ] : с целью выполнения информационных требований нужны разнородные элементы ПИМ или секции ссылочных путей;

b) ( ) : с целью выполнения информационных требований разнородные элементы ПИМ или секции ссылочных путей обозначены как альтернативные в пределах данного преобразования;

c) { } : с целью выполнения информационных требований заключенная в фигурные скобки секция ограничивает ссылочный путь;

d) -> : атрибут ссылается на объект или выбранный тип данных, указанный в следующей строке;

e) <- : ссылка на объект или выбранный тип данных приведена в следующей строке;

f) [i] : атрибут представлен агрегированным типом, единственный член которого указан в следующей строке;

g) [n] : атрибут представлен агрегированным типом, член *n* которого указан в следующей строке;

h) => : объект является супертипом объекта, указанного в следующей строке;

i) <= : объект является подтипом объекта, указанного в следующей строке;

j) = : для ограничения выбора или значения используют данные следующих типов: строкового (string), выбранного (select) или перечисления (enumeration).

Таблица 1 — Отображение ФЕ **advanced_boundary_representation**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
ADVANCED_B_REP	advanced_brep_shape_representation	203	73	advanced_brep_representation < = shape_representation

Таблица 2 — Отображение ФЕ **authorization**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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 для person_organization	PATH		2	cc_design_approval <= approval_assignment approval_assignment.assigned_approval-> approval <- approval_person_organization.authorized_approval 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

Продолжение таблицы 2

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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 < - person_address.people[i] personal_address < = address) (person_and_organization.the_ organization->organization < - organizational_address.organizations[i] organizational_address <= address)
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

Окончание таблицы 2

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
person_organization для 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 для 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 для part_version	PATH		46	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_formation product_definition_formation
person_organization для 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

Таблица 3 — Отображение ФЕ **bill_of_materials**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
ALTERNATE_PART	product	41		product
				{product<- alternate_product_relationship.alternate}
COMPONENT_ ASSEMBLY_POSITION	([shape_representation_ relationship] [representation_ relationship_ with_transformation])	43	16	
	(mapped_item)		73	
transformation	(transformation)	43		(shape_representation_relationship <= representation_relationship => representation_relationship_with_ transformation representation_relationship_with_ transformation.transformation_operator-> transformation)
	([mapped_item. mapping_target] [representation_map. mapping_ origin])			([mapped_item mapped_item.mapping_target] [mapped_item mapped_item.mapping_source-> representation_map representation_map.mapping_origin])
ENGINEERING_ ASSEMBLY	assembly_component_ usage	44	16, 70	
security_code	security_classification_ level	44	28, 37, 56	assembly_component_usage classified_item = 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 для planned_effectivity	PATH			assembly_component_usage <= product_definition_usage <= product_definition_relationship<- product_definition_effectivity.usage
engineering_assembly для 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	

Продолжение таблицы 3

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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'}
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 для component_assembly_ position	(context_dependent_ shape_representation) (mapped_item)	41 43	16 73	(next_assembly_usage_occurrence<= assembly_component_usage <= 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 product_definition_relationship.relating_ product_definition-> product_definition characterized_product_definition = product_definition characterized_definition = characterized_ product_definition characterized_definition <- property_definition.definition

Окончание таблицы 3

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
				property_definition{=>product_definition_shape} <- property_definition_representation.definition property_definition_representation{ => shape_definition_representation} <- property_definition_representation.used_representation->
engineering_next_higher_assembly для component_assembly_position (продолжение)	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 => product_definition_usage => assembly_component_usage <- assembly_component_usage_substitute.substitute}

Таблица 4 — Отображение ФЕ **design_activity_control**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
CHANGE_ORDER	change	203	10, 11, 74	change <= action_assignment {action_assignment.assigned_action-> action => executed_action => directed_action}

Продолжение таблицы 4

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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
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}

Продолжение таблицы 4

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
START_REQUEST	start_request	203	61, 62, 63	start_request <= action_request_assignment
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 для 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 для 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) (change change.items[i]-> work_item work_item = product_definition_ formation) product_definition_formation => product_definition_formation_with_ specified_source
work_order для 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		

Продолжение таблицы 4

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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
work_request_id	versioned_action_request.id	41		
work_request для 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 для 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

Окончание таблицы 4

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
work_request для 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

Таблица 5 — Отображение ФЕ **design_information**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
ADDITIONAL_ DESIGN_ INFORMATION	document	41	31	document {document <- document_relationship.relating_ document}
additional_design information для 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

Окончание таблицы 5

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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
specification для 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		

Таблица 6 — Отображение ФЕ **effectivity**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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 для approval	PATH		32	[product_definition_effectivity] [configuration_effectivity] <- approved_item = configuration_effectivity approved_item <- cc_design_approval.items[i] cc_design_approval

Окончание таблицы 6

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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
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		

Таблица 7 — Отображение ФЕ **end_item_identification**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
PRODUCT_ CONFIGURATION	configuration_item	44	12, 13	
item_id	configuration_item. identification	44		
phase_of_product	configuration_item. purpose	41		
product_configuration для approval	PATH		12	configuration_item approved_item = configuration_item approved_item <- cc_design_approval.items[i] cc_design_approval
product_configuration для part	configuration_design. design	44		configuration_item<- configuration_design.configuration configuration_design configuration_design.design-> product_definition_formation{ => product_definition_formation_with_ specified_source} product_definition_formation. of_product-> product
product_configuration для 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 для product_configuration	configuration_item .item_ concept	44		product_concept <- configuration_item.item_concept configuration_item

Таблица 8 — Отображение ФЕ **faceted_boundary_representation**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
FACETED_B_REP	faceted_brep_shape_ representation	203	73	faceted_brep_shape_representation <= shape_representation

Таблица 9 — Отображение ФЕ **manifold_surface_with_topology**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
MANIFOLD_ SURFACE_ WITH_TOPOLOGY	manifold_surface_shape_ representation	203	73	rnanifold_surface_shape_ representation <= shape_representation

Таблица 10 — Отображение ФЕ **non_topological_surface_and_wireframe**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
NON_TOPOLOGICAL_ SURFACE_AND_ WIREFRAME	(geometrically_bounded_ wireframe_shape_ representation)	203	73	(geometrically_bounded_wireframe_ shape_representation <= shape_representation)
	(geometrically_bounded_ surface_shape_ representation)		73	(geometrically_bounded_surface_shape_ representation <= shape_representation)

Таблица 11 — Отображение ФЕ **part_identification**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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]	41	30	product_definition product_definition.frame_of_reference-> product_definition_context [product_definition_context.life_ cycle_stage]
	[application_context_ element.name]	41		[product_definition_context <= application_context_element application_context_element.name]

Продолжение таблицы 11

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
design_discipline_ product_definition для 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 для approval	PATH		39	product_definition approved_item = product_definition approved_item <- cc_design_approval.items[i] cc_design_approval
design_discipline_ product_definition для engineering_assembly (является сборочной единицей)	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 для engineering_assembly (используется в качестве компонента)	product_definition_ relationship.related_ product_definition	41		product_definition <- product_definition_relationship.related_ product_definition product_definition_relationship => product_definition_usage => assembly_component_usage
design_discipline_ product_definition для engineering_make_ from (базовая конструкция)	PATH			product_definition <- product_definition_relationship.relating_ product_definition product_definition_relationship => design_make_from_relationship
design_discipline_ product_definition для engineering_make_ from (производная конструкция)	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

Продолжение таблицы 11

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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 для alternate_part	PATH			product <- alternate_product_relationship.base alternate_product_relationship alternate_product_relationship.alternate-> product
part для 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 для 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		

Окончание таблицы 11

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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 для 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 для design_discipline_product_definition	product_definition.formation	41		product_definition_formation_with_specified_source <= product_definition_formation <- product_definition.formation product_definition
part_version для 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

Таблица 12 — Отображение ФЕ **shape**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
GEOMETRIC_ MODEL_ REPRESENTATION	shape_representation	41	33, 34, 25, 27, 29, 71, 72, 73	
geometric_model_ representation для component_assembly_ position (представляет компоненты)	(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 для component_assembly_ position (представляет сборочную единицу)	(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})
SHAPE	shape_definition_ representation	41		
shape для 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 для geometric_ model_representation	PATH	41		shape_definition_representation <= property_definition_representation property_definition_representation.used_ representation-> representation => shape_representation
shape для 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		

Окончание таблицы 12

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
shape_aspect для 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
shape_aspect для 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

Таблица 13 — Отображение ФЕ **source_control**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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.formation product_definition <- product_definition_relationship.related_ product_definition product_definition_relationship => supplied_part_relationship} product_definition_formation.of_ product ->product product.id

Окончание таблицы 13

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
supplied_part_version для approval	PATH		45	product_definition_formation_with_specified_source <= 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 для 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_assignment.items[i]-> person_organization_item person_organization_item = product_definition_formation product_definition_formation => product_definition_formation_with_specified_source

Таблица 14 — Отображение ФЕ **wireframe_with_topology**

Прикладной элемент	Элемент ПИМ	Источник	Правила	Ссылочный путь
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)

В таблицах 1 — 14 используют ссылки на следующие правила:

1) approval_requires_approval_date_time;
2) approval_requires_approval_person_organization;
3) approvals_are_assigned;
4) as_required_quantity;
5) certification_requires_approval;
6) certification_requires_date_time;
7) change_request_requires_approval;
8) change_request_requires_date_time;
9) change_request_requires_person_organization;
10) change_requires_approval;
11) change_requires_date_time;
12) configuration_item_requires_approval;
13) configuration_item_requires_person_organization;
14) contract_requires_approval;
15) contract_requires_person_organization;
16) coordinated_assembly_and_shape;
17) dependent_instantiable_action_directive;
18) dependent_instantiable_approval_status;
19) dependent_instantiable_certification_type;
20) dependent_instantiable_contract_type;
21) dependent_instantiable_date;
22) dependent_instantiable_date_time_role;
23) dependent_instantiable_document_type;
24) dependent_instantiable_named_unit;
25) dependent_instantiable_parametric_representation_context;
26) dependent_instantiable_person_and_organization_role;
27) dependent_instantiable_representation_item;
28) dependent_instantiable_security_classification_level;
29) dependent_instantiable_shape_representation;
30) design_context_for_property;
31) document_to_product_definition;
32) effectivity_requires_approval;
33) geometric_representation_item_3d;
34) global_unit_assignment;
35) no_shape_for_make_from;
36) no_shape_for_supplied_part;
37) pdu_requires_security_classification;
38) product_concept_requires_configuration_item;
39) product_definition_requires_approval;
40) product_definition_requires_date_time;
41) product_definition_requires_person_organization;
42) product_requires_person_organization;
43) product_requires_product_category;
44) product_requires_version;
45) product_version_requires_approval;
46) product_version_requires_person_organization;
47) product_version_requires_security_classification;
48) restrict_action_request_status;
49) restrict_approval_status;
50) restrict_certification_type;
51) restrict_contract_type;
52) restrict_date_time_role;
53) restrict_document_type;
54) restrict_person_organization_role;
55) restrict_product_category_value;
56) restrict_security_classification_level;

57) security_classification_optional_date_time;
58) security_classification_requires_approval;
59) security_classification_requires_date_time;
60) security_classification_requires_person_organization;
61) start_request_requires_approval;
62) start_request_reqiures_date_time;
63) start_request_requires_person_organization;
64) start_work_requires_approval;
65) start_work_requires_date_time;
66) subtype_mandatory_action;
67) subtype_mandatory_effectivity;
68) subtype_mandatory_product_context;
69) subtype_mandatory_product_definition_formation;
70) subtype_mandatory_product_definition_usage;
71) subtype_mandatory_representation;
72) subtype_mandatory_representation_context;
73) subtype_mandatory_shape_representation;
74) unique_version_change_order_rule;
75) versioned_action_request_requires_solution;
76) versioned_action_request_requires_status.

5.2 Сокращенный EXPRESS-листинг прикладной интерпретированной модели

В настоящем подразделе определена EXPRESS-схема, использующая элементы интегрированных ресурсов и содержащая типы, специализированные объекты, правила и функции, определенные в настоящем стандарте. Данный подраздел также определяет модификации текстовых материалов для конструктивов, импортированных из других интегрированных ресурсов. Определения и EXPRESS-описания, представленные в конкретных интегрированных ресурсах для конструктивов, использованных в ПИМ, могут включать элементы списка выбора и подтипы, которые не импортированы в ПИМ. Требования, установленные в конкретных интегрированных ресурсах и относящиеся к таким спискам и подтипам, применяются только для тех элементов, которые импортированы в ПИМ.

EXPRESS-спецификация

```
*)
SCHEMA config_control_design;
  USE FROM application_context_schema — ГОСТ Р ИСО 10303-41
      (application context,
      application_protocol_definition,
      product_context,
      product_definition_context,
      product_concept_context);

  USE FROM product_definition_schema — ГОСТ Р ИСО 10303-41
      (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 — ГОСТ Р ИСО 10303-44
      (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 — ГОСТ Р ИСО 10303-44
    (configuration_item,
    configuration_design,
    configuration_effectivity);
  USE FROM product_concept_schema — ГОСТ Р ИСО 10303-44
    (product_concept);
  USE FROM product_property_definition_schema — ГОСТ Р ИСО 10303-41
    (product_definition_shape,
    property_definition,
    shape_aspect,
    shape_aspect_relationship);
  USE FROM product_property_representation_schema — ГОСТ Р ИСО 10303-41
    (context_dependent_shape_representation,
    property_definition_representation,
    shape_representation,
    shape_representation_relationship,
    shape_definition_representation);
  USE FROM representation_schema — ГОСТ Р ИСО 10303-43
    (functionally_defined_transformation,
    item_defined_transformation,
    global_uncertainty_assigned_context,
    mapped_item,
    representation,
    representation_context,
    parametric_representation_context,
    representation_item,
    representation_map,
    representation_relationship,
    representation_relationship_with_transformation,
    using_representations);
  USE FROM geometry_schema — ИСО 10303-42
    (axis1_placement,
    axis2_placement_2d,
    axis2_placement_3d,
    b_spline_curve,
    b_spline_curve_with_knots,
    b_spline_surface,
    b_spline_surface_with_knots,
    bezier_curve,
    bezier_surface,
    boundary_curve,
    cartesian_point,
    cartesian_transformation_operator_3d,
    circle,
    composite_curve,
    composite_curve_on_surface,
    composite_curve_segment,
    conic,
    conical_surface,
    curve,
    curve_bounded_surface,
    curve_replica,
    cylindrical_surface,
    degenerate_pcurve,
```

degenerate_toroidal_surface,
dimension_count,
dimension_of,
direction,
ellipse,
evaluated_degenerate_pcurve,
geometric_representation_context,
geometric_representation_item,
hyperbola,
intersection_curve,
line,
offset_curve_3d,
offset_surface,
outer_boundary_curve,
parabola,
pcurve,
plane,
point,
point_on_curve,
point_on_surface,
point_replica,
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,
swept_surface,
toroidal_surface,
trimmed_curve,
uniform_curve,
uniform_surface,
vector);

USE FROM topology_schema — ИСО 10303-42
(closed_shell,
connected_edge_set,
connected_face_set,
edge_curve,
edge_loop,
face_bound,
face_outer_bound,
face_surface,
open_shell,
oriented_closed_shell,
oriented_face,
path,
poly_loop,
topological_representation_item,

```
        vertex_loop,
        vertex_point,
        vertex_shell,
        wire_shell);
    USE FROM geometric_model_schema -- ИСО 10303-42
        (brep_with_voids,
        edge_based_wireframe_model,
        faceted_brep,
        geometric_curve_set,
        geometric_set,
        manifold_solid_brep,
        shell_based_surface_model,
        shell_based_wireframe_model);
    USE FROM action_schema -- ГОСТ Р ИСО 10303-41
        (action,
        action_method,
        action_request_solution,
        action_request_status,
        action_status,
        action_directive,
        directed_action,
        versioned_action_request);
    USE FROM certification_schema -- ГОСТ Р ИСО 10303-41
        (certification,
        certification_type);
    USE FROM approval_schema -- ГОСТ Р ИСО 10303-41
        (approval_date_time,
        approval_person_organization,
        approval,
        approval_status,
        approval_relationship);
    USE FROM contract_schema -- ГОСТ Р ИСО 10303-41
        (contract,
        contract_type);
    USE FROM security_classification_schema -- ГОСТ Р ИСО 10303-41
        (security_classification,
        security_classification_level);
    USE FROM person_organization_schema -- ГОСТ Р ИСО 10303-41
        (person_and_organization,
        organization_relationship,
        personal_address,
        organizational_address,
        organizational_project,
        person_and_organization_role);
    USE FROM date_time_schema -- ГОСТ Р ИСО 10303-41
        (date_and_time,
        date,
        calendar_date,
        ordinal_date,
        week_of_year_and_day_date,
        date_time_role);
    USE FROM document_schema -- ГОСТ Р ИСО 10303-41
        (document_with_class,
        document_usage_constraint,
```

```
        document_type,
        document_relationship);
USE FROM effectivity_schema — ГОСТ Р ИСО 10303-41
        (effectivity,
        serial_numbered_effectivity,
        dated_effectivity,
        lot_effectivity);
USE FROM management_resources_schema — ГОСТ Р ИСО 10303-41
        (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 measure_schema — ГОСТ Р ИСО 10303-41
        (measure_value,
        area_measure,
        count_measure,
        descriptive_measure,
        context_dependent_measure,
        parameter_value,
        plane_angle_measure,
        positive_length_measure,
        positive_plane_angle_measure,
        mass_measure,
        solid_angle_measure,
        volume_measure,
        named_unit,
        context_dependent_unit,
        conversion_based_unit,
        si_unit,
        area_unit,
        length_unit,
        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,
        solid_angle_measure_with_unit,
        volume_measure_with_unit,
        global_unit_assigned_context);
  USE FROM aic_edge_based_wireframe; — ИСО 10303-501
  USE FROM aic_shell_based_wireframe; — ИСО 10303-502
  USE FROM aic_geometrically_bouded_surface; — ИСО 10303-507
  USE FROM aic_manifold_surface; — ИСО 10303-509
  USE FROM aic_geometrically_bouded_wireframe; — ИСО 10303-510
```

```
USE FROM aic_topologically_bouded_surface; — ИСО 10303-511
USE FROM aic_faceted_brep; — ИСО 10303-512
USE FROM aic_advanced_brep; — ИСО 10303-514
(*
```

Примечание — Схемы, на которые выше даны ссылки, можно найти в следующих стандартах серий ГОСТ Р ИСО 10303 (ИСО 10303):

application_context_schema	— ГОСТ Р ИСО 10303-41;
product_definition_schema	— ГОСТ Р ИСО 10303-41;
product_structure_schema	— ГОСТ Р ИСО 10303-44;
configuration_management schema	— ГОСТ Р ИСО 10303-44;
product_concept_schema	— ГОСТ Р ИСО 10303-44;
product_property_definition_schema	— ГОСТ Р ИСО 10303-41;
product_property_representation_schema	— ГОСТ Р ИСО 10303-41;
representation_schema	— ГОСТ Р ИСО 10303-43;
geometry_schema	— ИСО 10303-42;
geometric_model_schema	— ИСО 10303-42;
action_schema	— ГОСТ Р ИСО 10303-41;
certification_schema	— ГОСТ Р ИСО 10303-41;
approval_schema	— ГОСТ Р ИСО 10303-41;
contract_schema	— ГОСТ Р ИСО 10303-41;
security_classification_schema	— ГОСТ Р ИСО 10303-41;
person_organization_schema	— ГОСТ Р ИСО 10303-41;
date_time_schema	— ГОСТ Р ИСО 10303-41;
document_schema	— ГОСТ Р ИСО 10303-41;
effectivity_schema	— ГОСТ Р ИСО 10303-41;
management_resources_schema	— ГОСТ Р ИСО 10303-41;
measure_schema	— ГОСТ Р ИСО 10303-41;
aic_edge_based_wireframe	— ИСО 10303-501;
aic_shell_based_wireframe	— ИСО 10303-502;
aic_geometrically_bouded_surface	— ИСО 10303-507;
aic_manifold_surface	— ИСО 10303-509;
aic_geometrically_bouded_wireframe	— ИСО 10303-510;
aic_topologically_bouded_surface	— ИСО 10303-511;
aic_faceted_brep	— ИСО 10303-512;
aic_advanced_brep	— ИСО 10303-514.

5.2.1 Фундаментальные понятия и допущения

Настоящий стандарт разработан применительно к управлению конфигурацией трехмерных данных о конструкции изделия. Фундаментальная концепция рассматриваемой схемы состоит в том, что проектная организация может управлять конфигурацией конструкций изделий конкретных типов. Данная схема не предназначена для управления конструкциями изделий посредством корректировки чертежей. Она разработана в качестве средства передачи данных при управлении конфигурацией трехмерных конструкций (проектов) изделий.

5.2.1.1 *Связь формы изделия с данным о его конфигурации*

Форма изделий в настоящем стандарте представлена посредством объекта **shape_representation**. Этот объект и его подтипы определяют геометрические и/или топологические объекты, задающие конкретный тип представления. Каждую необходимую механическую деталь или сборочную единицу следует определять посредством экземпляра объекта **product**. Каждое изделие может, в свою очередь, иметь, по крайней мере, одну версию, заданную экземпляром объекта **product_definition_formation**. Каждая версия может иметь одно или несколько определений, заданных объектом **product_definition**. Каждое определение может иметь собственное представление формы. Это реализуется посредством использования объектов ПИМ, связывающих экземпляр объекта **product_definition** с соответствующим экземпляром объекта **shape_representation**. Общее представление формы объекта **product_definition** задают посредством экземпляра объекта **product_definition_shape**. Затем заданную форму

изделия связывают с объектом **shape_representation** посредством экземпляра объекта **shape_definition_representation**. Объект **shape_definition_representation** наследует атрибуты от соответствующих супертипов объекта **property_definition_representation**, ссылающегося на объект **shape_representation**, содержащий геометрию и/или топологию формы детали и выбираемый тип данных, названный **characterized_definition**. Данный тип допустим для представления объектов **shape_aspect**, **shape_aspect_relationship** или **characterized_product_definition**, которые также представлены выбираемыми типами. Для указания формы объекта **product_definition** должен использоваться заданный подтип **product_definition_shape** для объекта **product_definition**. Использование объекта **product_definition_shape** ограничено выбираемым типом объекта **characterized_product_definition**. Выбираемый тип объекта **characterized_product_definition** допустим для представления объектов **product_definition** или **product_definition_relationship** посредством применения списка выбора. Ссылка на данный список гарантирует, что объект **product_definition_shape** будет использоваться для определения формы объекта **product_definition**. При рассмотрении аспекта формы объекта **product_definition** или отношения между двумя аспектами его формы должен использоваться объект **property_definition**, а его атрибут **definition** должен ссылаться на экземпляр объекта **shape_aspect** или **shape_aspect_relationship**. Для определения формы изделия при этом должна быть приведена ссылка на объект **product_definition**. Ограничение правила **subtype_mandatory_shape_representation** (см. 5.2.5.73) указывает, что для определения формы детали должен быть использован один из подтипов объекта **shape_representation**. Представление объектов **shape_aspect** или **shape_aspect_relationship** может быть задано любым набором объектов **representation_items**.

5.2.1.2 *Связь формы компонента с формой сборочной единицы*

Существуют два метода, которые могут быть использованы для связи формы компонента детали с формой сборки детали, в которой он применяется. Первый метод заключается в определении формы каждой детали (компонента и сборочной единицы), последующего связывания двух форм и представления информации, определяющей ориентацию компонента детали в сборке посредством соответствующего преобразования. Второй метод состоит из определения формы каждой детали (компонента и сборочной единицы) и последующего непосредственного включения формы компонента в форму сборочной единицы. Первый метод должен быть использован для связи форм, отображаемых различными типами представлений. Второй метод может быть использован для объединения представлений компонентов в представление сборочной единицы, если использованы два одинаковых типа представления.

Оба метода используют объекты **shape_representation** и **product_definition**. Первый метод также использует объекты отношения **product_definition_relationship**, **shape_representation_relationship** и **representation_relationship_with_transformation** (экземпляр каждого из этих объектов входит в экземпляр сложного объекта на основе двух объектов, связанных отношением AND) и объект **context_dependent_shape_representation**. Второй метод использует объекты **mapped_item** и **representation_map**.

Первый метод позволяет связать форму компонента с формой сборочной единицы для каждого объекта **shape_representation**, определяющего формы компонентов и сборочной единицы из объекта **product_definition,** связанных посредством ссылок в объекте **shape_representation_relationship**. В этом случае информация об ориентации может быть представлена посредством формирования сложного экземпляра, образованного из объектов **shape_representation_relationship** и **representation_relationship_with_transformation**, связанных оператором AND. Объект **representation_relationship_with_transformation** ссылается на объект **transformation**, который является выбираемым типом, позволяющим определить ориентацию на основе объекта **axis2_placement_3d** в каждом представлении объектов **item_defined_transformation** или **cartesian_transformation_operator** в операторе преобразования для **functionally_defined_transformation**. Кроме того, должен быть задан экземпляр объекта **context_dependent_shape_representation**, явно связывающий объект **shape_representation_relationship**, определяющий отношение двух форм с объектом **product_definition_relationship**, определяющим отношение соподчиненности компонента и сборочной единицы между двумя объектами **product_definitions**.

При использовании второго метода установления отношения формы компонента с формой сборочной единицы, на объект **shape_representation**, определяющий геометрию и/или топологию формы компонента детали, дается ссылка из экземпляра объекта **representation_map**, определяемая атрибутом **mapping_source** экземпляра объекта **mapped_item**. Атрибут **mapped_representation** объекта **representation_map** ссылается на подтип **shape_representation**, определяющий геометрическое и/или топологическое представление формы. Экземпляр объекта **mapped_item** затем добавляется к набору **items** в объекте **shape_representation**, определяющем геометрию и/или топологию сборки детали.

5.2.1.3 *Типы представления формы*

Настоящий стандарт устанавливает восемь типов представления форм деталей: каркасные представления с использованием граничных и оболочных моделей, геометрически ограниченные каркасные представления, разнородные модели поверхности, геометрически ограниченные поверхностные модели, фасетно ограниченные представления сплошных трехмерных моделей и ограниченные представления сплошных трехмерных моделей. Каждый из этих типов представления является самодостаточным, то есть один тип не может содержать другой тип. Каждый из типов задают посредством подтипов объекта **shape_representation**. Каждый подтип содержит локальные правила, управляющие типами используемых в нем геометрических и/или топологических объектов. Каждый **shape_representation** должен быть одним из подтипов, за исключением случая применения его для представления формы сборочной единицы в соответствии с методом 1, описанным выше. В этом случае экземпляр объекта **shape_representation** будет содержать только объекты **axis2_placement_3d** в наборе **items** для определения ориентации в нем представлений компонентов. Так как правила в каждом из подтипов будут противоречить правилам других подтипов, любой **shape_representation**, на который дается ссылка из объекта **representation_map** для описания формы сборочной единицы по методу 2, должен иметь тот же самый тип, что и **shape_representation**, содержащий **mapped_item**, ссылающийся на этот **representation_map** в наборе **items**.

5.2.1.4 *Использования глобальных правил*

Многие из отношений между различными объектами в интегрированных ресурсах, описанных стандартами серии ГОСТ Р ИСО 10303, определены посредством общего количества связей (нулевого или множественного) между двумя связанными объектами. Это количество подразумевает, что данное отношение является необязательным или может включать один или несколько экземпляров объектов, связанных с единственным экземпляром другого объекта. В настоящем стандарте для ограничения количества связей использованы глобальные правила. В некоторых случаях данное ограничение сформулировано как "один к одному", а в других — "не менее одного". Примерами этих правил являются: **contract_requires_person_organization**, **approval_requires_approval_date_time** и **certification_requires_approval** (для задания только одной связи), а **change_request_requires_person_organization** и **product_requires_version** — для задания одной или нескольких связей.

Глобальные правила также применяют для ограничения значений только тех атрибутов строкового (STRING) типа данных, которые используются в контексте управления конфигурацией трехмерных конструкций (проектов) механических деталей и сборочных единиц. Примерами этих правил являются **restrict_approval_status** и **restrict_person_organization_role**.

5.2.1.5 *Назначение единиц измерения*

Должны быть заданы общие единицы измерения для представления формы. Это выполняют посредством создания экземпляра объекта **global_unit_assigned_context**. Данный объект содержит атрибут, учитывающий допустимый набор единиц измерения, заданных для **representation_context**. Каждый из объектов **shape_representation** содержит контекст для представления этих единиц. Если для конкретного экземпляра **shape_representation** необходимы единицы измерения, то этот экземпляр должен содержать объект **global_unit_assigned_context** в соответствующем атрибуте **context_of_items**.

5.2.2 Константы проектов с управляемой конфигурацией

EXPRESS-спецификация

```
*)
CONSTANT
(*
```

5.2.2.1 *Константа dummy_gri*

Константа **dummy_gri** обозначает объект **geometric_representation_item**, в котором атрибут **name**, используемый в функциях конструирования, имеет нулевое значение.

EXPRESS-спецификация

```
*)
  dummy_gri : geometric_representation_item := representation_item ( ' ' ) ||
                geometric_representation_item ( );
(*
```

5.2.2.2 *Константа dummy_tri*

Константа **dummy_tri** обозначает объект **topological_representation_item**, в котором атрибут **name**, используемый в функциях конструирования, имеет нулевое значение.

EXPRESS-спецификация

```
*)
    dummy_tri : topological_representation_item := representation_item ( ' ' ) ||
                topological_representation_item ( );
END_CONSTANT;
(*
```

5.2.3 Типы проектов с управляемой конфигурацией

5.2.3.1 *Тип work_item*

Тип **work_item** обозначает конкретный объект **product_definition_formation**, являющийся результатом исходной деятельности по проектированию или модификации проекта (конструкции).

EXPRESS-спецификация

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

5.2.3.2 *Тип change_request_item*

Типом **change_request_item** является конкретный объект **product_definition_formation** данной детали, изменяемой посредством объекта **change_request**.

EXPRESS-спецификация

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

5.2.3.3 *Тип start_request_item*

Типом **start_request_item** является конкретный объект **product_definition_formation** данной детали, конструируемой на основе объекта **start_request**.

EXPRESS-спецификация

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

5.2.3.4 *Тип certified_item*

Тип **certified_item** подтверждает сертификацию (**certification**) детали, поставляемой сторонней организацией. Наличие объекта **certification** подтверждает аттестацию (полномочия) сторонней организации на производство данной детали.

EXPRESS-спецификация

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

5.2.3.5 *Тип approved_item*

Тип **approved_item** определяет наличие объектов **approval** для объектов **product_definition_formation**, **product_definition**, **planned_effectivity**, **configuration_item**, **security_classification**, **change_request**, **change**, **start_request**, **start_work**, **certification** или **contract**, подтверждающих статус конкретных аспектов проекта.

EXPRESS-спецификация

```
(*
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;
(*
```

5.2.3.6 *Тип contracted_item*

Тип **contracted_item** связывает конкретный **product_definition_formation** с объектом **contract**.

EXPRESS-спецификация

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

5.2.3.7 *Тип classified_item*

Тип **classified_item** связывает объект **security_classification** с конкретным объектом **product_definition_formation** или отношением между двумя объектами **product_definition**, при конкретном применении данных элементов.

EXPRESS-спецификация

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

5.2.3.8 *Тип person_organization_item*

Тип **person_organization_item** задает объект **person_and_organization** для объектов **change_request**, **start_request**, **approval**, **configuration_item**, **product**, **product_definition_formation**, **product_definition**, **contract** или **security_classification**. Роль конкретного объекта **person_and_organization** контролируется функцией **cc_design_person_and_organization_correlation**, заданной в 5.2.6.2.

EXPRESS-спецификация

```
*)
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;
(*
```

5.2.3.9 *Тип date_time_item*

Тип **date_time_item** задает объект **date_and_time** для объектов **product_definition**, **change_request**, **start_request**, **change**, **start_work**, **approval_person_organization**, **contract**, **security_classification** или **certification**. Роль конкретного объекта **date_and_time** контролируется функцией **cc_design_date_time_correlation**, заданной в 5.2.6.3.

EXPRESS-спецификация

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

5.2.3.10 *Тип specified_item*

Тип **specified_item** связывает объект **specification** либо с объектом **product_definition**, либо с **shape_aspect**.

EXPRESS-спецификация

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

5.2.4 Объекты проекта с управляемой конфигурацией

5.2.4.1 *Определения объектов проекта с управляемой конфигурацией*

5.2.4.1.1 Объект mechanical_context

Объектом **mechanical_context** является объект **product_context**, применяемый для механических деталей.

Примечание — Применение данного объекта определяют с точки зрения контекста изделия в целом. Данный объект не используют при классификации или категорировании типа изделия. Описание объекта **product_context** в части механического использования рассматриваемого объекта определяет способ, посредством которого данное изделие влияет на обмен данными.

Пример 21 — Сборочная единица печатной платы определена в механическом контексте, когда заданы ее соответствующие физические свойства, такие как форма, описывающие ее местоположение в соответствующей сборке. Другие свойства печатной платы (например, требования по стыковке или функциональные требования) в механическом контексте могут быть не определены.

EXPRESS-спецификация

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

Формальное утверждение

WR1 — конкретный атрибут **discipline_type** объекта **mechanical_context,** который должен содержать значение 'mechanical'.

5.2.4.1.2 Объект design_context

Объектом **design_context** является объект **product_definition_context**, определяющий стадию жизненного цикла проекта посредством группы (фрейма) ссылок для объектов **product_definition**.

Примечание — Дальнейшее уточнение стадии в рамках контекста проектирования может быть проведено посредством объекта **application_context_element**.

EXPRESS-спецификация

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

Формальное утверждение

WR1 — конкретный атрибут **life_cycle_stage** объекта **design_context,** который должен содержать значение 'design'.

5.2.4.1.3 Объект design_make_from_relationship

Объект **design_make_from_relationship** определяет, что проект одного изделия (**product)** должен базироваться на проекте другого изделия. Применение объекта **design_make_from_relationship** также подразумевает, что физическое изделие, полученное в результате производства конкретного **relating_product_definition**, будет использовано для создания **related_product_definition**.

EXPRESS-спецификация

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

Определения атрибутов

SELF\product_definition_relationship.relating_product_definition — конкретный объект **product definition**, являющийся исходным для вывода из отношения;

SELF\product_definition_relationship.related_product_definition — конкретный объект **product_definition**, созданный из исходного **product_definition**.

Соответствующее глобальное правило

Для объекта **design_make_from_relationship** применяют следующее глобальное правило из настоящего стандарта:

- no_shape_for_make_from (см. 5.2.5.76).

5.2.4.1.4 Объект supplied_part_relationship

Объект **supplied_part_relationship** связывает обозначения двух изделий и определяет, что одно является обозначением, используемым внутри проектной организации, а другое — обозначением, используемым поставщиком.

EXPRESS-спецификация

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

Определения атрибутов

SELF\product_definition_relationship.relating_product_definition — конкретный **product_definition**, являющийся описанием и обозначением изделия (**product**) в проектной организации;

SELF\product_definition_relationship.related_product_definition — конкретный **product_definition**, являющийся описанием и обозначением изделия в организации, поставляющей деталь или проект в проектную организацию.

Неформальное утверждение

IP1 — виды описаний изделия в проектной организации и у поставщика должны быть эквивалентны в отношении его формы, местоположения и функций.

Соответствующее глобальное правило

Для объекта **supplied_part_relationship** применяют следующее глобальное правило из настоящего стандарта:

- no_shape_for_supplied_part (см. 5.2.5.77).

5.2.4.1.5 Объект change_request

Объект **change_request** является формальным извещением об изменении конкретной части (порции) данных об изделии.

EXPRESS-спецификация

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

Определение атрибута

items — набор объектов **change_request_item**, определяющих версии конкретного изделия, подлежащие изменению.

Соответствующие глобальные правила

Для объекта **change_request** применяют следующие глобальные правила из настоящего стандарта:

- change_request_requires_approval (см. 5.2.5.6);
- change_request_requires_person_organization (см. 5.2.5.7);
- change_request_requires_date_time (см. 5.2.5.8).

5.2.4.1.6 Объект start_request

Объект **start_request** является формальным извещением о начале нового проекта изделия (**product**) или группы изделий (**product**).

EXPRESS-спецификация

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

Определение атрибута

items — набор объектов **start_request_item**, обозначающих версии конкретных изделий.

Соответствующие глобальные правила

Для объекта **start_request** применяют следующие глобальные правила из настоящего стандарта:

- start_request_requires_approval (см. 5.2.5.11);
- start_request_requires_person_organization (см. 5.2.5.12);
- start_request_requires_date_time (см. 5.2.5.13).

5.2.4.1.7 Объект change

Объект **change** обозначает конкретные объекты **change_request**, входящие в данный проект (конструкцию) и определяющие новую версию изделия (**product**).

EXPRESS-спецификация

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

Определение атрибута

items — набор объектов **work_item**, обозначающих версии конкретных изделий, созданные в результате применения данного объекта **change**.

Соответствующие глобальные правила

Для объекта **change** применяют следующие глобальные правила из настоящего стандарта:

- change_requires_approval (см. 5.2.5.9);
- change_requires_date_time (см. 5.2.5.10);
- unique_version_change_order_rule (см. 5.2.5.19).

5.2.4.1.8 Объект start_work

Объект **start work** обозначает конкретные реализованные объекты **start_request**, на основе которых создана новая версия изделия (**product**).

EXPRESS-спецификация

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

Определение атрибута

items — набор объектов **work_item**, определяющих версии изделий (**product**), созданных в результате начала конкретной работы (**work**).

Соответствующие глобальные правила

Для объекта **start_work** применяют следующие глобальные правила из настоящего стандарта:

- start_work_requires_approval (см. 5.2.5.14);
- start_work_requires_date_time (см. 5.2.5.15).

5.2.4.1.9 Объект cc_design_certification

Объект **cc_design_certification** связывает объект **certification** с изделием (**product**), поставляемым внешней организацией.

EXPRESS-спецификация

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

Определение атрибута

items — набор объектов **certified_item**, определяющих объекты **supplied_part_relationship**, для которых задан конкретный объект **certification**.

5.2.4.1.10 Объект cc_design_approval

Объект **cc_design_approval** связывает объект **approval** с конкретной частью данных об изделии.

EXPRESS-спецификация

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

Определение атрибута

items — набор объектов **approved_item**, обозначающих конкретные объекты **product_definition_formation, product_definition, planned_effectivity, configuration_item, security_classification, change_request, change, start_request, start_work, certification** или **contract**, для которых применяют заданный объект **approval**.

Соответствующие глобальные правила

Для объекта **cc_design_approval** применяют следующие глобальные правила из настоящего стандарта:

- change_request_requires_approval (см. 5.2.5.6);
- change_requires_approval (см. 5.2.5.9);
- start_request_requires_approval (см. 5.2.5.11);
- start_work_requires_approval (см. 5.2.5.14);
- product_version_requires_approval (см. 5.2.5.22);
- product_definition_requires_approval (см. 5.2.5.26);
- certification_requires_approval (см. 5.2.5.28);

- contract_requires_approval (см. 5.2.5.35);
- security_classification_requires_approval (см. 5.2.5.38);
- effectivity_requires_approval (см. 5.2.5.66);
- configuration_itern_requires_approval (см. 5.2.5.67).

5.2.4.1.11 Объект cc_design_contract

Объект **cc_design_contract** связывает объект **contract** с объектом **product_definition_formation**.

EXPRESS-спецификация

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

Определение атрибута

items — набор объектов **contracted_item**, определяющих версии конкретных изделий (**product**), связанных с данным объектом **contract.**

5.2.4.1.12 Объект cc_design_security_classification

Объект **cc_design_security_classification** связывает объект **security_classification** с конкретной частью данных об изделии.

EXPRESS-спецификация

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

Определение атрибута

items — набор объектов **classified_item**, обозначающий версии или определенные отношения конкретных изделий (**product**), для которых задан объект **security_classification**.

Соответствующие глобальные правила

Для объекта **cc_design_security_classification** применяют следующие глобальные правила из настоящего стандарта:

- product_version_requires_security_classification (см. 5.2.5.24);
- pdu_requires_secunty_classitication (см. 5.2.5.70).

5.2.4.1.13 Объект cc_desin_person_and_organization_assignment

Объект **cc_design_person_and_organization_assignment** связывает объект **person_and_organization** с конкретной частью данных об изделии.

EXPRESS-спецификация

```
*)
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;
(*
```

Определение атрибута

items — набор объектов **person_organization_item,** обозначающих конкретные объекты **change_request, start_request, configuration_item, product, product_definition_formation, contract** или **security_classification,** связанные с объектом **person_and_organization.**

Формальное утверждение

WR1 — должна быть выполнена функция **cc_design_person_and_-organization_correlation**, уточняющая роли лиц и организаций в части данных об изделии. Полное описание функции **cc_design_person_and_organization_correlation** приведено в 5.2.6.2.

Соответствующие глобальные правила

Для объекта **cc_design_person_and_organization_assignment** применяют следующие глобальные правила из настоящего стандарта:

- change_request_requires_person_organization (см. 5.2.5.7);
- start_request_requires_person_organization (см. 5.2.5.12);
- product_requires_person_organization (см. 5.2.5.21);
- product_version_requires_person_organization (см. 5.2.5.23);
- product_definition_requires_person_organization (см. 5.2.5.25);
- contract_requires_person_organization (см. 5.2.5.36);
- secunty_classification_requires_person_organization (см. 5.2.5.39);
- configuration_item_requires_person_organization (см. 5.2.5.64).

5.2.4.1.14 Объект cc_design_date_and_time_assignment

Объект **cc_design_date_and_time_assignment** связывает объект **date_and_time_assignment** с конкретной частью данных об изделии.

EXPRESS-спецификация

```
*)
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;
(*
```

Определение атрибута

items — набор объектов **date_time_item,** обозначающий конкретные объекты **product_definition, change_request, start_request, change, start_work, contract, security_classification, certification** или **approval_person_and_organization,** для которых заданы времена и даты.

Формальное определение

WR1 — должна быть выполнена функция **cc_design_date_time_correlation**, уточняющая роли дат и времен элементов данных об изделии. Полное описание функции **cc_design_date_time_correlation** приведено в 5.2.6.3.

Соответствующие глобальные правила

Для объекта **cc_design_date_and_time_assignment** применяют следующие глобальные правила из настоящего стандарта:

- change_request_requires_date_time (см. 5.2.5.8);
- change_requires_date_time (см. 5.2.5.10);
- start_request_requires_date_time (см. 5.2.5.13);
- start_work_requires_date_time (см. 5.2.5.15);
- product_definition_requires_date_time (см. 5.2.5.27);
- certification_requires_date_time (см. 5.2.5.30);
- security_classification_requires_date_time (см. 5.2.5.40);
- security_classification_optional_date_time (см. 5.2.5.41).

5.2.4.1.15 Объект cc_design_specification_reference

Объект **cc_design_specification_reference** связывает ссылку на спецификацию полного описания изделия или на его аспекты с формой этого изделия.

EXPRESS-спецификация

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

Определение атрибута

items — набор объектов **product_definition** и/или **shape_aspect**, для которых используют объекты **specification_reference**.

Соответствующее глобальное правило

Для объекта **cc_design_specification_reference** применяют следующее глобальное правило из настоящего стандарта:

- document_to_product_definition (см. 5.2.5.46).

5.2.4.2 *Модификации (изменения) объектов, импортированных в проект с управляемой конфигурацией*

5.2.4.2.1 Изменения объекта application_context

Основное описание объекта **application_context** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **application_context** применяют следующее глобальное правило из настоящего стандарта:

- application_context_requires_ap_definition (см. 5.2.5.1).

5.2.4.2.2 Изменение объекта application_protocol_definition

Основное описание объекта **application_protocol_definition** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **application_protocol_definition** применяют следующее глобальное правило из настоящего стандарта:

- application_context_requires_ap_definition (см. 5.2.5.1).

5.2.4.2.3 Изменение объекта product_context

Основное описание объекта **product_context** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **product_context** применяют следующее глобальное правило из настоящего стандарта:

- subtype_mandatory_product_context (см. 5.2.5.2).

5.2.4.2.4 Изменение объекта product_definition_context

Основное описание объекта **product_definition_context** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Описание объекта **product_definition_context** изменено следующим образом.

Атрибут **name** объекта **application_context_element** должен быть использован при изменении объекта **product_definition_context** для точного определения контекста, на который распространяется действие объекта **product_definition**.

5.2.4.2.5 Изменение объекта product_category

Основное описание объекта **product_category** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Описание объекта **product_category** изменено следующим образом.

В настоящем стандарте "деревья" из объектов **product_category** применяют для классификации изделий. Промежуточная вершина (узел) "дерева" может быть использована для выделения стандартных деталей из нестандартных (собственного изготовления). В этом случае атрибут **name** объекта **product_category** должен иметь значение "standard_part".

5.2.4.2.6 Изменение объекта product_related_product_category

Основное описание объекта **product_related_pruduct_category** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **product_related_product_category** применяют следующие глобальные правила из настоящего стандарта:

- restrict_product_category_value (см. 5.2.5.4);
- product_requires_product_category (см. 5.2.5.5).

5.2.4.2.7 Изменение объекта product

Основное описание объекта **product** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **product** применяют следующие глобальные правила из настоящего стандарта:

- product_requires_product_category (см. 5.2.5.5);
- product_requires_version (см. 5.2.5.20);
- product_requires_person_organization (см. 5.2.5.21).

5.2.4.2.8 Изменение объекта product_definition_formation

Основное описание объекта **product_definition_formation** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **product_definition_formation** применяют следующие глобальные правила из настоящего стандарта:

- product_requires_version (см. 5.2.5.20);
- product_version_requires_approval (см. 5.2.5.22);
- product_version_requires_person_organization (см. 5.2.5.23);
- product_version_requires_security_classification (см. 5.2.5.24);
- subtype_mandatory_product_definition_formation (см. 5.2.5.50).

5.2.4.2.9 Изменение объекта product_definition

Основное описание объекта **product_definition** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **product_definition** применяют следующие глобальные правила из настоящего стандарта:

- design_context_for_property (см. 5.2.5.3);
- product_definition_requires_person_organization (см. 5.2.5.25);
- product_definition_requires_approval (см. 5.2.5.26);
- product_definition_requires_date_time (см. 5.2.5.27);
- document_to_product_definition (см. 5.2.5.46).

5.2.4.2.10 Изменение объекта action_request_status

Основное описание объекта **action_request_status** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **action_request_status** применяют следующие глобальные правила из настоящего стандарта:

- restrict_action_request_status (см. 5.2.5.16);
- versioned_action_request_requires_status (см. 5.2.5.17).

5.2.4.2.11 Изменение объекта versioned_action_request

Основное описание объекта **versioned_action_request** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **versioned_action_request** применяют следующие глобальные правила из настоящего стандарта:

- versioned_action_request_requires_status (см. 5.2.5.17);
- versioned_action_request_requires_solution (см. 5.2.5.18).

5.2.4.2.12 Изменение объекта action_request_solution

Основное описание объекта **action_request_solution** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **action_request_solution** применяют следующее глобальное правило из настоящего стандарта:

- versioned_action_request_requires_solution (см. 5.2.5.18).

5.2.4.2.13 Изменение объекта certification_type

Основное описание объекта **certification_type** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **certification_type** применяют следующие глобальные правила из настоящего стандарта:

- restrict_certification_type (см. 5.2.5.29);
- dependent_instantiable_certification_type (см. 5.2.5.62).

5.2.4.2.14 Изменение объекта certification

Основное описание объекта **certification** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **certification** применяют следующие глобальные правила из настоящего стандарта:

- certification_requires_approval (см. 5.2.5.28);
- certification_requires_date_time (см. 5.2.5.30).

5.2.4.2.15 Изменение объекта approval_status

Основное описание объекта **approval_status** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **approval_status** применяют следующие глобальные правила из настоящего стандарта:

- restrict_approval_status (см. 5.2.5.34);
- dependent_instantiable_approval_status (см. 5.2.5.59).

5.2.4.2.16 Изменение объекта approval

Основное описание объекта **approval** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **approval** применяют следующие глобальные правила из настоящего стандарта:

- approvals_are_assigned (см. 5.2.5.31);
- approval_requires_approval_person_organization (см. 5.2.5.32);
- approval_requires_approval_date_time (см. 5.2.5.33).

5.2.4.2.17 Изменение объекта approval_assignment

Основное описание объекта **approval_assignment** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **approval_assignment** применяют следующее глобальное правило из настоящего стандарта:

- approvals_are_assigned (см. 5.2.5.31).

5.2.4.2.18 Изменение объекта approval_person_organization

Основное описание объекта **approval_person_organization** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **approval_person_organization** применяют следующие глобальные правила из настоящего стандарта:

- approval_person_organization_constraints (см. 5.2.5.79);
- approval_requires_approval_person_organization (см. 5.2.5.32).

5.2.4.2.19 Изменение объекта approval_date_time

Основное описание объекта **approval_date_time** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **approval_date_time** применяют следующие глобальные правила из настоящего стандарта:

- approval_date_time_constraints (см. 5.2.5.78);
- approval_requires_approval_date_time (см. 5.2.5.33).

5.2.4.2.20 Изменение объекта contract_type

Основное описание объекта **contract_type** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **contract_type** применяют следующие глобальные правила из настоящего стандарта:

- restrict_contract_type (см. 5.2.5.37);
- dependent_instantiable_contract_type (см. 5.2.5.61).

5.2.4.2.21 Изменение объекта contract

Основное описание объекта **contract** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **contract** применяют следующие глобальные правила из настоящего стандарта:

- contract_requires_approval (см. 5.2.5.35);
- contract_requires_person_organization (см. 5.2.5.36).

5.2.4.2.22 Изменение объекта security_classification_level

Основное описание объекта **security_classification_level** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **security_classification_level** применяют следующие глобальные правила из настоящего стандарта:

- restrict_security_classification_level (см. 5.2.5.42);
- dependent_instantiable_security_classification_level (см. 5.2.5.58).

5.2.4.2.23 Изменение объекта security_classification

Основное описание объекта **security_classification** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **security_classification** применяют следующие глобальные правила из настоящего стандарта:

- security_classification_requires_approval (см. 5.2.5.38);
- security_classification_requires_person_organization (см. 5.2.5.39);
- security_classification_requires_date_time (см. 5.2.5.40);
- security_classification_optional_date_time (см. 5.2.5.41).

5.2.4.2.24 Изменение объекта person_and_organization_role

Основное описание объекта **person_and_organization_role** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **person_and_organization_role** применяют следующие глобальные правила из настоящего стандарта:

- restrict_person_organization_role (см. 5.2.5.43);
- dependent_instantiable_person_and_organization_role (см. 5.2.5.56).

5.2.4.2.25 Изменение объекта person

Основное описание объекта **person** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Определение атрибута

Определение атрибута изменено следующим образом:

id — средства, позволяющие идентифицировать данный объект **person**.

Примечание — Однозначность данного атрибута должна быть обеспечена во всей области значений соответствующих данных. Данное положение может быть достаточно просто удовлетворено на уровне отдельной организации, страны или рабочей группы. В случае необходимости обеспечения глобальной или универсальной однозначности данного атрибута следует руководствоваться рекомендациями ГОСТ Р ИСО/МЭК 8824-1.

5.2.4.2.26 Изменение объекта date_time_role

Основное описание объекта **date_time_role** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **date_time_role** применяют следующие глобальные правила из настоящего стандарта:

- restrict_date_time_role (см. 5.2.5.44);
- dependent_instantiable_date_time_role (см. 5.2.5.55).

5.2.4.2.27 Изменение объекта document_type

Основное описание объекта **document_type** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **document_type** применяют следующие глобальные правила из настоящего стандарта:

- restrict_document_type (см. 5.2.5.45);
- dependent_instantiable_document_type (см. 5.2.5.60).

5.2.4.2.28 Изменение объекта measure_with_unit

Основное описание объекта **measure_with_unit** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **measure_with_unit** применяют следующее глобальное правило из настоящего стандарта:

- as_required_quantity (см. 5.2.5.47).

5.2.4.2.29 Изменение объекта global_unit_assigned_context

Основное описание объекта **global_unit_assigned_context** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **global_unit_assigned_context** применяют следующее глобальное правило из настоящего стандарта:

- global_unit_assignment (см. 5.2.5.48).

5.2.4.2.30 Изменение объекта action

Основное описание объекта **action** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **action** применяют следующее глобальное правило из настоящего стандарта:

- subtype_mandatory_action (см. 5.2.5.49).

5.2.4.2.31 Изменение объекта action_directive

Основное описание объекта **action_directive** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **action_directive** применяют следующее глобальное правило из настоящего стандарта:

- dependent_instantiable_action_directive (см. 5.2.5.57).

5.2.4.2.32 Изменение объекта date

Основное описание объекта **date** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **date** применяют следующее глобальное правило из настоящего стандарта:

- dependent_instantiable_date (см. 5.2.5.51).

5.2.4.2.33 Изменение объекта representation

Основное описание объекта **representation** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **representation** применяют следующее глобальное правило из настоящего стандарта:

- subtype_mandatory_representation (см. 5.2.5.74).

5.2.4.2.34 Изменение объекта representation_context

Основное описание объекта **representation_context** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **representation_context** применяют следующее глобальное правило из настоящего стандарта:

- subtype_mandatory_representation_context (см. 5.2.5.75).

5.2.4.2.35 Изменение объекта shape_representation

Основное описание объекта **shape_representation** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **shape_representation** применяют следующие глобальные правила из настоящего стандарта:

- dependent_instantiable_shape_representation (см. 5.2.5.52);
- subtype_mandatory_shape_representation (см. 5.2.5.73).

5.2.4.2.36 Изменение объекта context_dependent_shape_representation

Основное описание объекта **context_dependent_shape_representation** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Описание объекта **context_dependent_shape_representation** изменено следующим образом.

В настоящем стандарте объект **context_dependent_shape_representation** должен быть использован для обеспечения связи объекта **next_assembly_usage_occurrence**, определяющего применение компо-

нента в сборочной единице, с объектом **shape_representation_relationship**, определяющим положение и ориентацию формы компонента в описании формы сборочной единицы.

Примечание — Объект **context_dependent_shape_representation** может быть использован для связи форм компонентов и сборочных единиц независимо от типов применяемых для них объектов **shape_representation**. Конструкция объекта **mapped_item** также может быть использована для связи форм компонентов и сборочных единиц, но в настоящем стандарте применение объекта **mapped_item** требует наличия одинаковых типов для данного объекта **shape_representation**.

5.2.4.2.37 Изменение объекта named_unit

Основное описание объекта **named_unit** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **named_unit** применяют следующее глобальное правило из настоящего стандарта:

- dependent_instantiable_named_unit (см. 5.2.5.53).

5.2.4.2.38 Изменение объекта representation_item

Основное описание объекта **representation_item** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **representation_item** применяют следующее глобальное правило из настоящего стандарта:

- dependent_instantiable_representation_item (см. 5.2.5.54).

5.2.4.2.39 Изменение объекта product_definition_usage

Основное описание объекта **product_definition_usage** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **product_definition_usage** применяют следующее глобальное правило из настоящего стандарта:

- subtype_mandatory_product_definition_usage (см. 5.2.5.69).

5.2.4.2.40 Изменение объекта geometric_representation_item

Основное описание объекта **geometric_representation_item** приведено в ИСО 10303-42. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **geometric_representation_item usage** применяют следующее глобальное правило из настоящего стандарта:

- geometric_representation_item_3d (см. 5.2.5.71).

5.2.4.2.41 Изменение объекта parametric_representation_context

Основное описание объекта **parametric_representation_context** приведено в ГОСТ Р ИСО 10303-43. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **parametric_representation_context** применяют следующее глобальное правило из настоящего стандарта:

- dependent_instantiable_parametric_representation_context (см. 5.2.5.72).

5.2.4.2.42 Изменение объекта product_concept

Основное описание объекта **product_concept** приведено в ГОСТ Р ИСО 10303-44. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **product_concept** применяют следующее глобальное правило из настоящего стандарта:

- product_concept_requires_configuration_item (см. 5.2.5.63).

5.2.4.2.43 Изменение объекта configuration_item

Основное описание объекта configuration_item приведено в ГОСТ Р ИСО 10303-44. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **configuration_item** применяют следующие глобальные правила из настоящего стандарта:

- product_concept_requires_configuration_item (см. 5.2.5.63);
- configuration_item_requires_person_organization (см. 5.2.5.64);
- configuration_item_requires_approval (см. 5.2.5.67).

5.2.4.2.44 Изменение объекта effectivity

Основное описание объекта **effectivity** приведено в ГОСТ Р ИСО 10303-41. Уточнение данного описания приведено в настоящем стандарте.

Соответствующие глобальные правила

Для объекта **effectivity** применяют следующие глобальные правила из настоящего стандарта:

- subtype_mandatory_effectivity (см. 5.2.5.65);
- effectivity_requires_approval (см. 5.2.5.66).

5.2.4.2.45 Изменение объекта next_assembly_usage_occurrence

Основное описание объекта **next_assembly_usage_occurrence** приведено в ГОСТ Р ИСО 10303-44. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **next_assembly_usage_occurrence** применяют следующее глобальное правило из настоящего стандарта:

- coordinated_assembly_and_shape (см. 5.2.5.68).

5.2.4.2.46 Изменение объекта assembly_component_usage

Основное описание объекта **assembly_componet_usage** приведено в ГОСТ Р ИСО 10303-44. Уточнение данного описания приведено в настоящем стандарте.

Соответствующее глобальное правило

Для объекта **assembly_component_usage** применяют следующее глобальное правило из настоящего стандарта:

- acu_requires_security_classification (см. 5.2.5.70).

5.2.5 Правила в проекте с управляемой конфигурацией

5.2.5.1 *Правило application_context_requires_ap_definition*

Правило **application_context_requires_ap_definition** определяет, что на каждый экземпляр объекта **application_context** должна быть только одна ссылка из объекта **application_protocol_definition**, определенного в настоящем стандарте.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

application_context — идентифицирует набор всех экземпляров объектов **application_context**;

application_protocol_definition — идентифицирует набор всех экземпляров объектов **application_protocol_definition**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **application_context** должен существовать только один ссылающийся на него экземпляр объекта **application_protocol_definition**, устанавливающий область действия со значением 'config_control_design', соответствующим наименованию схемы **application_interpreted_model_schema_name**.

5.2.5.2 *Правило subtype_mandatory_product_context*

Правило **subtype_mandatory_product_context** определяет допустимое использование объекта **product_context**. Для задания **discipline_type** в **product_context** должен быть использован объект **mechanical_context.** Его использование гарантирует, что данные об изделии представлены с точки зрения машиностроения.

EXPRESS-спецификация

```
*)
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;
    (*
```

Описание аргумента

product_context — идентифицирует набор всех экземпляров объектов **product_context**.

Формальное утверждение

WR1 — каждый экземпляр объекта **product_context** должен быть представлен объектом **mechanical_context**.

5.2.5.3 *Правило design_context_for_property*

Правило **design_context_for_property** определяет, что проектирование рассматривается в контексте только тех стадий жизненного цикла изделия, для которых могут быть определены характеристики, задаваемые в настоящем стандарте. Структуры изделий могут быть определены на любой стадии жизненного цикла посредством контекста из объекта **product_definition.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

product_definition — идентифицирует набор всех экземпляров объекта **product_definition.**

Формальное утверждение

WR1 — для каждого экземпляра объекта **product_definition** справедливо, что если на этот экземпляр дана ссылка посредством атрибута **definition** объекта **property_definition** или этот экземпляр представлен атрибутом **related_product_definition** в экземпляре объекта **product_definition_relationship,** ссылка на который задана посредством атрибута **definition** объекта **property_definition,** то атрибут **frame_of_reference** экземпляра объекта **product_definition** должен ссылаться на **design_context.**

5.2.5.4 *Правило restrict_product_category_value*

Правило **restrict_product_category_value** определяет ограниченный набор значений, которые может принимать объект **product_category**, связанный с объектом **product**.

EXPRESS-спецификация

```
*)
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;
(*

Описание аргумента

product_related_product_category — идентифицирует набор всех экземпляров объектов **product_related_product_category**.

Формальное утверждение

WR1 — атрибут **name** объекта **product_related_product_category** должен иметь только следующие значения: "assembly", "detail", "inseparable_assembly", "customer_furnished_equipment", "cast", "coined", "drawn", "extruded", "forged", "formed", "machined", "molded", "rolled" или "sheared".

Определения ограничений значения атрибута

assembly — идентифицирует деталь, содержащую набор других деталей, собранных вместе с целью реализации выполнения конкретной функции.

detail — идентифицирует деталь нижнего уровня спецификации.

customer_furnished_equipment — идентифицирует деталь, поставляемую конструкторскому бюро заказчиком.

inseparable_assembly — идентифицирует деталь (сборку), которая после сборки не может быть разобрана без физического повреждения хотя бы одного своего компонента.

cast — идентифицирует класс деталей, изготавливаемых посредством заливки жидкого материала в форму и последующего его отверждения.

coined — идентифицирует класс деталей, изготавливаемых посредством подачи жидкого материала в замкнутую форму под высоким давлением, обеспечивающим полное заполнение всего объема формы.

drawn — идентифицирует класс деталей, изготавливаемых путем протягивания металлической заготовки через фильеру для придания ей заданной формы.

extruded — идентифицирует класс деталей, изготавливаемых путем обжатия материала посредством прессования с изменением формы исходной заготовки.

forged — идентифицирует класс деталей, изготавливаемых путем придания металлическому материалу требуемой формы под влиянием внешней силы при высокой температуре.

formed — идентифицирует класс деталей, изготавливаемых путем формовки, при которой металлический материал подвергается силовому воздействию, приводящему к изменению его формы;

machined — идентифицирует класс деталей, изготавливаемых с использованием механического (станочного) оборудования.

molded — идентифицирует класс деталей, изготавливаемых путем получения детали требуемой формы посредством нагрева исходного материала выше точки плавления с последующей его заливкой в соответствующую форму.

rolled — идентифицирует класс деталей, изготавливаемых путем прокатки материала между двумя валками для придания ему соответствующей формы, соответствующей конфигурации валков.

sheared — идентифицирует класс деталей, изготавливаемых путем изменения формы исходного материала посредством его обрезки.

5.2.5.5 *Правило product_requires_product_category*

Правило **product_requires_product_category** определяет требование, согласно которому на каждый объект **product** должна быть ссылка только из одного объекта **product_related_product_category,** определяющего принадлежность данного изделия к категориям "assembly", "inseparable_assembly", "detail" или "customer_furnished_equipment".

Примечание — Правило **product_requires_product_category** ограничивает отношение между объектами **product_related_product_category** и **product.** Данное правило определяет, что на объект **product** должна быть дана ссылка только из одного объекта **product_related_product_category,** значениями которого могут быть "assembly", "inseparable_assembly", "detail" или "customer_furnished_equipment". Правило не запрещает использование других экземпляров объекта **product_related_product_category**, имеющих значения атрибута **name**, отличные от перечисленных выше. Допустимые значения атрибута **name** объекта **product_related_product_category** определены в правиле **restrict_product_category_value.**

EXPRESS-спецификация

*)
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;
(*
```

Описания аргументов

product — идентифицирует набор всех экземпляров объектов **product**.

product_related_product_category — идентифицирует набор всех экземпляров объектов **product_related_product_category**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **product** должен существовать только один экземпляр объекта **product_related_product_category,** значениями атрибута **name** которого являются "assembly", "inseparable_assembly", "detail" или "customer_furnished_equipment".

5.2.5.6 *Правило change_request_requires_approval*

Правило **change_request_requires_approval** определяет, что на каждый экземпляр объекта **change_request** должна быть дана ссылка только из одного объекта **cc_design_approval.** Это правило устанавливает обязательность утверждения каждого предложения (заявки) на изменение.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

change_request — идентифицирует набор всех экземпляров объектов **change_request**;

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **change_request** должен существовать только один экземпляр объекта **cc_design_approval**, содержащий данный экземпляр объекта **change_request** в соответствующем наборе объектов **items**.

5.2.5.7 *Правило change_request_requires_person_organization*

Правило **change_request_requires_person_organization** определяет, что на любой объект **change_request** должна быть дана ссылка хотя бы из одного объекта **cc_design_person_and_organization_assignment.** Это правило определяет необходимость наличия для любого объекта **change_request** соответствующего адресата, принимающего данный объект. Смысл понятия адресата разъяснен в описании атрибута **role** объекта **person_and_organization_assignment**.

Примечание — Допустимость использования различных значений ролей при связывании объекта **person_and_organization_assignment** с различными объектами определяется функцией **cc_design_person_and_organization_correlation**. Эту функцию используют локально для объекта **cc_design_person_and_organization_assignment.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

change_request — идентифицирует набор всех экземпляров объектов **change_request**.

cc_design_person_and_organisation_assignment — идентифицирует набор всех экземпляров объектов **cc_design_person_and_organisation_assignment.**

Формальное утверждение

WR1 — для каждого экземпляра объекта **change_request** должен существовать хотя бы один экземпляр объекта **cc_design_person_and_organization_assignment,** содержащий экземпляр объекта **change_request** в соответствующем наборе объектов **items**.

5.2.5.8 *Правило change_request_requires_date_time*

Правило **change_request_requires_date_time** определяет, что на каждый экземпляр объекта **change_request** должна быть дана ссылка только из одного объекта **cc_design_date_and_time_assignment.** Это правило определяет необходимость наличия для любого объекта **change_request** соответствующей даты выпуска.

Примечание — Допустимость использования различных значений ролей при связывании **date_time_assignment** с разными объектами определяется функцией **cc_design_date_and_time_correlation**. Эту функцию используют локально для объекта **cc_design_date_and_time_assignment.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

change_request — идентифицирует набор всех экземпляров объектов **change_request**.

cc_design_date_and_time_assignment — идентифицирует набор всех экземпляров объектов **cc_design_date_and_time_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **change_request** должен существовать только один экземпляр объекта **cc_design_date_and_time_assignment**, содержащий экземпляр объекта **change_request** в соответствующем наборе объектов **items**.

5.2.5.9 *Правило change_requires_approval*

Правило **change_requires_approval** определяет, что на каждый экземпляр объекта **change** должна быть дана ссылка только из одного экземпляра объекта **cc_design_approval.** Это правило определяет необходимость утверждения любого изменения (**change**).

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

change — идентифицирует набор всех экземпляров объектов **change**.

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **change** должен существовать только один экземпляр объекта **cc_design_approval**, содержащий экземпляр объекта **change** в соответствующем наборе объектов **items**.

5.2.5.10 *Правило change_requires_date_time*

Правило **change_requires_date_time** определяет, что на каждый экземпляр объекта **change** должна быть дана ссылка только из одного объекта **cc_design_date_and_time_assignment.** Это правило определяет необходимость наличия для любого объекта **change** даты внесения конкретного изменения в проект.

Примечание — Допустимость использования различных значений ролей при связывании **date_time_assignment** с разными объектами определяется функцией **cc_design_date_and_time_correlation**. Эту функцию используют локально для объекта **cc_design_date_and_time_assignment.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

change — идентифицирует набор всех экземпляров объектов **change**.

cc design_date_and_time_assignment — идентифицирует набор всех экземпляров объектов **cc_design_date_and_time_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **change** должен существовать только один экземпляр объекта **cc_design_date_and_time_assignment**, содержащий экземпляр объекта **change** с наименованием роли "start_date" в соответствующем наборе объектов **items**.

5.2.5.11 *Правило start_request_requires_approval*

Правило **start_request_requires_approval** определяет, что на каждый экземпляр объекта **start_request** должна быть дана ссылка только из одного экземпляра объекта **cc_design_approval.** Это правило определяет необходимость наличия для любого объекта **start_request** соответствующего утверждения, санкционирующего начало проектирования.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

start_request — идентифицирует набор всех экземпляров объектов **start_request**.

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **start_request** должен существовать только один объект **cc_design_approval**, содержащий экземпляр данного объекта **start_request** в соответствующем наборе объектов **items**.

5.2.5.12 *Правило start_request_requires_person_organization*

Правило **start_request_requires_person_organization** определяет, что на кажый объект **start_request** должна быть дана хотя бы одна ссылка из объекта **cc_design_person_and_organization_assignment.** Это правило определяет необходимость наличия для любого объекта **start_request** соответствующего адресата, принимающего данный объект. Смысл понятия адресата разъяснен в описании атрибута **role** объекта **person_and_organization_assignment**.

Примечание — Допустимость использования различных значений ролей при связывании **person_and_organization_assignment** с разными объектами определяется функцией **cc_design_person_and_organization_correlation**. Эту функцию используют локально для объекта **cc_design_person_and_organization_assignment.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

start_request — идентифицирует набор всех экземпляров объектов **start_request.**

cc_design_person_and_organization_assignment — идентифицирует набор всех экземпляров объектов **cc_design_person_and_organization_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **start_request** должен существовать хотя бы один экземпляр объекта **cc_design_person_and_organization_assignment,** содержащий данный экземпляр объекта **start_request** в соответствующем наборе объектов **items**.

5.2.5.13 *Правило start_request_requires_date_time*

Правило **start_request_requires_date_time** определяет, что на каждый экземпляр объекта **start_request** должна быть ссылка только из одного объекта **cc_design_date_and_time_assignment.** Это правило определяет необходимость наличия для любого объекта **start_request** даты выпуска данного объекта.

Примечание — Допустимость использования различных значений ролей при связывании **date_time_assignment** с разными объектами определяется функцией **cc_design_date_and_time_correlation**. Эту функцию используют локально для объекта **cc_design_date_and_time_assignment.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

start_request — идентифицирует набор всех экземпляров объектов **start_request**.

cc_design_date_and_time_assignment — идентифицирует набор всех экземпляров объектов **cc_design_date_and_time_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **start_request** должен существовать только один экземпляр объекта **cc_design_date_and_time_assignment**, содержащий данный экземпляр объекта **start_request** в соответствующем наборе объектов **items**.

5.2.5.14 *Правило start_work_requires_approval*

Правило **start_work_requires_approval** определяет, что на каждый экземпляр объекта **start_work** должна быть дана ссылка только из одного экземпляра объекта **cc_design_approval.** Это правило определяет необходимость утверждения (санкции) начала (запуска) любого процесса проектирования.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

start_work — идентифицирует набор всех экземпляров объектов **start_work.**

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **start_work** должен существовать только один экземпляр объекта **cc_design_approval**, содержащий данный экземпляр объекта **start_work** в соответствующем наборе объектов **items.**

5.2.5.15 *Правило start_work_requires_date_time*

Правило **start_work_requires_date_time** определяет, что на каждый экземпляр объекта **start_work** должна быть дана ссылка только из одного объекта **cc_design_date_and_time_assignment.** Это правило определяет необходимость задания для любого объекта **start_work** даты начала работы над проектом.

Примечание — Допустимость использования различных значений ролей при связывании **date_time_assignment** с разными объектами определяется функцией **cc_design_date_and_time_correlation**. Эту функцию используют локально для объекта **cc_design_date_and_time_assignment.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

start_work — идентифицирует набор всех экземпляров объектов **start_work.**

cc_design_date_and_time_assignment — идентифицирует набор всех экземпляров объектов **cc_design_date_and_time_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра **start_work** должен существовать только один экземпляр объекта **cc_design_date_and_time_assignment,** содержащий экземпляр данного объекта **start_work** с наименованием роли "start_date" в соответствующем наборе объектов **items**.

5.2.5.16 *Правило restrict_action_request_status*

Правило **restrict_action_request_status** определяет допустимые значения для статуса объекта **action_request.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

action_request_status — идентифицирует набор всех экземпляров объектов **action_request_status**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **action_request_status** значениями атрибута статуса должны быть: "proposed", "in_work", "issued" или "hold".

Определения значений атрибутов

proposed — определяет завершение запроса **versioned_action_request** и его нахождение в состоянии ожидания проверки и авторизации.

in_work — определяет выработку запроса **versioned_action_request** для возможного его включения в проект.

issued — определяет, что запрос **versioned_action_request** был авторизован для включения в проект.

hold — определяет, что запрос **versioned_action_request** был проверен, но не получил одобрения для включения в проект.

5.2.5.17 *Правило versioned_action_request_requires_status*

Правило **versioned_action_request_requires_status** определяет, что каждый экземпляр объекта **versioned_action_request_requires** должен иметь только один статус. Статус объекта **versioned_action_request** определяется значением объекта **action_request_status**.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

versioned_action_request — идентифицирует набор всех экземпляров объекта **versioned_action_request.**

action_request_status — идентифицирует набор всех экземпляров объекта **action_request_status.**

Формальное утверждение

WR1 — для каждого экземпляра объекта **versioned_action_request** должен существовать только один экземпляр объекта **action_request_status,** содержащий значение атрибута **assigned_request,** соответствующее данному экземпляру объекта **versioned_action_request**.

5.2.5.18 *Правило versioned_action_request_requires_solution*

Правило **versioned_action_request_requires_solution** определяет, что каждый экземпляр запроса **versioned_action_request** должен иметь одно или несколько предложений по его реализации. Конкретное решение (реализация) для **versioned_action_request** определяется объектом **action_request_solution**.

EXPRESS-спецификация

```
*)
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;
*)
```

Описания аргументов

versioned_action_request — идентифицирует набор всех экземпляров объекта **versioned_action_request;**

action_request_solution — идентифицирует набор всех экземпляров объекта **action_request_solution**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **versioned_action_request** должен существовать только один экземпляр объекта **action_request_solution**, содержащий значение атрибута **request**, соответствующее данному объекту **versioned_action_request**.

5.2.5.19 *Правило unique_version_change_order_rule*

Правило **unique_version_change_order_rule** вызывает функцию, которая возвращает значение "true", если объект **change** изменяет несколько объектов **product_definition_formation**, а каждый измененный объект **product_definition_formation** является версией разных изделий. Это правило определяет, что один объект **change** не должен изменять более одной версии данного изделия (**product**), но может изменять несколько объектов **product_definition_formation**, если каждый **product_definition_formation** ссылается на разные изделия (**product**).

EXPRESS-спецификация

```
*)
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 ;
(*
```

Описание аргумента

change — идентифицирует набор всех экземпляров объектов **change**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **change** функция **unique_version_change_order** должна возвращать значение "true".

5.2.5.20 *Правило product_requires_version*

Правило **product_requires_version** определяет, что на каждый экземпляр объекта **product** должна быть дана ссылка по крайней мере из одного экземпляра объекта **product_definition_formation**. Это правило устанавливает, что каждое изделие может иметь одну или несколько версий.

EXPRESS-спецификация

```
*)
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 ;
(*
```

Описания аргументов

product — идентифицирует набор всех экземпляров объектов **product**;

product_definition_formation — идентифицирует набор всех экземпляров объекта **product_definition_formation**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **product** должен быть один или несколько экземпляров объекта **product_definition_formation,** в которых значения атрибута **of_product** должны соответствовать данному объекту **product**.

5.2.5.21 *Правило product_requires_person_organization*

Правило **product_requires_person_organization** определяет, что на каждый экземпляр объекта **product** должна быть дана ссылка из экземпляра объекта **cc_design_person_and_organization_assignment**. Это правило устанавливает, что для каждого изделия должен быть определен собственник (объект **design_owner**).

EXPRESS-спецификация

```
*)
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 ;
(*
```

Описания аргументов

product — идентифицирует набор всех экземпляров объектов **product.**

cc_design_person_and_organization_assignment — идентифицирует набор всех экземпляров объектов **cc_design_person_and_organization_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **product** должен существовать экземпляр объекта **cc_design_person_and_organization_assignment**, содержащий атрибут **items** с значением эквивалентным данному экземпляру объекта **product**.

Примечание — Ролью атрибута **person_and_organization** объекта **product** является "design_owner". Данная роль связана с формальным описанием функции, определенной в 5.2.6.2.

5.2.5.22 *Правило product_version_requires_approval*

Правило **product_version_requires_approval** определяет, что на каждый экземпляр объекта **product_definition_formation** должна быть дана ссылка только из одного экземпляра объекта **cc_design_approval**. Это правило устанавливает необходимость утверждения каждой версии проекта.

EXPRESS-спецификация

```
*)
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 ;
(*
```

Описания аргументов

product_definition_formation — идентифицирует набор всех экземпляров объектов **product_definition_formation**.

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **product_definition_formation** должен существовать только один экземпляр объекта **cc_design_approval**, содержащий данный экземпляр объекта **product_definition_formation** в соответствующем наборе объектов **items**.

5.2.5.23 *Правило product_version_requires_person_organization*

Правило **product_version_requires_person_organization** определяет, что на любой объект **prod-**

uct_definition_formation должна быть дана ссылка только из одного объекта **cc_design_person_and_organization_assignment**, имеющего роль автора, или из одного или нескольких объектов **cc_design_person_and_-organization_assignment**, имеющих роль поставщика детали (part_supplier) или проектировщика (design_supplier). Это правило устанавливает необходимость наличия для каждого объекта **product_definition_formation** автора или поставщика, отвечающего за создание или поставку конкретной версии проекта. Смысл понятий автор (creator), поставщик детали (part_supplier) и проектировщик (design_supplier) рассмотрен в 5.2.5.43 как часть описания правила **restrict_person_organization_role**.

Примечание — Допустимость использования различных значений ролей при связывании объекта **person_and_organization_assignment** с разными объектами определяется функцией **cc_design_person_and_organization_correlation.** Эту функцию используют локально для объекта **cc_design_person_and_organization_assignment.** См. описание функции в 5.2.6.2.

EXPRESS-спецификация

```
*)
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_assigniment |
            (pdf IN ccdpoa.items) AND
            (ccdpoa.role.name IN ['design_supplier', 'part_supplier' ])))  >= 1 ))) = 0;
END_RULE ;
(*
```

Описания аргументов

product_definition_formation — идентифицирует набор всех экземпляров объектов **product_definition_formation**.

cc_design_person_and_organization_assignment — идентифицирует набор всех экземпляров объектов **cc_design_person_and_organization_assignment**.

Формальные утверждения

WR1 — для каждого экземпляра объекта **product_definition_formation** должен существовать только один экземпляр объекта **cc_design_person_and_-organization_assignment**, содержащий экземпляр данного объекта **product_-definition_formation** в соответствующем наборе объектов **items**, атрибут **role** которого ссылается на объект **person_and_organization_role**, имеющий значение 'creator' для атрибута **name**.

WR2 — для каждого экземпляра объекта **product_derinition_formation** должен существовать по крайней мере один экземпляр объекта **cc_design_person_-and_organization_assignment**, содержащий экземпляр данного объекта **product_definition_formation** в соответствующем наборе объектов **items**, атрибут **role** которого ссылается на объект **person_and_organization_role**, имеющий значение 'design_supplier' или 'part_supplier' для атрибута **name**.

5.2.5.24 *Правило product_version_requires_security_classification*

Правило **product_version_requires_security_classification** определяет, что на каждый экземпляр объекта **product_definition_formation** должна быть дана ссылка только из одного экземпляра объекта **cc_design_security_classification**. Это правило устанавливает, что каждая версия проекта должна иметь свою классификацию защиты.

EXPRESS-спецификация

```
*)
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 ;
(*
```

Описания аргументов

product_definition_formation — идентифицирует набор всех экземпляров объектов **product_definition_formation**.

cc_design_security_classification — идентифицирует набор всех экземпляров объектов **cc_design_security_classification**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **product_definition_formation** должен существовать единственный экземпляр объекта **cc_design_security_classification**, содержащий данный экземпляр объекта **product_definition_formation** в соответствующем наборе объектов **items**.

5.2.5.25 *Правило product_definition_requires_person_organization*

Правило **product_definition_requires_person_organization** определяет, что на любой объект **product_definition** должна быть дана ссылка только из одного объекта **cc_design_person_and_organization_assignment**. Это правило определяет необходимость наличия для любого объекта **product_definition** автора (разработчика), отвечающего за конкретное описание проекта (конструкции). Смысл понятия автор (разработчик) пояснен в описании атрибута **role** объекта **person_and_organization_assignment**.

Примечание — Допустимость использования различных значений ролей при связывании **person_and_organization_assignment** с разными объектами определяется функцией **cc_design_person_and_organization_correlation**. Эту функцию используют локально для объекта **cc_design_person_and_organization_assignment.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

product_definition — идентифицирует набор всех экземпляров объектов **product definition**.

cc_design_person_and organization_assignment — идентифицирует набор всех экземпляров объектов **cc_design_person_and_organization_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **product_definition** должен существовать только один экземпляр объекта **cc_design_person_and_organization_assignment,** содержащий данный экземпляр объекта **product_definition** в соответствующем наборе объектов **items.**

5.2.5.26 *Правило product_definition_requires_approval*

Правило **product_definition_requires_approval** определяет, что на каждый экземпляр объекта **product_definition** должна быть дана ссылка только из одного экземпляра объекта **cc_design_approval.** Это правило определяет необходимость наличия утверждения любого определения проектируемого изделия (конструкции).

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

product_definition — идентифицирует набор всех экземпляров объектов **product_definition**.

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **product_definition** может существовать только один экземпляр объекта **cc_design_approval,** содержащий данный экземпляр объекта **product_definition** в соответствующем наборе объектов **items**.

5.2.5.27 *Правило product_definition_requires_date_time*

Правило **product_definition_requires_date_time** определяет, что на каждый экземпляр объекта **product_definition** должна быть дана ссылка только из одного объекта **cc_design_date_and_time_assignment.** Это правило устанавливает необходимость наличия для любого объекта **product_definition** даты создания соответствующего изделия.

Примечание — Допустимость использования различных значений ролей объекта при связывании **date_time_assignment** с разными объектами определяется функцией **cc_design_date_and_time_correlation**. Эту функцию используют локально для объекта **cc_design_date_and_time_assignment.**

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

product_definition — идентифицирует набор всех экземпляров объектов **product_definition**;

cc_design_date_and_time_assignment — идентифицирует набор всех экземпляров объектов **cc_design_date_and_time_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **product_definition** должен существовать только один экземпляр объекта **cc_design_date_and_time_assignment**, содержащий данный экземпляр объекта **product_definition** в соответствующем наборе объектов **items**.

5.2.5.28 *Правило certification_requires_approval*

Правило **certification_requires_approval** определяет, что на каждый экземпляр объекта **certification** должна быть дана ссылка только из одного экземпляра объекта **cc_design_approval**. Это правило устанавливает необходимость наличия сертификации (аттестации) любого поставщика конструкции (проекта) в целом или ее детали (составной части).

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

certification — идентифицирует набор всех экземпляров объектов **certification**.

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **certification** должен существовать только один экземпляр объекта **cc_design_approval**, содержащий данный экземпляр объекта **certification** в соответствующем наборе объектов **items**.

5.2.5.29 *Правило restrict_certification_type*

Правило **restrict_certification_type** определяет, что объект **certification** может быть связан только с типом "part_supplier" или "design_supplier".

EXPRESS-спецификация

```
*)
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;
*)
```

Описание аргумента

certification_type — идентифицирует набор всех экземпляров объектов **certification_type**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **certification_type** значением атрибута **kind** должно быть "design_supplier" или "part_supplier".

Определения значений атрибутов

design_supplier — идентифицирует поставщика части проекта.

part_supplier — идентифицирует поставщика детали.

5.2.5.30 *Правило certification_requires_date_time*

Правило **certification_requires_date_time** определяет, что на каждый экземпляр объекта **certification** должна быть дана ссылка только из одного объекта **cc_design_date_and_time_assignment.** Это правило устанавливает необходимость наличия даты, показывающей начало действия объекта **certification**.

Примечание — Допустимость использования различных значений ролей при связывании объекта **date_time_assignment** с разными объектами определяется функцией **cc_design_date_and_time_correlation**. Эту функцию используют локально для объекта **cc_design_date_and_time_assignment.** Определение функции приведено в 5.2.6.3

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

certification — идентифицирует набор всех экземпляров объектов **certification**.

cc_design_date_and_time_assignment — идентифицирует набор всех экземпляров объектов **cc_design_date_and_time_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **certification** должен существовать только один экземпляр объекта **cc_design_date_and_time_assignment**, содержащий данный экземпляр объекта **certification** в соответствующем наборе объектов **items**.

5.2.5.31 *Правило approvals_are_assigned*

Правило **approvals_are_assigned** определяет, что на каждый экземпляр объекта **approval** должна существовать хотя бы одна ссылка из экземпляра объекта **approval_assignment**.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

approval — идентифицирует набор всех экземпляров объектов **approval**.

approval_assignment — идентифицирует набор всех экземпляров объектов **approval_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **approval** должен существовать один или несколько экземпляров объекта **approval_assignment**, содержащих экземпляр данного объекта **approval** в атрибуте **assigned_approval**.

5.2.5.32 *Правило approval_requires_approval_person_organization*

Правило **approval_requires_approval_person_organization** определяет, что на каждый экземпляр объекта **approval** должна быть дана ссылка хотя бы из одного объекта **approval_person_organization**. Это правило устанавливает необходимость наличия утверждения соответствующего объекта одним или несколькими лицами из данной организации.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргументов

approval — идентифицирует набор всех экземпляров объектов **approval**.

approval_person_organization — идентифицирует набор всех экземпляров объектов **approval_person_organization**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **approval** должен существовать один или несколько экземпляров объекта **approval_person_organization**, содержащих экземпляр данного объекта **approval** в атрибуте **authorized_approval**.

5.2.5.33 *Правило approval_requires_approval_date_time*

Правило **approval_requires_approval_date_time** определяет, что на каждый экземпляр объекта **approval** должна быть дана ссылка только из одного объекта **approval_date_time**. Данное правило устанавливает необходимость наличия для любого объекта **approval** даты присвоения ему конкретного статуса.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

approval — идентифицирует набор всех экземпляров объектов **approval**.

approval_date_time — идентифицирует набор всех экземпляров объектов **approval_date_time**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **approval** должен существовать только один экземпляр объекта **approval_date_time,** содержащий экземпляр данного объекта **approval** в соответствующем атрибуте **dated_approval**.

5.2.5.34 *Правило restrict_approval_status*

Правило **restrict_approval_status** определяет, что значениями атрибута **approval_status** могут быть только “approved”, “not_yet_approved”, “disapproved” или “withdrawn”.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

approval — идентифицирует набор всех экземпляров объекта **approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **approval** значениями атрибута **status** должны быть “approved”, “not_yet_approved”, “disapproved” или “withdrawn”.

Определения значений атрибутов

approved — устанавливает, что проведена требуемая авторизация конкретной роли по утверждению части данных об изделии.

not_yet_approved — устанавливает, что проводится авторизация конкретной роли по утверждению части данных об изделии.

disapproved — устанавливает, что конкретная роль части данных об изделии не была авторизована.

withdrawn — устанавливает, что проведенная авторизация для конкретной роли по утверждению части данных об изделии была аннулирована.

5.2.5.35 *Правило contract_requires_approval*

Правило **contract_requires_approval** определяет, что на каждый экземпляр объекта **contract** должна быть дана ссылка только из одного экземпляра объекта **cc_design_approval.** Это правило устанавливает необходимость утверждения любого контракта по выполнению проектирования (конструирования).

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

contract — идентифицирует набор всех экземпляров объектов **contract**.

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **contract** должен существовать только один экземпляр объекта **cc_design_approval**, ссылающийся на данный экземпляр объекта **contract** в соответствующем наборе объектов **items**.

5.2.5.36 *Правило contract_requires_person_organization*

Правило **contract_requires_person_organization** определяет, что на каждый объект **contract** должна быть дана ссылка только из одного объекта **cc_design_person_and_organization_assignment.** Данное правило устанавливает, что за каждый контракт должно отвечать определенное лицо. Смысл понятия лица, отвечающего за контракт, разъяснен в описании атрибута **role** объекта **person_and_organization_assignment**.

Примечание — Допустимость использования различных значений ролей при связывании **person_and_organization_assignment** с разными объектами определяется функцией **cc_design_person_and_organization_correlation**. Эту функцию используют локально для объекта **cc_design_person_and_organization_assignment.** Определение функции приведено в 5.2.6.2.

EXPRESS-спецификация

```
*)
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
(*
```

Описания аргументов

contract — идентифицирует набор всех экземпляров объектов **contract**.

cc_design_person_and_organization_assignment — идентифицирует набор всех экземпляров объектов **cc_design_person_and_organization_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **contract** должен существовать только один экземпляр объекта **cc_design_person_and_organization_assignment**, содержащий данный экземпляр объекта **contract** в соответствующем наборе объектов **items**.

5.2.5.37 *Правило restrict_contract_type*

Правило **restrict_contract_type** определяет допускаемые типы контрактов. Это правило устанавливает для типов контрактов значение “fixed_price” или “cost_plus”.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

contract_type — идентифицирует набор всех экземпляров объектов **contract_type**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **contract_type** атрибут **kind** должен иметь значение “fixed_price” или “cost_plus”.

Определения значений атрибутов

fixed_price — идентифицирует контракт, в соответствии с которым устанавливается фиксированная оплата за выполняемые работы.

cost_plus — идентифицирует контракт (дополнительное соглашение), в соответствии с которым

устанавливается дополнительная оплата за выполняемые организацией работы помимо основного контракта.

5.2.5.38 *Правило security_classification_requires_approval*

Правило **security_classification_requires_approval** определяет, что на каждый экземпляр объекта **security_classification** должна быть дана ссылка только из одного экземпляра объекта **cc_design_approval**. Это правило устанавливает необходимость утверждения любого уровня конфиденциальности (секретности), присвоенного проекту.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

security_classification — идентифицирует набор всех экземпляров объектов **security_classification**.

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **security_classification** должен существовать только один экземпляр объекта **cc_design_approval**, содержащий данный экземпляр объекта **security_classification** в соответствующем наборе объектов **items**.

5.2.5.39 *Правило security_classification_requires_person_organization*

Правило **security_classification_requires_person_organization** определяет, что на каждый объект **security_classification** должна быть дана ссылка только из одного объекта **cc_design_person_and_organization_assignment.** Это правило устанавливает, что за каждый объект **security_classification** должно отвечать лицо, имеющее соответствующие полномочия. Смысл понятия ответственного лица разъяснен в описании атрибута **role** объекта **person_and_organization_assignment**.

Примечание — Допустимость использования различных значений ролей при связывании **person_and_organization_assignment** с разными объектами определяется функцией **cc_design_person_and_organization_correlation**. Эту функцию используют локально для объекта **cc_design_person_and_organization_assignment.** Определение функции приведено в 5.2.6.2.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

security_classification — идентифицирует набор всех экземпляров объектов **security_classification**.

cc_design_person_and_organization_assignment — идентифицирует набор всех экземпляров объектов **cc_design_person_and_organization_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **security_classification** должен существовать только один

экземпляр объекта **cc_design_person_and_organization_assignment**, содержащий данный экземпляр объекта **security_classification** в соответствующем наборе объектов **items**.

5.2.5.40 *Правило security_classification_requires_date_time*

Правило **security_classification_requires_date_time** определяет, что на каждый экземпляр объекта **security_classification** должна быть дана ссылка только из одного объекта **cc_design_date_and_time_assignment**, имеющего значение роли "classification_date". Это правило устанавливает необходимость наличия даты присвоения конкретного уровня конфиденциальности (секретности), соответствующего данному объекту **security_classification**.

Примечание — Допустимость использования различных значений ролей при связывании **date_time_assignment** с разными объектами определяется функцией **cc_design_date_and_time_correlation**. Эту функцию используют локально для объекта **cc_design_date_and_time_assignment.** Определение функции приведено в 5.2.6.3.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

security_classincation — идентифицирует набор всех экземпляров объектов **security_classification**.

cc_design_date_and_time_assignment — идентифицирует набор всех экземпляров объектов **cc_design_date_and_time_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **security_classification** должен существовать только один экземпляр объекта **cc_design_date_and_time_assignment**, содержащий данный экземпляр объекта **security_classification** в соответствующем наборе объектов **items**, и атрибут **role**, ссылающийся на экземпляр объекта **date_and_time_role,** имеющий значение атрибута **name** "classification_date".

5.2.5.41 *Правило security _classification_optional_date_time*

Правило **security_classification_optional_date_time** определяет, что на каждый экземпляр объекта **security_classification** может ссылаться один или ни одного экземпляра объекта **cc_design_date_and_time_assignment**, имеющего роль "declassification_date". Это правило устанавливает возможность наличия для любого объекта **security_classification** даты окончания действия присвоенного ему уровня конфиденциальности (секретности).

Примечание — Допустимость использования различных значений ролей объекта при связывании объекта **date_time_assignment** с разными объектами определяется функцией **cc_design_date_and_time_correlation**. Эту функцию используют локально для объекта **cc_design_date_and_time_assignment.** Определение функции приведено в 5.2.6.3

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

security_classification — идентифицирует набор всех экземпляров объектов **security_classification**.

cc_design_date_and_time_assignment — идентифицирует набор всех экземпляров объектов **cc_design_date_and_time_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **security_classification** должен существовать нулевой или один экземпляр объекта **cc_design_date_and_time_assignment**, содержащий экземпляр данного объекта **security_classification** в соответствующем наборе объектов **items** и имеющий атрибут **role**, ссылающийся на экземпляр объекта **date_and_time_role** с значением атрибута **name** "declassification_date".

5.2.5.42 *Правило restrict_security_classification_level*

Правило **restrict_security_classification_level** определяет допустимые уровни конфиденциальности (секретности). Это правило устанавливает следующие значения уровней конфиденциальности: "unclassified", "classified", "proprietary", "confidential", "secret" или "top_secret".

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

security_classification_level — идентифицирует набор всех экземпляров объектов **security_classification_level**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **security_classification_level** атрибут **name** должен содержать значения "unclassified", "classified", "proprietary", "confidential", "secret" или "top_secret".

Определения значений атрибутов

unclassified — идентифицирует уровень конфиденциальности без установления соответствующей защиты.

classified — идентифицирует необходимость указания уровня конфиденциальности без уточнения его значения.

proprietary — идентифицирует уровень конфиденциальности, согласно которому раскрытие информации о детали или проекте (конструкции) создает риск уменьшения рынка сбыта или снижения конкурентоспособности.

confidential — идентифицирует такой уровень конфиденциальности, согласно которому раскрытие информации о детали или проекте (конструкции) может угрожать национальной безопасности или безопасности компании.

secret — идентифицирует такой уровень конфиденциальности, согласно которому раскрытие информации о детали или проекте (конструкции) создает серьезную угрозу национальной безопасности или безопасности компании.

top_secret — идентифицирует такой уровень конфиденциальности, согласно которому раскрытие информации о детали или проекте (конструкции) создает исключительно серьезную угрозу национальной безопасности или безопасности компании.

5.2.5.43 *Правило restrict_person_organization_role*

Правило **restrict_person_organization_role** определяет допустимые роли для объектов **person_and_organization**. Это правило устанавливает следующие значения данных ролей: "request_recipient", "initiator", "part_supplier", "design_supplier", "configuration_manager", "contractor", "classification_officer", "creator" или "design_owner".

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

person_organization_role — идентифицирует набор всех экземпляров объектов **person_organization_role**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **person_organization_role** атрибут **name** должен иметь одно из следующих значений: "request_recipient", "initiator", "part_supplier", "design_supplier", "configuration_manager", "contractor", "classification_officer", "creator" или "design_owner".

Определения значений атрибутов

request_recipient — идентифицирует лицо, отвечающее в организации за получение **change_request** или **start_request** и проведение действий по этим запросам.

initiator — идентифицирует лицо, отвечающее в организации за создание **change_request** или **start_request**.

part_supplier — идентифицирует лицо, отвечающее в организации за поставку детали.

design_supplier — идентифицирует лицо, отвечающее в организации за проект (конструкцию) детали.

configuration_manager — идентифицирует лицо, отвечающее в организации за информацию о конфигурации проекта (конструкции).

contractor — идентифицирует лицо, отвечающее в организации за информацию, связанную с контрактом по проектированию.

classification_officer — идентифицирует лицо, отвечающее в организации за присвоение и снятие уровней конфиденциальности (секретности) для деталей.

creator — идентифицирует лицо, отвечающее в организации за разработку конкретного **product_definition_formation** или **product_definition**.

design_owner — идентифицирует лицо, отвечающее в организации за проект в целом, включая все аспекты проектируемого изделия (**product**).

5.2.5.44 *Правило restrict_date_time_role*

Правило **restrict_date_time_role** определяет допустимые роли для объектов **date_and_time**. Это правило устанавливает, что значениями ролей объектов **date_and_time** должны быть только "creation_date", "request_date", "release_date", "start_date", "contract_date", "certification_date", "sign_off_date", "classification_date" или "declassification_date".

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

date_time_role — идентифицирует набор всех экземпляров объектов **date_time_role**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **date_time_role** атрибут **name** должен иметь следующие значения: "creation_date", "request_date", "release_date", "start_date", "contract_date", "certification_date", "sign_off_date", "classification_date" или "declassification_date".

Определения значений атрибута

creation_date — идентифицирует дату и время создания версии проекта (конструкции) или его нового определения, взамен существующего.

request_date — идентифицирует дату и время получения запроса на проектирование.

release_date — идентифицирует дату и время первоначального выпуска проекта или внесения в него изменения.

start_date — идентифицирует дату и время начала работы над новым проектом или деятельности по внесению изменений в существующий проект.

contract_date — идентифицирует дату и время вступления в силу контракта на проектирование.

certification_date — идентифицирует дату и время проведения сертификации.

sign_off_date — идентифицирует дату и время авторизации утверждения.

classification_date — идентифицирует дату и время присвоения соответствующего уровня конфиденциальности (секретности).

declassification_date — идентифицирует дату и время аннулирования присвоенного уровня конфиденциальности (секретности).

5.2.5.45 *Правило restrict_document_type*

Правило **restrict_document_type** определяет допустимые типы документов. Это правило устанавливает, что значениями типов документов могут быть только: "material_specification", "process_specification", "design_specification", "surface_finish_specification", "cad_filename" или "drawing".

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

document_type — идентифицирует набор всех экземпляров объектов **document_type**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **document_type** атрибут **product_data_type** должен иметь следующие значения: "material_specification", "process_specification", "design_specification", "surface_finish_specification", "cad_filename" или "drawing".

Определения значений атрибута

material_specification — идентифицирует тип документа, устанавливающего требования к сырью (материалам), смесям или полуфабрикатам, используемым для изготовления детали.

process_specification — идентифицирует тип документа, устанавливающего требования к процессам обработки изделия или материала.

design_specification — идентифицирует тип документа, устанавливающего конструкционные требования к детали.

surface_finish_specification — идентифицирует тип документа, устанавливающего требования к текстуре или свойствам поверхности (защитных покрытий) при реализации конкретного процесса или к детали в целом.

cad_filename — идентифицирует тип документа, представленного в электронной форме в виде файла системы автоматизированного проектирования (САПР).

drawing — идентифицирует тип документа, представляющего графическую форму проекта детали(ей) (например чертеж).

5.2.5.46 *Правило document_to_product_definition*

Правило **document_to_product_definition** определяет допустимые связи между документами, сгруппированные на основе объектов **document_relationship** и **product_definition**. Это правило устанавливает, что только объекты **product_definition** могут иметь относящиеся к ним группы документов.

EXPRESS-спецификация

```
*)
RULE document_to_product_definition FOR
```

```
(cc_design_specification_reference);
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;
(*
```

Описания аргументов

cc_design_specification_reference — идентифицирует набор всех экземпляров объектов **cc_design_specification_reference**;

product_definition — идентифицирует набор всех экземпляров объектов **product_definition**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **cc_design_specification_reference**, если ссылающийся на него объект **document** использован в атрибуте **relating_document** объекта **document_relationship**, каждым элементом набора соответствующих объектов **items** должен быть объект **product_definition**.

5.2.5.47 *Правило as_required_quantity*

Правило **as_required_quantity** определяет использование типа **descriptive_measure** в объекте **measure_with_unit**. Значением строкового (STRING) типа объекта **descriptive_measure** всегда должно быть “as_required”. Это правило устанавливает требование к указанию необходимой величины в заданной единице измерения.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

measure_with_unit — идентифицирует набор всех экземпляров объектов **measure_with_unit**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **measure_with_unit**, если атрибут **value** представлен типом **descriptive_measure**, значением этого атрибута должно быть “as_required”.

5.2.5.48 *Правило global_unit_assignment*

Правило **global_unit_assignment** определяет единицы измерения, которые должны быть установлены для объекта **global_unit_assigned_context.** Это правило устанавливает, что каждый объект **global_unit_assigned_context** должен иметь три элемента в наборе **units** и содержать единицы измерения длины, плоских и телесных углов.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

global_unit assigned_context — идентифицирует набор всех экземпляров объектов **global_unit_assigned_context**.

Формальные утверждения

WR1 — для каждого экземпляра объекта **global_unit_assigned_contex** набор **units** должен иметь точно три элемента;

WR2 — для каждого экземпляра объекта **global_unit_assigned_context** первым элементом в наборе **units** должна быть единица измерения длины **length_unit,** вторым элементом — единица измерения плоских углов **plane_angle_unit** и третьим элементом — единица измерения телесных углов **solid_angle_unit**.

5.2.5.49 *Правило subtype_mandatory_action*

Правило **subtype_mandatory_action** определяет, что все объекты **action** должны быть представлены объектами **directed_action**.

EXPRESS-спецификация

```
*)
RULE subtype_mandatory_action FOR
    (action);
WHERE
    WR1: SIZEOF (QUERY (act <* action |
        NOT ('CONFIG_CONTROL_DESIGN.DIRECTED_ACTION' IN
        TYPEOF (act)))) = 0;
END_RULE;
(*
```

Описание аргумента

action — идентифицирует набор всех ограниченных экземпляров объектов **action**.

Формальное утверждение

WR1 — каждым экземпляром объекта **action** должен быть объект **directed_action**.

5.2.5.50 *Правило subtype_mandatory_product_definition_formation*

Правило **subtype_mandatory_product_definition_formation** определяет, что все объекты **product_definition_formation** должны быть представлены объектами **product_definition_formation_with_specified_source**.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

product_dennition_formation — идентифицирует набор всех ограниченных экземпляров объектов **product_definition_formation**.

Формальное утверждение

WR1 — каждый экземпляр объекта **product_definition_formation** должен быть представлен объектом **product_definition_formation_with_specified_source**.

5.2.5.51 *Правило dependent_instantiable_date*

Правило **dependent_instantiable_date** определяет, что все экземпляры объекта **date** связаны с определениями других объектов.

EXPRESS-спецификация

```
*)
RULE dependent_instantiable_date FOR (date);
WHERE
    WR1: SIZEOF (QUERY (dt <* date |
        NOT (SIZEOF (USEDIN (dt, ' ')) >= 1 ))) = 0;
END_RULE;
(*
```

Описание аргумента

date — идентифицирует набор всех экземпляров объектов **date**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **date** должна быть дана ссылка из атрибута другого объекта.

5.2.5.52 *Правило dependent_instantiable_shape_representation*

Правило **dependent_instantiable_shape_representation** определяет, что использование всех экземпляров объекта **shape_representation** зависит от определений других объектов.

EXPRESS-спецификация

```
*)
RULE dependent_instantiable_shape_representation FOR
    (shape_representation);
WHERE
    WR1: SIZEOF (QUERY (sr <* shape_representation |
        NOT (SIZEOF (USEDIN (sr, ' ')) >= 1 ))) = 0;
END_RULE ;
(*
```

Описание аргумента

shape_representation — идентифицирует набор всех экземпляров объектов **shape_representation**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **shape_representation** должна быть дана ссылка из атрибута другого объекта **shape_representation**.

5.2.5.53 *Правило dependent_instantiable_named_unit*

Правило **dependent_instantiable_named_unit** определяет, что использование всех экземпляров объекта **named_unit** зависит от определения других объектов.

EXPRESS-спецификация

```
*)
RULE dependent_instantiable_named_unit FOR (named_unit);
WHERE
    WR1: SIZEOF (QUERY (nu <* named_unit |
        NOT (SIZEOF (USEDIN (nu, ' ')) >= 1))) = 0;
END_RULE;
(*
```

Описание аргумента

named_unit — идентифицирует набор всех экземпляров объектов **named_unit**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **named_unit** должна быть дана ссылка из атрибута другого объекта.

5.2.5.54 *Правило dependent_instantiable_representation_item*

Правило **dependent_instantiable_representation_item** определяет, что все экземпляры объекта **representation_item** зависят от использования определений других объектов.

EXPRESS-спецификация

```
*)
RULE dependent_instantiable_representation_item FOR (representation_item);
WHERE
    WR1: SIZEOF (QUERY (ri <* representation_item |
        NOT (SIZEOF (USEDIN (ri, ' ')) >= 1 ))) = 0;
END_RULE;
(*
```

Описание аргумента

representation_item — идентифицирует набор всех экземпляров объектов **representation_item**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **representation_item** должна быть дана ссылка из атрибута другого объекта.

5.2.5.55 *Правило dependent_instantiable_date_time_role*

Правило **dependent_instantiable_date_time_role** устанавливает, что все экземпляры объектов **date_time_role** зависят от определения других объектов.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

date_time_role — идентифицирует набор всех экземпляров объектов **date_time_role.**

Формальное утверждение

WR1 — на каждый экземпляр объекта **date_time_role** должна быть дана ссылка из атрибута другого объекта.

5.2.5.56 *Правило dependent_instantiable_person_and_organization_role*

Правило **dependent_instantiable_person_and_organization_role** устанавливает, что использование всех экземпляров объекта **person_and_organization_role** зависит от определения других объектов.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

person_and_organization_role — идентифицирует набор всех экземпляров объектов **person_and_organization_role**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **person_and_organization_role** должна быть дана ссылка из атрибута другого объекта.

5.2.5.57 *Правило dependent_instantiable_action_directive*

Правило **dependent_instantiable_action_directive** устанавливает, что использование всех экземпляров объекта **action_directive** зависит от определения других объектов.

EXPRESS-спецификация

```
*)
RULE dependent_instantiable_action_directive FOR (action_directive) ;
WHERE
    WR1: SIZEOF (QUERY (ad <* action_directive |
        NOT (SIZEOF (USEDIN (ad, ' ' )) >= 1 ))) = 0;
END_RULE ;
(*
```

Определение аргумента

action directive — идентифицирует набор всех экземпляров объектов **action_directive**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **action_directive** должна быть дана ссылка из атрибута другого объекта.

5.2.5.58 *Правило dependent_instantiable_security_classification_level*

Правило **dependent_instantiable_security_classification_level** устанавливает, что использование всех экземпляров объекта **security_classification_level** зависит от определения других объектов.

EXPRESS-спецификация

```
*)
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 ;
(*
```

Определение аргумента

security_classification_level — идентифицирует набор всех экземпляров объектов **security_classification_level**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **security_classification_level** должна быть дана ссылка из атрибута другого объекта.

5.2.5.59 *Правило dependent_instantiable_approval_status*

Правило **dependent_instantiable_approval_status** устанавливает, что использование всех экземпляров объекта **approval_status_level** зависит от определения других объектов.

EXPRESS-спецификация

```
*)
RULE dependent_instantiable_approval_status FOR (approval_status);
WHERE
    WR1: SIZEOF (QUERY (ast <* approval_status |
        NOT (SIZEOF (USEDIN (ast, ' ' )) >= 1 ))) = 0;
END_RULE ;
(*
```

Определение аргумента

approval_status — идентифицирует набор всех экземпляров объектов **approval_status**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **approval_status** должна быть дана ссылка из атрибута другого объекта.

5.2.5.60 *Правило dependent_instantiable_document_type*

Правило **dependent_instantiable_document_type** устанавливает, что использование всех экземпляров объекта **document_type** зависит от определения других объектов.

EXPRESS-спецификация

```
*)
RULE dependent_instantiable_document_type FOR (document_type);
WHERE
    WR1: SIZEOF (QUERY (dt <* document_type |
        NOT (SIZEOF (USEDIN (dt, ' ' )) >= 1 ))) = 0;
END_RULE ;
(*
```

Определение аргумента

document type — идентифицирует набор всех экземпляров объектов **document_type**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **document_type** должна быть дана ссылка из атрибута другого объекта.

5.2.5.61 *Правило dependent_instantiable_contract_type*

Правило **dependent_instantiable_contract_type type** устанавливает, что использование всех экземпляров объекта **contract_type** зависит от определения других объектов.

EXPRESS-спецификация

```
*)
RULE dependent_instantiable_contract_type FOR (contract_type) ;
WHERE
    WR1: SIZEOF (QUERY (ct <* contract_type |
        NOT (SIZEOF (USEDIN (ct, ' ' )) >= 1 ))) = 0;
END_RULE ;
(*
```

Определение аргумента

contract_type — идентифицирует набор всех экземпляров объектов **contract_type**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **contract_type type** должна быть дана ссылка из атрибута другого объекта.

5.2.5.62 *Правило dependent_instantiable_certification_type*

Правило **dependent_instantiable_certification_type** устанавливает, что использование всех экземпляров объекта **certification_type** зависит от определения других объектов.

EXPRESS-спецификация

```
*)
RULE dependent_instantiable_certification_type FOR (certification_type);
WHERE
    WR1: SIZEOF (QUERY (ct <* certification_type |
        NOT (SIZEOF (USEDIN (ct, ' ' )) >= 1 ))) = 0;
END_RULE ;
(*
```

Определение аргумента

certification_type — идентифицирует набор всех экземпляров объектов **certification_type**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **certification_type** должна быть дана ссылка из атрибута другого объекта.

5.2.5.63 *Правило product_concept_requires_configuration_item*

Правило **product_concept_requires_configuration_item** определяет, что на каждый объект **product_concept** должна быть дана ссылка по крайней мере из одного объекта **configuration_item**. Это правило устанавливает необходимость наличия связи любого объекта **product_concept** по крайней мере с одним объектом **configuration_item**.

EXPRESS-спецификация

```
*)
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 ;
(*
```

Определения аргументов

product_concept — идентифицирует набор всех экземпляров объектов **product_concept.**

configuration_item — идентифицирует набор всех экземпляров объектов **configuration_item**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **product_concept** должен существовать по крайней мере один экземпляр объекта **configuration_item**, содержащий экземпляр объекта **product_concept** в качестве значения соответствующего атрибута **item_concept**.

5.2.5.64 *Правило configuration_item_requires_person_organization*

Правило **configuration_item_requires_person_organization** определяет, что на каждый объект **configuration_item** должна быть дана ссылка только из одного объекта **cc_design_person_and_organization_assignment.** Это правило устанавливает необходимость наличия для каждого объекта **configuration_item** соответствующего ответственного лица (**configuration_manager**). Понятие **configuration_manager** пояснено в атрибуте **role** объекта **person_organization_assignment**.

Примечание — Связь значений различных ролей **person_organization_assignment** для разных объектов определена в функции **cc_design_person_and_organization_correlation.** Эту функцию используют локально для объекта **cc_design_person_and_organization_assignment**. Описание функции приведено в 5.2.6.2.

EXPRESS-спецификация

```
*)
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_assigninent |
        ci IN ccdpoa.items )) = 1 ))) = 0;
END_RULE ;
(*
```

Определения аргументов

configuration_item — идентифицирует набор всех экземпляров объектов **configuration_item**;

cc_design_person_and_organization_assignment — идентифицирует набор всех экземпляров объектов **cc_design_person_and_organization_assignment**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **configuration_item** должен существовать только один экземпляр объекта **cc_design_person_organization_assignment**, содержащий данный экземпляр объекта **configuration_item** в соответствующем наборе объектов **items**.

5.2.5.65 *Правило subtype_mandatory_effectivity*

Правило **subtype_mandatory_effectivity** определяет, что каждый экземпляр объектов **effectivity** должен быть экземпляром сложного объекта, сопоставимым с одним из экземпляров объектов **serial_numbered_effectivity**, **dated_effectivity** или **lot_effectivity** и экземпляром объекта **configuration_effectivity**.

EXPRESS-спецификация

```
* )
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 ;
(*
```

Определение аргумента

effectivity — идентифицирует набор всех экземпляров объектов **effectivity**.

Формальное утверждение

WR1 — каждым экземпляром объектов **effectivity** должен быть экземпляр объектов **serial_numbered_effectivity**, **lot_effectivity** или **dated_effectivity** и **configuration_effectivity**.

5.2.5.66 *Правило effectivity_requires_approval*

Правило **effectivity_requires_approval** определяет, что на каждый экземпляр объекта **effectivity** должна быть дана ссылка только из одного объекта **cc_design_approval**. Это правило устанавливает, что каждый объект **effectivity** должен быть утвержден.

EXPRESS-спецификация

```
*)
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 ;
(*
```

Определения аргументов

effectivity — идентифицирует набор всех экземпляров объектов **effectivity**.

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **effectivity** должен существовать только один экземпляр объекта **cc_design_approval**, содержащий данный экземпляр объекта **effectivity** в соответствующем наборе объектов **items**.

5.2.5.67 *Правило configuration_item_requires_approval*

Правило **configuration_item_requires_approval** определяет, что на каждый экземпляр объекта **configuration_item** должна быть дана ссылка только из одного объекта **cc_design_approval**. Это правило устанавливает необходимость наличия утверждения каждого объекта **configuration_item**.

EXPRESS-спецификация

```
*)
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 ;
(*
```

Определения аргументов

configuration_item — идентифицирует набор всех экземпляров объектов **configuration_item**.

cc_design_approval — идентифицирует набор всех экземпляров объектов **cc_design_approval**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **configuration_item** должен существовать только один экземпляр объекта **cc_design_approval**, содержащий экземпляр данного объекта **configuration_item** в соответствующем наборе объектов **items**.

5.2.5.68 *Правило coordinated_assembly_and_shape*

Правило **coordinated_assembly_and_shape** определяет, что отношение между двумя объектами **product_definition**, представляющими сборочную единицу и компонент в объекте **next_assembly_usage_occurrence**, и отношение между двумя объектами **shape_representation**, содержащими представления форм сборочной единицы и компонентов в объекте **shape_representation_relationship** должны быть явно связаны с использованием объекта **context_dependent_shape_representation.** Это правило вызывает функцию **assembly_shape_is_defined**, возвращающую значение 'true", если отношения сборочной единицы и формы заданы явно.

EXPRESS-спецификация

```
*)
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, 'CONFIG_CONTROL_DESIGN'))))
            = 0;
END_RULE;
(*
```

Описание аргумента

next_assembly_usage_occurrence — идентифицирует набор всех экземпляров объектов **next_assembly_usage_occurrence**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **next_assembly_usage_occurrence** значением функции **assembly_shape_is_defined** должно быть 'true'.

5.2.5.69 *Правило subtype_mandatory_product_definition_usage*

Правило **subtype_mandatory_product_definition_usage** определяет, что все объекты **product_definition_usage** должны быть представлены как объекты **assembly_component_usage**.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

product_definition_usage — идентифицирует набор всех ограниченных экземпляров объектов **product_definition_usage**.

Формальное утверждение

WR1 — каждый экземпляр объекта **product_definition_usage** должен быть представлен как **assembly_component_usage**.

5.2.5.70 *Правило acu_requires_security_classification*

Правило **acu_requires_security_classification** определяет, что на каждый экземпляр объекта **assembly_component_usage** должна быть дана ссылка только из одного объекта **cc_design_security_classification.** Это правило устанавливает, что любому определению изделия из объекта **product_definition**, представленному атрибутом **related_product_definition** в объекте **assembly_component_usage**, должен быть присвоен уровень конфиденциальности в контексте его использования в сборочной единице, заданном объектом **assembly_component_usage**.

EXPRESS-спецификация

```
*)
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;
(*
```

Описания аргументов

assembly_component_usage — идентифицирует набор всех ограниченных экземпляров объектов **assembly_component_usage**.

cc_design_security_classification — идентифицирует набор всех ограниченных экземпляров объектов **cc_design_security_classification**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **assembly_component_usage** должен существовать только один экземпляр объекта **cc_design_security_classification**, содержащий данный экземпляр объекта **assembly_component_usage** в соответствующем наборе объектов **items**.

5.2.5.71 *Правило geometric_representation_item_3d*

Правило **geometric_representation_item_3d** определяет, что каждый объект **geometric_representation_item** должен быть создан на основе объекта **geometric_representation_context,** имеющего трехмерную размерность, за исключением случая, когда рассматриваемый объект используется для определения объекта **pcurve**. Это правило устанавливает, что вся геометрия соответствующих объектов должна быть трехмерной. Объект **pcurve** является единственным исключением, так как может иметь двухмерный контекст, представляющий параметрическое пространство поверхности.

EXPRESS-спецификация

```
*)
RULE geometric_representation_item_3d FOR
   (geometric_representation_item);
WHERE
   WR1: SIZEOF (QUERY (gri <* geometric_representation_item |
           NOT ((dimension_of (gri) = 3) OR
           (SIZEOF (QUERY (ur <* using_representations (gri) |
           'CONFIG_CONTROL_DESIGN.DEFINITIONAL_REPRESENTATION'
           IN TYPEOF (ur))) > 0 )))) = 0;
END_RULE;
(*
```

Описание аргумента

geometric_representation_item — идентифицирует набор всех ограниченных экземпляров объектов **geometric_representation_item**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **geometric_representation_item** значение атрибута **dim** должно быть равно трем, или объект **geometric_representation_item** должен быть использован как элемент в объекте **definitional_representation**.

Примечание — Объект **definitional_representation** предназначен для задания точек или кривых в параметрическом пространстве поверхности, используемом в объекте **pcurve**.

5.2.5.72 *Правило dependent_instantiable_parametric_representation_context*

Правило **dependent_instantiable_parametric_representation_context** определяет, что использование всех экземпляров объекта **parametric_representation_context** зависит от определений других объектов.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

parametric_representation_context — идентифицирует набор всех экземпляров объектов **parametric_representation_context**.

Формальное утверждение

WR1 — на каждый экземпляр объекта **parametric_representation_context** должна быть дана ссылка из атрибута другого объекта.

5.2.5.73 *Правило subtype_mandatory_shape_representation*

Правило **subtype_mandatory_shape_representation** требует, чтобы все объекты **shape_representation** были представлены объектами **geometrically_bounded_wireframe_shape_representation**, **geometrically_bounded_surface_shape_representation**, **edge_based_wireframe_shape_representation**, **shell_based_wireframe_shape_representation**, **manifold_surface_shape_representation**, **faceted_brep_shape_representation** или **advanced_brep_shape_representation**, или в соответствующем наборе объектов **items** содержали только объекты **axis2_placement_3d**, или являлись представлением объекта **shape_aspect** или отношением между двумя объектами **shape_aspect**. Это правило устанавливает ограничение на различные типы представления формы, разрешенные в настоящем стандарте.

EXPRESS-спецификация

```
*)
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.definition)) = 1 ))) = 0 ))) = 0;
END_RULE;
(*
```

Описание аргумента

shape_representation — идентифицирует набор всех ограниченных экземпляров объектов **shape_representation**.

Формальное утверждение

WR1 — каждый экземпляр объекта **shape_representation** должен быть представлен объектом **geometrically_bounded_wireframe_representation**, **geometrically_bounded_surface_representation**, **edge_based_wireframe_representation**, **shell_based_wireframe_representation**, **manifold_surface_with_topology_representation**, **faceted_brep_representation** или **advanced_brep_representation**, или в соответствующем наборе атрибута **items** содержать только объекты **axis2_placement_3d**, или быть представлением объектов **shape_aspect** или **shape_aspects_relationship**.

5.2.5.74 *Правило subtype_mandatory_representation*

Правило **subtype_mandatory_representation** требует, чтобы все объекты **representation** были представлены объектами **shape_representation**.

EXPRESS-спецификация

```
*)
RULE subtype_mandatory_representation FOR (representation);
WHERE
    WR1: SIZEOF (QUERY (rep <* representation |
         NOT ('CONFIG_CONTROL_DESIGN.SHAPE_REPRESENTATION' IN
         TYPEOF (rep)))) = 0;
END_RULE;
(*
```

Описание аргумента

representation — идентифицирует набор всех ограниченных экземпляров объектов **representation**.

Формальное утверждение

WR1 — каждый экземпляр объекта **representation** должен быть представлен объектом **shape_representation**.

5.2.5.75 *Правило subtype_mandatory_representation_context*

Правило **subtype_mandatory_representation_context** требует, чтобы все объекты **representation_context** были представлены посредством объектов **geometric_representation_context**.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

representation_context — идентифицирует набор всех ограниченных экземпляров объектов **representation_context**.

Формальное утверждение

WR1 — каждый экземпляр объекта **representation_context** должен быть представлен объектом **geometric_representation_context**.

5.2.5.76 *Правило no_shape_for_make_from*

Правило **no_shape_for_make_from** определяет, что объекты **product_derinition_relationship**, представляющие отношение "полуфабрикат—деталь" через объекты **design_make_from_relationship**, не должны иметь определенной формы. Это правило работает путем запрета на использование объекта **design_make_from_relationship** в объекте **product_definition_shape**, устанавливающем форму компонента, входящего в сборочную единицу.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

design_make_from_relationship — идентифицирует набор всех ограниченных экземпляров объекта **design_make_from_relationship**.

Формальное утверждение

WR1 — ни на один объект **design_make_from_relationship** не должна быть дана ссылка из атрибута **definition** объекта **property_definition,** представленного объектом **product_definition_shape**.

5.2.5.77 *Правило no_shape_for_supplied_part*

Правило **no_shape_for_supplied_part** определяет, что объекты **product_derinition_relationship,** представляющие отношение "деталь—покупная деталь" через объекты **supplied_part_relationship,** не должны иметь определенной формы. Это правило работает путем запрета на использование объекта **supplied_part_relationship** в объекте **product_definition_shape**, устанавливающем форму компонента, входящего в сборочную единицу.

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

supplied_part_relationship — идентифицирует набор всех ограниченных экземпляров объекта **supplied_part_relationship**.

Формальное утверждение

WR1 — ни на один объект **supplied_part_relationship** не должна быть дана ссылка из атрибута **definition** объекта **property_definition**, представленного объектом **product_definition_shape**.

5.2.5.78 *Правило approval_date_time_constraints*

Правило **approval_date_time_constraints** определяет, что каждый экземпляр объекта **approval_date_time** должен ссылаться только на экземпляр объекта **date_and_time**. Это правило устанавливает необходимость связи дат с конкретным временем.

EXPRESS-спецификация

```
*)
RULE approval_date_time_constraints FOR (approval_date_time) ;
WHERE
    WR1: SIZEOF (QUERY (adt <* approval_date_time |
        NOT (SIZEOF (TYPEOF(adt.date_time) *
        [ 'CONFIG_CONTROL_DESIGN.DATE_AND_TIME' ]) = 1 ))) =0 ;
END_RULE ;
(*
```

Описание аргумента

approval_date_time — идентифицирует набор всех экземпляров объектов **approval_date_time.**

Формальное утверждение

WR1 — для каждого экземпляра объекта **approval_date_time** из атрибута **date_time** должна быть дана ссылка на экземпляр объекта **date_and_time**.

5.2.5.79 *Правило approval_person_organization_constraints*

Правило **approval_person_organization_constraints** определяет, что каждый экземпляр объекта **approval_person_organization** должен иметь ссылку только на экземпляр объекта **person_and_organization**. Это правило устанавливает необходимость наличия связи конкретных лиц с какой-либо организацией.

EXPRESS-спецификация

```
*)
RULE approval_person_organization_constraints FOR
    (approval_person_organization) ;
WHERE
    WR1: SIZEOF (QUERY (apo <* approval_person_organization |
          NOT (SIZEOF (TYPEOF (apo.person_organization) *
          [ 'CONFIG_CONTROL_DESIGN.PERSON_AND_ORGANIZATION' ]) = 1 ))) =0 ;
END_RULE ;
(*
```

Описание аргумента

approval_person_organization — идентифицирует набор всех экземпляров объектов **approval_person_organization**.

Формальное утверждение

WR1 — для каждого экземпляра объекта **approval_person_organization** из атрибута **person_organization** должна быть дана ссылка на экземпляр объекта **person_and_organization**.

5.2.6 Функции проекта с управляемой конфигурацией

5.2.6.1 *Функция unique_version_change_order*

Булева функция **unique_version_change_order** принимает в качестве исходных данных (параметра) объект **action_execution** и возвращает значение "true", если функция **ordered_action**, вызываемая объектом **action_execution**, охватывает объекты **requested_action**, ссылающиеся на разные объекты **product_derinition_formation**, ссылающиеся в свою очередь на разные объекты **product**. Функция возвращает значение "false", если объекты **requested_action** ссылаются на объекты **product_derinition_formation**, охваченные объектами **action_execution**, а соответствующие объекты **product_derinition_formation** ссылаются на один и тот же объект **product**. Эта функция может возвращать значение "true", если посредством сложного объекта **change** изменяют разные версии отдельных деталей. Однократное изменение может не затрагивать разные версии конкретной детали.

EXPRESS-спецификация

```
*)
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;
    - - определяет набор объектов change_request, заданных для объектов
    - - versioned_action_request, объединенных данным объектом 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;
    - - конкретизирует объекты product_definition_formation, на которые дана ссылка
    - - из объектов change_request
    REPEAT k := 1 TO SIZEOF(assign);
       versions := versions + assign[k].items;
    END_REPEAT;
```

```
    - - проверяет отсутствие объекта product_definition_formation, ссылающегося на
    - - тот же экземпляр объекта product
    RETURN (SIZEOF (QUERY (vers <* versions |
      NOT (SIZEOF (QUERY (other_vers <* versions – vers |
      vers.of_product :=: other_vers.of_product)) = 0 ))) = 0);
END_FUNCTION;
(*
```

Описание аргумента

c — входной параметр, идентифицирующий проверяемый объект **directed_action**.

5.2.6.2 *Функция cc_design_person_and_organization_correlation*

Булева функция **cc_design_person_and_organization_correlation** возвращает значение "true", если значение атрибута **name** объекта **person_organization_role** согласовано с типом объекта, выбранного из набора в атрибуте **items** объекта **cc_design_person_and_organization_assignment**.

Эта функция устанавливает, что:

- лицо и организация, заданные в объектах **change_request** или **start_request**, могут иметь роль "request_recipient";

- лицо и организация, заданные в объектах **change_request**, **start_request**, **change** или **start_work**, могут иметь роль "initiator";

- лицо и организация, заданные в объектах **product_definition_formation** или **product_definition**, могут иметь роль "creator";

- лицо и организация, заданные в объекте **product_definition_formation**, могут иметь роль "part_supplier";

- лицо и организация, заданные в объекте **product_definition_formation**, могут иметь роль "design_supplier";

- лицо и организация, заданные в объекте **product,** могут иметь роль "design_owner";

- лицо и организация, заданные в объекте **configuration_item**, могут иметь роль "configuration_manager";

- лицо и организация, заданные в объекте **contract,** могут иметь роль "contractor";

- лицо и организация, заданные в объекте **security_classification**, могут иметь роль "classification_officer".

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

e — входной параметр, идентифицирующий проверяемый объект **cc_design_person_and_organization_assignment**.

5.2.6.3 *Функция cc_design_date_time_correlation*

Булева функция **cc_design_date_time_correlation** возвращает значение "true", если значение атрибута **name** объекта **date_time_role** согласовано с типом объекта, выбранного из набора атрибута **items** объекта **cc_design_date_and_time_assignment**.

Эта функция устанавливает, что:

- присвоенные значения даты и времени для объекта **product_definition** могут иметь роль "creation_date";
- присвоенные значения даты и времени для объектов **change_request** или **start_request** могут иметь роль "request_date";
- присвоенные значения даты и времени для объектов **change** или **start_work** могут иметь роль "release_date";
- присвоенные значения даты и времени для объектов **change** или **start_work** могут иметь роль "start_date";
- присвоенные значения даты и времени для объекта **approval_date_time** могут иметь роль "sign_off_date";
- присвоенные значения даты и времени для объекта **contract** могут иметь роль "contract_date";
- присвоенные значения даты и времени для объекта **certification** могут иметь роль "certification_date";
- присвоенные значения даты и времени для объекта **security_classification** могут иметь роль "classification_date";
- присвоенные значения даты и времени для объекта **security_classification** могут иметь роль "declassification_date".

EXPRESS-спецификация

```
*)
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;
(*
```

Описание аргумента

e — входной параметр, идентифицирующий проверяемый объект **cc_design_date_and_time_assignment.**

5.2.6.4 *Функция assembly_shape_is_defined*

Функция **assembly_shape_is_defined** принимает в качестве исходных данных (параметра) объект **next_assembly_usage_occurrence** и возвращает булевый результат. Функция возвращает значение "true", если определены формы для объектов **product_definition**, представленных объектам **related_product_definition** и **relating_product_definition**, в объекте **next_assembly_usage_occurrence**, а также две формы, связанные посредством объекта **shape_representation_relationship**, и два отношения, связанные через объект **context_dependent_shape_representation**. Функция также возвращает значение "true", если формы объектов **related_product_definition** или **relating_product_definition** не определены.

Функция возвращает значение "false" только в случае, если определены формы для объектов **related_product_definition** и **relating_product_definition** и эти формы связаны через объект **shape_representation_relationship**, но объекты **next_assembly_usage_occurrence** и **shape_representation_relationship** явно не связаны посредством объекта **context_dependent_shape_representation**.

EXPRESS-спецификация

```
*)
FUNCTION assembly_shape_is_defined (
                assay: next_assembly_usage_occurrence ;
                schema : STRING
    ) : BOOLEAN;
  LOCAL
    srr_set      : SET OF shape_representation_relationship := [];
    i            : INTEGER ;
    j            : INTEGER ;
    sdr_set      : SET OF shape_definition_representation : = [ ];
    prl_set      : SET OF property_definition := [ ];
    pdrel_set    : SET OF product_definition_relationship := [ ];
    pr2_set      : SET OF property_definition := [ ];
  END_LOCAL ;
  prl_set := bag_to_set (USEDIN (assy.related_product_definition, schema +
     '.PROPERTY_DEFINITION.DEFINITION' )) ;
  REPEAT i := 1 TO HIINDEX (prl_set) BY 1;
    sdr_set := sdr_set + QUERY ( pdr <* USEDIN (prl_set[i], schema +
         '.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | ((schema +
         '.SHAPE_DEFINITION_REPRESENTATION') IN TYPEOF (pdr) ));
  END_REPEAT;
   pdrel_set := bag_to_set (USEDIN (assy.related_product_definition, schema +
     '.PRODUCT_DEFINITION_RELATIONSHIP.' +
     'RELATED_PRODUCT_DEFINITION' )) ;
```

```
  REPEAT j := 1 TO HIINDEX (pdrel_set) BY 1;
    pr2_set := pr2_set + USEDIN (pdrel_set [j], schema +
        '.PROPERTY_DEFINITION.DEFINITION' ) ;
  END_REPEAT ;
  REPEAT i := 1 TO HIINDEX (pr2_set) BY 1;
    sdr_set := sdr_set + QUERY ( pdr <* USEDIN (pr2_set[i], schema +
        '.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | ((schema +
        '.SHAPE_DEFINITION_REPRESENTATION') IN TYPEOF(pdr) ));
  END_REPEAT ;
  IF SIZEOF (sdr_set) > 0 THEN
  REPEAT i := 1 TO HIINDEX (sdr_set) BY 1;
    srr_set : = QUERY ( rr <* bag_to_set (USEDIN (sdr_set [i]\
        property_definition_representation.used_representation, schema +
        '.REPRESENTATION_RELATIONSHIP.REP_2' )) | ((schema +
        '.SHAPE_REPRESENTATION_RELATIONSHIP' ) IN TYPEOF (rr) ));
    IF SIZEOF (srr_set) > 0 THEN
        REPEAT j : = 1 TO HIINDEX (srr_set) BY 1;
          IF SIZEOF (QUERY ( pdr <* bag_to_set (USEDIN (srr_set[j]\
              representation_relationship.rep_1, schema +
              '.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION'))
              | ((schema + '.SHAPE_DEFINITION_REPRESENTATION') IN TYPEOF (
              pdr)) ) * QUERY ( pdr <* bag_to_set (USEDIN (assy.
              relating_product_definition, schema +
              '.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) | ((
              schema + '.SHAPE_DEFINITION_REPRESENTATION')
              IN TYPEOF (pdr)) )) >= 1 THEN
            IF SIZEOF (QUERY ( cdsr <* USEDIN (srr_set[j], schema +
                '.CONTEXT_DEPENDENT_SHAPE_REPRESENTATION.' +
                'REPRESENTATION_RELATION') | (NOT (cdsr\
                context_dependent_shape_representation.
                represented_product_relation\property_definition.
                definition : = : assy)) ))  0 THEN RETURN (FALSE);
            END_IF ;
          END_IF ;
        END_REPEAT ;
      END_IF ;
    END_REPEAT ;
  END_IF;
  RETURN (TRUE) ;
END_FUNCTION; - - assembly_shape_is_defined
(*
```

Описание аргумента

assy — входной параметр, идентифицирующий объект **next_assembly_usage_occurrence**, отношения которого подлежат проверке.

EXPRESS-спецификация

```
*)
END_SCHEMA; — config_control_design
(*
```

6 Требования соответствия

Соответствие настоящему стандарту включает в себя удовлетворение требованиям, установленным в данном стандарте, требованиям к обеспечиваемым методам реализации и соответствующим требованиям из стандартов, указанных в разделе 2.

Любая реализация должна обеспечивать как минимум один из методов реализации, а именно определяемый ГОСТ Р ИСО 10303-21. Требования, относящиеся к методам реализации, приведены в приложении D.

Форма заявки о соответствии реализации протоколу (ЗСРП) содержит перечень вариантов (опций) или их комбинаций, которые могут быть включены в реализацию. Форма ЗСРП приведена в приложении С.

Примечание — Комплект абстрактных тестов, используемых для проверки соответствия настоящему стандарту, формулируется в рамках разработки проекта стандарта ИСО 10303-303. Процесс оценки соответствия описан в ГОСТ Р ИСО 10303-32.

В настоящем стандарте рекомендован некоторый набор опций, которые может обеспечивать реализация. Эти опции сгруппированы в классы соответствия. Определены 12 классов соответствия. Удовлетворение настоящему стандарту требует как минимум, соответствия классу 1a. Класс 1b является подмножеством класса 1a. Опции, определяемые классами 2—6, могут быть реализованы выборочно в дополнение к классу 1a или 1b. Соответствие конкретному классу означает, что должны поддерживаться все объекты, типы и налагаемые ограничения прикладной интерпретированной модели (ПИМ) данных об изделии, определенные как часть данного класса. Обеспечение конкретного класса соответствия требует поддержки всех опций, перечисленных в этом классе.

Классы соответствия характеризуются следующими признаками:

1a — обозначение изделия без формы;

1b — информация о проекте с управляемой конфигурацией без формы;

2a — класс 1a и форма, представленная геометрически ограниченными каркасными моделями, поверхностными моделями или теми и другими;

2b — класс 1b и форма, представленная геометрически ограниченными каркасными моделями, поверхностными моделями или теми и другими;

3a — класс 1a и форма, представленная каркасными моделями с топологией;

3b — класс 1b и форма, представленная каркасными моделями с топологией;

4a — класс 1a и форма, представленная множественными поверхностными моделями с топологией;

4b — класс 1b и форма, представленная множественными поверхностными моделями с топологией;

5a — класс 1a и форма, представленная фасеточными гранично заданными моделями (b-rep);

5b — класс 1b и форма, представленная фасеточными гранично заданными моделями (b-rep);

6a — класс 1a и форма, представленная усовершенствованными фасеточными гранично заданными моделями (advanced b-rep);

6b — класс 1b и форма, представленная усовершенствованными фасеточными гранично заданными моделями (advanced b-rep).

Класс 1a является общим основанием для классов 2—6. Если реализация соответствует любому классу 2—6, то она должна соответствовать и классу 1a.

Классы соответствия 2—6 определены в терминах подтипов геометрических объектов **shape_representation** ПИМ данных об изделии, служащих для определения форм, указанных в таблице 15. Каждый подтип объекта **shape_representation** задан в локальной области правил, содержащей все элементы ПИМ, определяющие данный класс соответствия.

Таблица 15 — Опции соответствия

Подтип объекта **shape_representation**	Класс				
	2	3	4	5	6
geometrically_bounded_surface_shape_representation	X				
geometrically_bounded_wireframe_shape_representation	X				
edge_based_wireframe_shape_representation		X			

Окончание таблицы 15

Подтип объекта **shape_representation**	Класс				
	2	3	4	5	6
shell_based_wireframe_shape_representation		X			
manifold_surface_shape_representation			X		
faceted_brep_shape_representation				X	
advanced_brep_shape_representation					X

6.1 Объекты класса соответствия 1a

Реализация класса соответствия 1a должна содержать следующие объекты и связанные с ними конструкции:

- application_context;
- application_context_element;
- application_protocol_definition;
- approval;
- approval_assignment;
- approval_date_time;
- approval_person_organization;
- approval_role;
- approval_status;
- calendar_date;
- cc_design_approval;
- cc_design_date_and_time_assignment;
- cc_design_person_and_organization_assignment;
- cc_design_security_classification;
- coordinated_universal_time_offset;
- date;
- date_and_time;
- date_and_time_assignment;
- date_and_time_role;
- design_context;
- local_time;
- mechanical_context;
- organization;
- person;
- person_and_organization;
- person_and_organization_assignment;
- person_and_organization_role;
- product;
- product_category;
- product_category_relationship;
- product_context;
- product_definition;
- product_definition_context;
- product_definition_formation;
- product_definition_formation_with_specified_source;
- product_deftnition_shape;
- product_related_product_category;
- property_definition;
- security_classification;
- security_classification_assignment;
- security_classification_level.

6.2 Объекты класса соответствия 1b

Реализация класса соответствия 1b должна содержать следующие объекты и связанные с ними конструкции:

- action;
- action_assignment;
- action_directive;
- action_method;
- action_request_assignment;
- action_request_solution;
- action_request_status;
- action_status;
- address;
- alternate_product_relationship;
- approval_relationship;
- area_measure_with_unit;
- area_unit;
- assembly_component_usage;
- assembly_component_usage_substitute;
- cc_design_certification;
- cc_design_contract;
- cc_design_specification_reference;
- certification;
- certification_assignment;
- certification_type;
- change;
- change_request;
- configuration_design;
- configuration_effectivity;
- configuration_item;
- context_dependent_unit;
- contract;
- contract_assignment;
- contract_type;
- conversion_based_unit;
- dated_effectivity;
- design_make_from_relationship;
- dimensional_exponents;
- directed_action;
- document;
- document_reference;
- document_relationship;
- document_type;
- document_usage_constraint;
- document_with_class;
- effectivity;
- executed_action;
- length_measure_with_unit;
- length_unit;
- lot_effectivity;
- mass_measure_with_unit;
- mass_unit;
- measure_with_unit;
- named_unit;
- next_assembly_usage_occurrence;
- ordinal_date;
- organization_relationship;
- organizational_address;

- organizational_project;
- personal_address;
- product_concept;
- product_concept_context;
- product_definition_effectivity;
- product_definition_relationship;
- product_definition_usage;
- product_definition_with_associated_documents;
- promissory_usage_occurrence;
- quantified_assembly_component_usage;
- serial_numbered_effectivity;
- shape_aspect;
- shape_aspect_relationship;
- si_unit;
- specified_higher_usage_occurrence;
- start_request;
- start_work;
- supplied_part_relationship;
- versioned_action_request;
- volume_measure_with_unit;
- volume_unit;
- week_of_year_and_day_date.

6.3 Объекты класса соответствия 2

Реализация класса соответствия 2 должна обеспечивать класс 1a или 1b (обозначенные как 2a, 2b), а также содержать следующие объекты и связанные с ними конструкции:

- axis1_placement;
- axis2_placement_2d;
- axis2_placement_3d;
- b_spline_curve;
- b_spline_curve_with_knots;
- b_spline_surface;
- b_spline_surface_with_knots;
- bezier_curve;
- bezier_surface;
- boundary_curve;
- bounded_curve;
- bounded_surface;
- cartesian_point;
- cartesian_transformation_operator;
- cartesian_transformation_operator_3d;
- circle;
- composite_curve;
- composite_curve_on_surface;
- composite_curve_segment;
- conic;
- conical_surface;
- context_dependent_shape_representation;
- curve;
- curve_bounded_surface;
- curve_replica;
- cylindrical_surface;
- definitional_representation;
- degenerate_pcurve;
- degenerate_toroidal_surface;
- direction;
- elementary_surface;
- ellipse;

- evaluated_degenerate_pcurve;
- functionally_defined_transformation;
- geometric_curve_set;
- geometric_representation_context;
- geometric_representation_item;
- geometric_set;
- geometrically_bounded_surface_shape_representation;
- geometrically_bounded_wireframe_surface_shape_representation;
- global_uncertainty_assined_context;
- global_unit_assigned_context;
- hyperbola;
- intersection_curve;
- item_defined_transformation;
- line;
- mapped_item;
- offset_curve_3d;
- offset_surface;
- outer_boundary_curve;
- parabola;
- parametric_representation_context;
- pcurve;
- placement;
- plane;
- plane_angle_measure_with_unit;
- plane_angle_unit;
- point;
- point_on_curve;
- point_on_surface;
- point_replica;
- polyline;
- property_definition_representation;
- quasi_uniform_curve;
- quasi_uniform_surface;
- rational_b_spline_curve;
- rational_b_spline_surface;
- rectangular_composite_surface;
- rectangular_trimmed_surface;
- reparametrised_composite_curve_segment;
- representation;
- representation_context;
- representation_item;
- representation_map;
- representation_relationship;
- representation_relationship_with_transformation;
- seam_curve;
- shape_aspect;
- shape_aspect_relationship;
- shape_definition_representation;
- shape_representation;
- shape_representation_relationship;
- solid_angle_measure_with_unit;
- solid_angle_unit;
- spherical_surface;
- surface;
- surface_curve;
- surface_of_linear_extrusion;
- surface_of_revolution;

- surface_patch;
- surface_replica;
- swept_surface;
- toroidal_surface;
- trimmed_curve;
- uncertainty_measure_with_unit;
- uniform_curve;
- uniform_surface;
- vector.

6.4 Объекты класса соответствия 3

Реализация класса соответствия 3 должна обеспечивать класс 1a или 1b (обозначенные как 3a, 3b), а также содержать следующие объекты и связанные с ними конструкции:

- axis2_placement_3d;
- b_spline_curve;
- b_spline_curve_with_knots;
- bezier_curve;
- bounded_curve;
- cartesian_point;
- cartesian_transformation_operator;
- cartesian_transformation_operator_3d;
- circle;
- conic;
- connected_edge_set;
- context_dependent_shape_representation;
- curve;
- curve_replica;
- direction;
- edge;
- edge_based_wireframe_model;
- edge_based_wireframe_shape_representation;
- edge_curve;
- edge_loop;
- ellipse;
- functionally_defined_transformation;
- geometric_representation_context;
- geometric_representation_item;
- global_uncertainty_assigned_context;
- global_unit_assigned_context;
- hyperbola;
- item_defined_transformation;
- line;
- loop;
- mapped_item;
- offset_curve_3d;
- oriented_edge;
- parabola;
- path;
- placement;
- plane_angle_measure_with_unit;
- plane_angle_unit;
- point;
- point_replica;
- polyline;
- property_definition_representation;
- quasi_uniform_curve;
- rational_b_spline_curve;
- representation;

- representation_context;
- representation_item;
- representation_map;
- representation_relationship;
- representation_relationship_with_transformation;
- shape_aspect;
- shape_aspect_relationship;
- shape_definition_representation;
- shape_representation;
- shape_representation_relationship;
- shell_based_wireframe_model;
- shell_based_wireframe_shape_representation;
- solid_angle_measure_with_unit;
- solid_angle_unit;
- topological_representation_item;
- uncertainty_measure_with_unit;
- uniform_curve;
- vector;
- vertex;
- vertex_loop;
- vertex_point;
- vertex_shell;
- wire_shell.

6.5 Объекты класса соответствия 4

Реализация класса соответствия 4 должна обеспечивать класс 1a или 1b (обозначенные как 4a, 4b), а также содержать следующие объекты и связанные с ними конструкции:

- advanced_face;
- axis1_placement;
- axis2_placement_2d;
- axis2_placement_3d;
- b_spline_curve;
- b_spline_curve_with_knots;
- b_spline_surface;
- b_spline_surface_with_knots;
- bezier_curve;
- bezier_surface;
- bounded_curve;
- bounded_surface;
- cartesian_point;
- cartesian_transformation_operator;
- cartesian_transformation_operator_3d;
- circle;
- closed_shell;
- composite_curve;
- composite_curve_on_surface;
- composite_curve_segment;
- conic;
- conical_surface;
- connected_face_set;
- context_dependent_shape_representation;
- curve;
- curve_replica;
- cylindrical_surface;
- definitional_representation;
- degenerate_pcurve;
- degenerate_toroidal_surface;
- direction;

- edge;
- edge_curve;
- edge_loop;
- elementary_surface;
- ellipse;
- evaluated_degenerate_pcurve;
- face;
- face_bound;
- face_outer_bound;
- face_surface;
- functionally_defined_transformation;
- geometric_representation_context;
- geometric_representation_item;
- global_uncertainty_assigned_context;
- global_unit_assigned_context;
- hyperbola;
- intersection_curve;
- item_defined_transformation;
- line;
- loop;
- manifold_surface_shape_representation;
- mapped_item;
- offset_curve_3d;
- offset_surface;
- open_shell;
- oriented_closed_shell;
- oriented_edge;
- oriented_face;
- oriented_open_shell;
- oriented_path;
- parabola;
- parametric_representation_context;
- path;
- pcurve;
- placement;
- plane;
- plane_angle_measure_with_unit;
- plane_angle_unit;
- point;
- point_on_curve;
- point_on_surface;
- polyline;
- property_definition_representation;
- quasi_uniform_curve;
- quasi_uniform_surface;
- rational_b_spline_curve;
- rational_b_spline_surface;
- representation;
- representation_context;
- representation_item;
- representation_map;
- representation_relationship;
- representation_relationship_with_transformation;
- seam_curve;
- shape_aspect;
- shape_aspect_relationship;
- shape_definition_representation;

- shape_representation;
- shape_representation_relationship;
- shell_based_surface_model;
- solid_angle_measure_with_unit;
- solid_angle_unit;
- spherical_surface;
- surface;
- surface_curve;
- surface_of_linear_extrusion;
- surface_of_revolution;
- surface_replica;
- swept_surface;
- topological_representation_item;
- toroidal_surface;
- uncertainty_measure_with_unit;
- uniform_curve;
- uniform_surface;
- vector;
- vertex;
- vertex_loop;
- vertex_point.

6.6 Объекты класса соответствия 5

Реализация класса соответствия 5 должна обеспечивать класс 1a или 1b (обозначенные как 5a, 5b), а также содержать следующие объекты и связанные с ними конструкции:

- axis2_placement_3d;
- brep_with_voids;
- cartesian_point;
- cartesian_transformation_operator;
- cartesian_transformation_operator_3d;
- closed_shell;
- connected_face_set;
- context_dependent_shape_representation;
- direction;
- edge;
- elementary_surface;
- face;
- face_bound;
- face_outer_bound;
- face_surface;
- faceted_brep;
- faceted_brep_shape_representation;
- functionally_defined_transformation;
- geometric_representation_context;
- geometric_representation_item;
- global_uncertainty_assigned_context;
- global_unit_assigned_context;
- item_defined_transformation;
- loop;
- manifold_solid_brep;
- mapped_item;
- open_shell;
- oriented_closed_shell;
- oriented_edge;
- oriented_face;
- oriented_open_shell;
- oriented_path;
- path;

- placement;
- plane;
- plane_angle_measure_with_unit;
- plane_angle_unit;
- point;
- poly_loop;
- property_definition_representation;
- representation;
- representation_context;
- representation_item;
- representation_map;
- representation_relationship;
- representation_relationship_with_transformation;
- shape_aspect;
- shape_aspect_relationship;
- shape_definition_representation;
- shape_representation;
- shape_representation_relationship;
- solid_angle_measure_with_unit;
- solid_angle_unit;
- solid_model;
- surface;
- topological_representation_item;
- uncertainty_measure_with_unit;
- vector;
- vertex.

6.7 Объекты класса соответствия 6

Реализация класса соответствия 6 должна обеспечивать класс 1a или 1b (обозначенные как 6a, 6b), а также содержать следующие объекты и связанные с ними конструкции:

- advanced_brep_shape_representation;
- advanced_face;
- axis1_placement;
- axis2_placement_2d;
- axis2_placement_3d;
- b_spline_curve;
- b_spline_curve_with_knots;
- b_spline_surface;
- b_spline_surface_with_knots;
- bezier_curve;
- bezier_surface;
- bounded_curve;
- bounded_surface;
- brep_with_voids;
- cartesian_point;
- cartesian transformation_operator;
- cartesian_transformation_operator_3d;
- circle;
- closed_shell;
- composite_curve;
- composite_curve_on_surface;
- composite_curve_segment;
- conic;
- conical_surface;
- connected_face_set;
- context_dependent_shape_representation;
- curve;
- cylindrical_surface;

- definitional_representation;
- degenerate_toroidal_surface;
- direction;
- edge;
- edge_curve;
- edge_loop;
- elementary_surface;
- ellipse;
- face;
- face_bound;
- face_outer_bound;
- face_surface;
- functionally_defined_transformation;
- geometric_representation_context;
- geometric_representation_item;
- global_uncertainty_assigned_context;
- global_unit_assigned_context;
- hyperbola;
- item_defined_transformation;
- line;
- loop;
- manifold_solid_brep;
- mapped_item;
- open_shell;
- oriented_closed_shell;
- oriented_edge;
- oriented_face;
- oriented_open_shell;
- oriented_path;
- parabola;
- parametric_representation_context;
- path:
- pcurve;
- placement;
- plane;
- plane_angle_measure_with_unit;
- plane_angle_unit;
- point;
- polyline;
- property_definition_representation;
- quasi_uniform_curve;
- quasi_uniform_surface;
- rational_b_spline_curve;
- rational_b_spline_surface;
- representation;
- representation_context;
- representation_item;
- representation_map;
- representation_relationship;
- representation_relationship_with_transformation;
- shape_aspect;
- shape_aspect_relationship;
- shape_definition_representation;
- shape_representation;
- shape_representation_relationship;
- solid_angle_measure_with_unit;
- solid_angle_unit;

- solid_model;
- spherical_surface;
- surface;
- surface_curve;
- surface_of_linear_extrusion;
- surface_of_revolution;
- swept_surface;
- topological_representation_item;
- toroidal_surface;
- uncertainty_measure_with_unit;
- uniform_curve;
- uniform_surface;
- vector;
- vertex;
- vertex_loop;
- vertex_point.

ПРИЛОЖЕНИЕ А
(обязательное)

Развернутый листинг ПИМ (AIM) на языке EXPRESS

```
*)
SCHEMA config_control_design;

    CONSTANT
    dummy_gri : geometric_representation_item := representation_item(' ') ||
                    geometric_representation_item ( ) ;
    dummy_tri : topological_representation_item := representation_item(' ')
                    || topological_representation_item( );
    END_CONSTANT ;

    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
         (polyline_form,
          circular_arc,
          elliptic_arc,
          parabolic_arc,
          hyperbolic_arc,
          unspecified) ;
    END_TYPE; - - b_spline_curve_form

    TYPE b_spline_surface_form = ENUMERATION OF
         (plane_surf,
          cylindrical_surf,
          conical_surf,
          spherical_surf,
          toroidal_surf,
```

```
        surf_of_revolution,
        ruled_surf,
        generalised_cone,
        quadric_surf,
        surf_of_linear_extrusion,
        unspecified) ;
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,
     local_time,
     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 founded_item_select = SELECT
    (founded_item,
     representation_item) ;
END_TYPE; - - founded_item_select

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,
     unspecified,
     quasi_uniform_knots,
     piecewise_bezier_knots) ;
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
    (length_measure,
     mass_measure,
     plane_angle_measure,
     solid_angle_measure,
     area_measure,
     volume_measure,
     parameter_value,
     context_dependent_measure,
     descriptive_measure,
     positive_length_measure,
     positive_plane_angle_measure,
     count_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,
     organization,
     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
     (curve_3d,
      pcurve_s1,
      pcurve_s2) ;
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
     (vertex_shell,
      wire_shell,
      open_shell,
      closed_shell) ;
END_TYPE; - - shell

TYPE si_prefix = ENUMERATION OF
     (exa,
      peta,
      tera,
      giga,
      mega,
      kilo,
      hecto,
      deca,
      deci,
      centi,
      milli,
```

```
        micro,
        nano,
        pico,
        femto,
        atto) ;
END_TYPE; - - si_prefix

TYPE si_unit_name = ENUMERATION OF
     (metre,
      gram,
      second,
      ampere,
      kelvin,
      mole,
      candela,
      radian,
      steradian,
      hertz,
      newton,
      pascal,
      joule,
      watt,
      coulomb,
      volt,
      farad,
      ohm,
      siemens,
      weber,
      tesla,
      henry,
      degree_celsius,
      lumen,
      lux,
      becquerel,
      gray,
      sievert) ;
END_TYPE; - - si_unit_name

TYPE solid_angle_measure = REAL;
END_TYPE; - - solid_angle_measure

TYPE source = ENUMERATION OF
     (made,
      bought,
      not_known) ;
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,
     continuous,
     cont_same_gradient,
     cont_same_gradient_same_curvature) ;
END_TYPE; - - transition_code

TYPE trimming_preference = ENUMERATION OF
    (cartesian,
     parameter,
     unspecified) ;
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_reguest_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) | (NOT (
                SIZEOF (QUERY ( fcs <* csh\connected_face_set.cfs_faces | (
                NOT ( 'CONFIG_CONTROL_DESIGN.ADVANCED_FACE'
                IN TYPEOF(fcs))) )) = 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 ( 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\oriented_edge.edge_element))) )) = 0)) )) = 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 (SIZEOF(['CONFIG_CONTROL_DESIGN.LINE',
                'CONFIG_CONTROL_DESIGN.CONIC',
                'CONFIG_CONTROL_DESIGN.POLYLINE' ,
                'CONFIG_CONTROL_DESIGN.SURFACE_CURVE' ,
                'CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE'] * TYPEOF(oe.
```

```
            edge_element\edge_curve.edge_geometry)) = 1)) )) = 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.edge_start)) AND (
            'CONFIG_CONTROL_DESIGN.CARTESIAN_POINT' IN TYPEOF(oe\edge.
            edge_start\vertex_point.vertex_geometry)) AND (
            'CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN TYPEOF(oe\edge.
            edge_end)) AND ('CONFIG_CONTROL_DESIGN.CARTESIAN_POINT' IN
            TYPEOF(oe\edge.edge_end\vertex_point.vertex_geometry)))) ))
            = 0)) )) = 0 );
      wr5 : (SIZEOF (QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
            'CONFIG_CONTROL_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
            | ('CONFIG_CONTROL_DESIGN.ORIENTED_PATH' IN TYPEOF(
            elp_fbnds. bound)) )) = 0);
      wr6 : ((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));
      wr7 : (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\face_bound.bound\vertex_loop.loop_vertex)) AND (
            'CONFIG_CONTROL_DESIGN.CARTESIAN_POINT' IN TYPEOF (vlp_fbnds
            \face_bound.bound\vertex_loop.loop_vertex\vertex_point.
            vertex_geometry)))) )) = 0);
      wr8 : (SIZEOF (QUERY ( bnd <* bounds | (NOT (SIZEOF ([
            'CONFIG_CONTROL_DESIGN.EDGE_LOOP' ,
            'CONFIG_CONTROL_DESIGN.VERTEX_LOOP'] * TYPEOF(bnd.bound)) =
            1)) )) = 0) ;
      wr9 : (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 | (('CONFIG_CONTROL_DESIGN.SURFACE_CURVE' IN
            TYPEOF (oe\oriented_edge.edge_element\edge_curve.
            edge_geometry)) AND (NOT (SIZEOF(QUERY ( sc_ag <* oe.
            edge_element\edge_curve.edge_geometry\surface_curve.
            associated_geometry | (NOT ('CONFIG_CONTROL_DESIGN.PCURVE'
            IN TYPEOF(sc_ag))) )) = 0))) )) = 0)) )) = 0);
      wr10: (((NOT ('CONFIG_CONTROL_DESIGN.SWEPT_SURFACE' IN TYPEOF(
            face_geometry))) OR (NOT ('CONFIG_CONTROL_DESIGN.POLYLINE'
            IN TYPEOF(face_geometry\swept_surface.swept_curve))) OR (
            SIZEOF (face_geometry\swept_surface.swept_curve\polyline.
            points) >= 3)) AND (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 | ((
            'CONFIG_CONTROL_DESIGN.POLYLINE' IN TYPEOF(oe\oriented_edge
            .edge_element\edge_curve.edge_geometry)) AND (NOT (SIZEOF(
            oe\oriented_edge.edge_element\edge_curve.edge_geometry\
            polyline.points) >= 3))) )) = 0)) )) = 0));
END_ENTITY; - - advanced_face

ENTITY alternate_product_relationship;
      name        : label;
      definition  : text;
```

```
        alternate    : product;
        base         : product;
        basis        : text;
    UNIQUE
        url : alternate, base;
    WHERE
        wrl: (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;
      definition :  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), dummy_gri ||
           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,
    bounded_pcurve, bounded_surface_curve, composite_curve) )
```

```
    SUBTYPE OF (curve) ;
END_ENTITY; - - bounded_curve

ENTITY bounded_pcurve
    SUBTYPE OF (pcurve, bounded_curve);
    WHERE
      wr1: ('CONFIG_CONTROL_DESIGN.BOUNDED_CURVE' IN TYPEOF(SELF\pcurve.
              reference_to_curve.items [1] )) ;
END_ENTITY; - - bounded_pcurve

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 bounded_surface_curve
    SUBTYPE OF (surface_curve, bounded_curve) ;
    WHERE
      wr1: ('CONFIG_CONTROL_DESIGN.BOUNDED_CURVE' IN TYPEOF(SELF\
              surface_curve.curve_3d)) ;
END_ENTITY; - - bounded_surface_curve

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 (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_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\geometric_representation_item.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 certitication_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
   SUBTYPE OF (founded_item) ;
      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
    url         : configuration, usage, id;
  WHERE
    wr1 : ('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
    url         : 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_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 degenerate_toroidal_surface
   SUBTYPE OF (toroidal_surface);
      select_outer       : BOOLEAN;
   WHERE
      wr1        : (major_radius < minor_radius);
END_ENTITY; - - degenerate_toroidal_surface

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
    url         : 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.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.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.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.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.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.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))
               )) = 0)) )) = 0)) )) = 0) ;
      wr7: (SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.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) AND valid_wireframe_vertex_point (edges.
            edge_end\vertex_point.vertex_geometry))) )) = 0)) )) = 0))
            )) = 0 ) ;
      wr8: (SIZEOF (QUERY ( mi <* QUERY ( it <* SELF.items | (
            'CONFIG_CONTROL_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT
            (('CONFIG_CONTROL_DESIGN.' +
            'EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION') IN TYPEOF(mi\
            mapped_item.mapping_source.mapped_representation))) )) = 0);
      wr9 : ( SELF.context_of_items\geometric_representation_context.
            coordinate_space_dimension = 3);
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) | (NOT ( SIZEOF (
          QUERY ( fcs <* csh\connected_face_set.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\elementary_surface.
          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) | (NOT (SIZEOF(
          QUERY ( fcs <* csh\connected_face_set.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 founded_item;
    END_ENTITY; - - founded_item

    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, poly_loop, vertex_point, 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 <* SELF.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 <* SELF.items | (SIZEOF([
         'CONFIG_CONTROL_DESIGN.GEOMETRIC_SET' ,
         'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1) )) >
         0) ;
    wr3: (SIZEOF (QUERY ( mi <* QUERY ( it <* SELF.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))
         AND (SIZEOF(QUERY ( mr_it <* mi\mapped_item.mapping_source.
         mapped_representation.items | (
         'CONFIG_CONTROL_DESIGN.GEOMETRIC_SET' IN TYPEOF (mr_it)) )) >
         0))) )) = 0);
    wr4: (SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.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)) )) = 0)) )) = 0);
    wr5: (SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.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)) )) = 0)) )) = 0);
    wr6: (SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.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)) ))
         = 0)) )) = 0);
    wr7: (SIZEOF (QUERY ( gs <* QUERY ( it <* SELF.items | (
         'CONFIG_CONTROL_DESIGN.GEOMETRIC_SET' IN TYPEOF (it)) ) | (
         SIZEOF(QUERY ( gsel <* gs\geometric_set.elements | (
         'CONFIG_CONTROL_DESIGN.SURFACE' IN TYPEOF(gsel)) )) > 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.items | (NOT (SIZEOF(TYPEOF(it) * [
         'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET' ,
         'CONFIG_CONTROL_DESIGN.AXIS2_PLACEMENT_3D',
         'CONFIG_CONTROL_DESIGN.MAPPED_ITEM']) = 1)) )) = 0);
    wr2: (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF(TYPEOF(it) * [
         'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET' ,
         'CONFIG_CONTROL_DESIGN.MAPPED_ITEM']) = 1) )) >= 1);
    wr3: (SIZEOF(QUERY ( gcs <* QUERY ( it <* SELF.items | (
         'CONFIG_CONTROL_DESIGN.GEOMETRIC_CURVE_SET' IN TYPEOF (it)) )
         | (NOT (SIZEOF(QUERY ( crv <* QUERY ( elem <* gcs\
         geometric_set.elements | ('CONFIG_CONTROL_DESIGN.CURVE' IN
         TYPEOF (elem)) ) | (NOT valid_geometrically_bounded_wf_curve(
         crv)) )) = 0)) ) ) = 0);
    wr4: (SIZEOF (QUERY ( gcs <* QUERY ( it <* SELF.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)) )) = 0)) )) = 0);
    wr5: (SIZEOF (QUERY ( gcs <* QUERY ( it <* SELF.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.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 ( it <* SELF.items | (
            'CONFIG_CONTROL_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT
            (('CONFIG_CONTROL_DESIGN.' +
            'GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_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 <* SELF.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 <* SELF.items | (SIZEOF([
          'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' ,
          'CONFIG_CONTROL_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1) ))
          > 0) ;
    wr3 : (SIZEOF (QUERY ( mi <* QUERY ( it <* SELF.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)) AND (SIZEOF(QUERY ( mr_it <* mi\
        mapped_item.mapping_source.mapped_representation.items | (
        'CONFIG_CONTROL_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
        TYPEOF (mr_it)) )) > 0))) )) = 0);
wr4 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.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.ORIENTED_CLOSED_SHELL' ,
        'CONFIG_CONTROL_DESIGN.CLOSED_SHELL'] * TYPEOF(sh)) = 1))
        )) = 0)) )) = 0);
wr5 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.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\connected_face_set.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 <* SELF.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\connected_face_set.
        cfs_faces | ('CONFIG_CONTROL_DESIGN.FACE_SURFACE' IN
        TYPEOF (fa)) ) | (NOT ((
        'CONFIG_CONTROL_DESIGN.ADVANCED_FACE' INTYPEOF(f_sf)) OR (
        SIZEOF (['CONFIG_CONTROL_DESIGN.B_SPLINE_SURFACE' ,
        'CONFIG_CONTROL_DESIGN.ELEMENTARY_SURFACE' ,
        'CONFIG_CONTROL_DESIGN.OFFSET_SURFACE' ,
        'CONFIG_CONTROL_DESIGN.SURFACE_REPLICA' ,
        'CONFIG_CONTROL_DESIGN.SWEPT_SURFACE'] * TYPEOF(f_sf\
        face_surface.face_geometry)) = 1))) )) = 0)) )) = 0)) )) = 0) ;
wr7 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.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\connected_face_set.cfs_faces | (NOT ((
        'CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (fa)) OR
        msf_surface_check(fa\face_surface.face_geometry))) ))
        = 0)) )) = 0)) )) = 0);
wr8 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.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\connected_face_set.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 <* SELF.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\connected_face_set.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\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 <* SELF.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\connected_face_set.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\
        path.edge_list | ('CONFIG_CONTROL_DESIGN.EDGE_CURVE' IN
        TYPEOF(oe.edge_element)) ) | (NOT (SIZEOF([
        'CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE' ,
        'CONFIG_CONTROL_DESIGN.CONIC' ,
        'CONFIG_CONTROL_DESIGN.CURVE_REPLICA' ,
        'CONFIG_CONTROL_DESIGN.LINE',
        'CONFIG_CONTROL_DESIGN.OFFSET_CURVE_3D' ,
        'CONFIG_CONTROL_DESIGN.PCURVE' ,
        'CONFIG_CONTROL_DESIGN.POLYLINE' ,
        '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 <* SELF.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\connected_face_set.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\path.edge_list | (
        NOT msf_curve_check (oe.edge_element\edge_curve.
        edge_geometry)) )) = 0)) )) = 0))) ))
         = 0)) )) = 0)) )) = 0);
wr12: (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.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\connected_face_set.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\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 <* SELF.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\connected_face_set.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\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) ;
      wr14: (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.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\connected_face_set.cfs_faces | (NOT ((
            'CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF(fa)) OR (
            SIZEOF (QUERY ( vlp_fbnds <* QUERY ( bnds <* fa.bounds | (
            'CONFIG_CONTROL_DESIGN.VERTEX_LOOP'
            IN TYPEOF (bnds.bound)) )
            | (NOT ('CONFIG_CONTROL_DESIGN.VERTEX_POINT' IN TYPEOF (
            vlp_fbnds\vertex_loop.loop_vertex))) )) = 0))) )) = 0)) ))
            = 0)) )) = 0 );
      wr15: (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.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\connected_face_set.cfs_faces | (NOT ((
            'CONFIG_CONTROL_DESIGN.ADVANCED_FACE' IN TYPEOF (fa)) OR (
            SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* fa.bounds | (
            'CONFIG_CONTROL_DESIGN.VERTEX_LOOP'
            IN TYPEOF (bnds.bound)) )
            | (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 (
            vlp_fbnds\vertex_loop.loop_vertex\vertex_point.
            vertex_geometry)) = 1)) )) = 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_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)
            AND (day_component <= 365)) OR (leap_year(SELF.
            year_component) AND (1 <= day_component) AND (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
    url         : usage, id;
END_ENTITY; - - product_definition_effectivity

ENTITY product_definition_formation;
    id              : identifier;
    description     : text;
    of_product      : product;
  UNIQUE
    url         ; 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: (NOT ('CONFIG_CONTROL_DESIGN.SHAPE_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') ;
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 > vl)));
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.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.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.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.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.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)) )) = 0)) )) = 0)) )) = 0)) )) =
        0) ;
wr6 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.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.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) AND
        valid_wireframe_vertex_point (el.edge_element.edge_end\
        vertex_point.vertex_geometry))) )) = 0)) ))
        = 0)) )) = 0)) )) = 0) ;
wr8 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.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);
wr9 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.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)) )) =
        0)) )) = 0)) )) = 0 );
wr10: (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.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) ;
      wr11: (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.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)) )) = 0)) )) = 0);
      wr12: (SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
            'CONFIG_CONTROL_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) ) | (
            NOT (('CONFIG_CONTROL_DESIGN.' +
            'SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION') IN TYPEOF (mi \
            mapped_item.mapping_source.mapped_representation))) ))
              = 0) ;
      wr13 : (SELF.context_of_items\geometric_representation_context.
            coordinate_space_dimension = 3);
            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 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 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 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) ANDOR
      bounded_surface_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 : = dummy_gri || curve ( ) || line (axis_position.
                location, dummy_gri || vector(axis_position.z, 1));
END_ENTITY; - - surface_of_revolution

ENTITY surface_patch
  SUBTYPE OF (founded_item);
    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, vertex_shell,
    wire_shell, connected_edge_set, connected_face_set, 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) OR (TYPEOF(trim_1[1]) < >
  TYPEOF (trim_1 [2])));
    wr2: ((HIINDEX(trim_2) = 1) OR (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 : valid_measure_value (SELF\measure_with_unit.value_component) ;
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 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 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_date_time_constraints FOR (approval_date_time) ;
WHERE
    wr1: (SIZEOF (QUERY ( adt <* approval_date_time | (NOT (SIZEOF(TYPEOF(
         adt.date_time) * ['CONFIG_CONTROL_DESIGN.DATE_AND_TIME'])
         = 1)) )) = 0);
END_RULE; - - approval_date_time_constraints

RULE approval_person_organization_constraints FOR (
         approval_person_organization) ;
WHERE
    wr1: (SIZEOF (QUERY( apo <* approval_person_organization | (NOT (
         SIZEOF (TYPEOF(apo.person_organization) * [
         'CONFIG_CONTROL_DESIGN.PERSON_AND_ORGANIZATION' ])
         = 1)) )) = 0 );
END_RULE; - - approval_person_organization_constraints

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, 'CONFIG_CONTROL_DESIGN')) )) = 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_parametric_representation_context FOR (
                parametric_representation_context ) ;
```

```
WHERE
    wr1: (SIZEOF(QUERY ( prc <* parametric_representation_context | (NOT (
            SIZEOF(USEDIN(prc,' ')) >= 1)) )) = 0);
END_RULE; - - dependent_instantiable_parametric_representation_context

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) ;
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 ( ceda <* cc_design_approval | (eff IN ceda.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(QUERY ( ur <*
                using_representations (gri) | (
                'CONFIG_CONTROL_DESIGN.DEFINITIONAL_REPRESENTATION' IN TYPEOF (
                ur)) )) > 0))) )) = 0);
    END_RULE; - - geometric_representation_item_3d

    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 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 ( ceda <* cc_design_approval | (pd IN ceda.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_reguires_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_reguires_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_reguires_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.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_reguires_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 [1:?] OF product_definition;
                  specific_relation: STRING): LOGICAL;
  LOCAL
      x : SET OF product_definition_relationship;
  END_LOCAL ;
  IF relation.relating_product_definition IN relatives THEN
      RETURN (FALSE) ;
  END_IF;
  x := QUERY ( pd <* bag_to_set(USEDIN(relation.
      relating_product_definition, 'CONFIG_CONTROL_DESIGN.' +
      'PRODUCT_DEFINITION_RELATIONSHIP.' + 'RELATED_PRODUCT_DEFINITION'))
      | (specific_relation IN TYPEOF (pd)) );
  REPEAT i := 1 TO HIINDEX (x) BY 1;
      IF NOT acyclic_product_definition_relationship (x[i], relatives +
          relation.relating_product_definition, specific_relation) THEN
        RETURN (FALSE) ;
      END_IF ;
```

```
END_REPEAT ;
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;
                  schema: STRING): BOOLEAN;
  LOCAL
    srr_set     : SET OF shape_representation_relationship := [ ];
    i           : INTEGER;
    j           : INTEGER;
    sdr_set     : SET OF shape_definition_representation := [ ];
    prl_set     : SET OF property_definition := [ ];
    pdrel_set   : SET OF product_definition_relationship := [ ];
    pr2_set     : SET OF property_definition := [ ];
  END_LOCAL ;
  prl_set := bag_to_set(USEDIN(assy.related_product_definition, schema +
    '.PROPERTY_DEFINITION.DEFINITION' )) ;
  REPEAT i := 1 TO HIINDEX (prl_set) BY 1;
    sdr_set := sdr_set + QUERY ( pdr <* USEDIN(prl_set[i], schema +
        '.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | ((schema +
        '.SHAPE_DEFINITION_REPRESENTATION') IN TYPEOF (pdr)) );
  END_REPEAT ;
  pdrel_set := bag_to_set(USEDIN(assy.related_product_definition, schema +
        '.PRODUCT_DEFINITION_RELATIONSHIP.' +
        'RELATED_PRODUCT_DEFINITION' )) ;
  REPEAT j := 1 TO HIINDEX (pdrel_set) BY 1;
    pr2_set := pr2_set + USEDIN (pdrel_set[j], schema +
        '.PROPERTY_DEFINITION.DEFINITION') ;
  END_REPEAT ;
  REPEAT i := 1 TO HIINDEX (pr2_set) BY 1;
    sdr_set := sdr_set + QUERY ( pdr <* USEDIN(pr2_set[i], schema +
        '.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | ((schema +
        '.SHAPE_DEFINITION_REPRESENTATION') IN TYPEOF (pdr)) );
  END_REPEAT ;
  IF SIZEOF(sdr_set) > 0 THEN
   REPEAT i := 1 TO HIINDEX(sdr_set) BY 1;
    srr_set := QUERY ( rr <* bag_to_set (USEDIN(sdr_set[i]\
        property_definition_representation.used_representation, schema +
        '.REPRESENTATION_RELATIONSHIP.REP_2')) | ((schema +
        '.SHAPE_REPRESENTATION_RELATIONSHIP') IN TYPEOF(rr)) );
    IF SIZEOF(srr_set) > 0 THEN
        REPEAT j := 1 TO HIINDEX (srr_set) BY 1;
          IF SIZEOF (QUERY ( pdr <* bag_to_set(USEDIN(srr_set[j]\
                representation_relationship.rep_1, schema +
                '.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION' ) )
                | ((schema + '.SHAPE_DEFINITION_REPRESENTATION') IN TYPEOF (
                pdr)) ) * QUERY ( pdr <* bag_to_set(USEDIN(assy.
```

```
                relating_product_definition, schema +
                '.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) | ((
                schema + '.SHAPE_DEFINITION_REPRESENTATION' ) IN
                TYPEOF (pdr)) )) >= 1 THEN
              IF SIZEOF(QUERY ( cdsr <* USEDIN(srr_set[j], schema +
                    '.CONTEXT_DEPENDENT_SHAPE_REPRESENTATION.' +
                    'REPRESENTATION_RELATION') | (NOT (cdsr\
                    context_dependent_shape_representation.
                    represented_product_relation\property_definition.
                    definition :=: assy)) )) > 0 THEN
                RETURN (FALSE) ;
              END_IF ;
            END_IF ;
          END_REPEAT ;
        END_IF ;
      END_REPEAT;
    END_IF;
    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
        i          : INTEGER;
        the_set    : SET OF GENERIC: intype := [ ];
    END_LOCAL ;
    IF SIZEOF(the_bag) > 0 THEN
        REPEAT i := 1 TO HIINDEX(the_bag) BY 1;
            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;
        d1         : direction;
        d2         : direction;
        factor     : REAL;
    END_LOCAL ;
    IF dim = 3 THEN
        d1         := NVL(normalise(axis3), dummy_gri | | direction([0,0,1]));
        d2         := first_proj_axis(d1, axis1);
        u          := [d2, second_proj_axis(d1,d2,axis2), d1];
    ELSE
     IF EXISTS (axis1) THEN
```

```
            d1 := normalise(axis1);
            u := [ d1, orthogonal_complement(d1)];
            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
           d1 := normalise(axis2);
           u := [orthogonal_complement(d1), d1];
           u[1].direction_ratios[1] := -u[1].direction_ratios[1];
           u[1].direction_ratios[2] := -u[1].direction_ratios[2];
         ELSE
           u := [dummy_gri || direction ([1, 0]), dummy_gri ||
           direction ([0, 1])];
         END_IF ;
       END_IF ;
     END_IF ;
     RETURN (u) ;
   END_FUNCTION; - - base_axis

   FUNCTION boolean_choose(b: BOOLEAN;
                    choice1, choice2: GENERIC: item): GENERIC: item;
      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
         d : direction := NVL(normalise(ref_direction), dummy_gri ||
              direction ([1,0])):
      END_LOCAL ;
      RETURN ([d, orthogonal_complement(d)]) ;
   END_FUNCTION; - - build_2axes

   FUNCTION build_axes (axis, ref_direction: direction
         ): LIST [3:3] OF direction;
      LOCAL
         d1 : direction;
         d2 : direction;
      END_LOCAL ;
      d1 := NVL(normalise(axis), dummy_gri || direction([0, 0, 1]));
      d2 := first_proj_axis(d1, ref_direction);
      RETURN ([d2, normalise (cross_product(d1, d2)) .orientation, d1]);
   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 closed_shell_reversed(a_shell: closed_shell
        ): oriented_closed_shell;
    LOCAL
the_reverse : oriented_closed_shell;
    END_LOCAL ;
    IF 'CONFIG_CONTROL_DESIGN.ORIENTED_CLOSED_SHELL' IN TYPEOF(a_shell)
            THEN
        the_reverse := dummy_tri | | connected_face_set (a_shell\
            connected_face_set.cfs_faces) | | closed_shell( ) | |
            oriented_closed_shell (a_shell\oriented_closed_shell.
            closed_shell_element, NOT a_shell\oriented_closed_shell.
            orientation) ;
    ELSE
        the_reverse := dummy_tri | | connected_face_set(a_shell\
            connected_face_set.cfs_faces) | | closed_shell( ) | |
            oriented_closed_shell (a_shell, FALSE) ;
    END_IF;
    RETURN (the_reverse) ;
END_FUNCTION; - - closed_shell_reversed

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;
        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 := dummy_gri || direction ([(v1[2] * v2[3]) - (v1[3] * v2[2]), (
            v1[3] * v2[1]) - (v1[1] * v2[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 := dummy_gri || vector(res, SQRT(mag));
        ELSE
            result := dummy_gri || vector(arg1, 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
     i          : INTEGER;
     result     : dimensional_exponents := dimensional_exponents(0, 0, 0, 0, 0, 0, 0);
    END_LOCAL ;
    IF 'CONFIG_CONTROL_DESIGN.DERIVED_UNIT' IN TYPEOF(x) THEN
        REPEAT i := LOINDEX (x.elements) TO HIINDEX(x.elements) BY 1;
                result.length_exponent := result.length_exponent +
                (x.elements [i].
                exponent * x.elements[i].unit.dimensions.length_exponent);
                result.mass_exponent := result.mass_exponent + (x.elements[i].
                exponent * x.elements[i].unit.dimensions.mass_exponent);
                result.time_exponent := result.time_exponent + (x.elements[i].
                exponent * x.elements[i].unit.dimensions.time_exponent);
                result.electric_current_exponent := result.
                electric_current_exponent + (x.elements [i].exponent * x.
                elements[i].unit.dimensions.electric_current_exponent) ;
                result.thermodynamic_temperature_exponent := result.
                thermodynamic_temperature_exponent + (x.elements[i].exponent *
                x.elements[i].unit.dimensions.
```

```
                thermodynamic_temperature_exponent) ;
                result.amount_of_substance_exponent := result.
                amount_of_substance_exponent + (x.elements[i].exponent * x.
                elements[i].unit.dimensions.amount_of_substance_exponent) ;
                result.luminous_intensity_exponent := result.
                luminous_intensity_exponent + (x.elements[i].exponent * x.
                elements[i].unit.dimensions.luminous_intensity_exponent) ;
            END_REPEAT ;
        ELSE
            result := x .dimensions ;
        END_IF ;
        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 ;
    x := using_representations(item);
    y := x[1].context_of_items;
    RETURN (y\geometric_representation_context.coordinate_space_dimension) ;
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): oriented_edge;
      LOCAL
        the_reverse : oriented_edge;
      END_LOCAL ;
      IF 'CONFIG_CONTROL_DESIGN.ORIENTED_EDGE' IN TYPEOF(an_edge) THEN
        the_reverse := dummy_tri || edge(an_edge.edge_end, an_edge.edge_start)
          || oriented_edge (an_edge\oriented_edge.edge_element, NOT an_edge\
          oriented_edge.orientation) ;
      ELSE
        the_reverse := dummy_tri || edge(an_edge.edge_end, an_edge.edge_start)
          || 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 := dummy_tri || face_bound(a_face_bound\face_bound.bound,
          NOT a_face_bound\face_bound.orientation) || face_outer_bound( );
      ELSE
        the_reverse := dummy_tri || 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): oriented_face;
      LOCAL
        the_reverse : oriented_face;
      END_LOCAL ;
      IF 'CONFIG_CONTROL_DESIGN.ORIENTED_FACE' IN TYPEOF(a_face) THEN
```

```
        the_reverse := dummy_tri || face(set_of_topology_reversed(a_face.
            bounds)) || oriented_face(a_face\oriented_face.face_element, NOT
            a_face\oriented_face.orientation) ;
    ELSE
        the_reverse := dummy_tri || face(set_of_topology_reversed(a_face.
            bounds)) || 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) THEN RETURN(?);
    ELSE
      z          := normalise(z_axis);
      IF NOT EXISTS (arg) THEN
        IF z.direction_ratios < > [1, 0, 0] THEN
          v := dummy_gri || direction ([1, 0, 0]);
        ELSE
          v := dummy_gri || direction ([0, 1, 0]);
        END_IF ;
      ELSE
        IF arg.dim < > 3 THEN RETURN(?);
        END_IF;
        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);
      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): BOOLEAN;
   IF SIZEOF (['CONFIG_CONTROL_DESIGN.BOUNDED_CURVE' ,
         'CONFIG_CONTROL_DESIGN.CONIC' ,
         'CONFIG_CONTROL_DESIGN.CURVE_REPLICA' ,
         'CONFIG_CONTROL_DESIGN.LINE',
         'CONFIG_CONTROL_DESIGN.OFFSET_CURVE_3D'] * TYPEOF(cv)) > 1 THEN
       RETURN (FALSE) ;
   ELSE
       IF SIZEOF (['CONFIG_CONTROL_DESIGN.CIRCLE' ,
           'CONFIG_CONTROL_DESIGN.ELLIPSE'] * TYPEOF(cv)) = 1 THEN
         RETURN (TRUE) ;
       ELSE
         IF (('CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE' IN TYPEOF(cv)) AND (cv
           \b_spline_curve.self_intersect = FALSE)) OR (cv\b_spline_curve.
           self_intersect = UNKNOWN) THEN
         RETURN (TRUE) ;
       ELSE
         IF (('CONFIG_CONTROL_DESIGN.COMPOSITE_CURVE' IN TYPEOF(cv)) AND
             (cv\composite_curve.self_intersect = FALSE)) OR (cv\
```

```
        composite_curve.self_intersect = UNKNOWN) THEN
    RETURN (SIZEOF (QUERY ( seg <* cv\composite_curve.segments | (
        NOT gbsf_check_curve(seg.parent_curve)) )) = 0);
ELSE
  IF 'CONFIG_CONTROL_DESIGN.CURVE_REPLICA' IN TYPEOF(cv) THEN
    RETURN(gbsf_check_curve(cv\curve_replica.parent_curve)) ;
ELSE
  IF ('CONFIG_CONTROL_DESIGN.OFFSET_CURVE_3D' IN TYPEOF(cv))
      AND ((cv\offset_curve_3d.self_intersect = FALSE) OR (cv\
      offset_curve_3d.self_intersect = UNKNOWN)) AND (NOT (
      'CONFIG_CONTROL_DESIGN.POLYLINE' IN
      TYPEOF (cv.basis_curve))) THEN
    RETURN(gbsf_check_curve(cv\offset_curve_3d.basis_curve)) ;
ELSE
  IF 'CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(cv) THEN
    RETURN (gbsf_check_curve (cv\pcurve.reference_to_curve\
        representation.items[1]) AND gbsf_check_surface(cv\
        pcurve.basis_surface)) ;
  ELSE
    IF 'CONFIG_CONTROL_DESIGN.POLYLINE' IN TYPEOF(cv) THEN
      IF SIZEOF (cv\polyline.points) >= 3 THEN
        RETURN (TRUE) ;
      END_IF;
    ELSE
      IF 'CONFIG_CONTROL_DESIGN.SURFACE_CURVE' IN TYPEOF(cv)
          THEN
        IF gbsf_check_curve(cv\surface_curve.curve_3d) THEN
        REPEAT i := 1 TO SIZEOF (cv\surface_curve.
            associated_geometry) BY 1;
          IF 'CONFIG_CONTROL_DESIGN.SURFACE' IN TYPEOF(cv\
              surface_curve.associated_geometry [i]) THEN
            IF NOT gbsf_check_surface (cv\surface_curve.
                associated_geometry[i]) THEN
              RETURN (FALSE) ;
            END_IF ;
          ELSE
            IF 'CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(cv
                \surface_curve.associated_geometry[i])
                  THEN
                IF NOT gbsf_check_curve(cv\surface_curve.
                    associated_geometry[i]) THEN
                  RETURN (FALSE) ;
                END_IF ;
            END_IF;
          END_IF ;
        END_REPEAT ;
        RETURN (TRUE) ;
      END_IF;
    ELSE
      IF 'CONFIG_CONTROL_DESIGN.TRIMMED_CURVE' IN TYPEOF(
          cv) THEN
        IF SIZEOF(['CONFIG_CONTROL_DESIGN.LINE' ,
            'CONFIG_CONTROL_DESIGN.PARABOLA' ,
            'CONFIG_CONTROL_DESIGN.HYPERBOLA'] * TYPEOF(cv\
            trimmed_curve.basis_curve)) = 1 THEN
          RETURN (TRUE) ;
        ELSE
          RETURN (gbsf_check_curve(cv\trimmed_curve.
              basis_curve)) ;
        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): BOOLEAN;
    IF 'CONFIG_CONTROL_DESIGN.CARTESIAN_POINT' IN TYPEOF(pnt) THEN
        RETURN (TRUE) ;
    ELSE
        IF 'CONFIG_CONTROL_DESIGN.POINT_ON_CURVE' IN TYPEOF(pnt) THEN
            RETURN(gbsf_check_curve(pnt\point_on_curve.basis_curve)) ;
        ELSE
          IF 'CONFIG_CONTROL_DESIGN.POINT_ON_SURFACE' IN TYPEOF(pnt) THEN
              RETURN (gbsf_check_surface(pnt\point_on_surface.basis_surface)) ;
          ELSE
              IF 'CONFIG_CONTROL_DESIGN.DEGENERATE_PCURVE' IN TYPEOF(pnt)
                    THEN
                  RETURN (gbsf_check_curve(pnt\degenerate_pcurve.
                          reference_to_curve\representation.items[1]) AND
                          gbsf_check_surface(pnt\degenerate_pcurve.basis_surface)) ;
              END_IF ;
          END_IF ;
        END_IF ;
    END_IF ;
    RETURN (FALSE) ;
END_FUNCTION; - - gbsf_check_point

FUNCTION gbsf_check_surface(sf: surface): BOOLEAN;
    IF (('CONFIG_CONTROL_DESIGN.B_SPLINE_SURFACE' IN TYPEOF(sf)) AND (sf\
        b_spline_surface.self_intersect = FALSE)) OR (sf\b_spline_surface.
        self_intersect = UNKNOWN) THEN RETURN(TRUE);
    ELSE
      IF SIZEOF (['CONFIG_CONTROL_DESIGN.SPHERICAL_SURFACE' ,
          'CONFIG_CONTROL_DESIGN.TOROIDAL_SURFACE'] * TYPEOF(sf)) = 1 THEN
      RETURN(TRUE) ;
    ELSE
      IF 'CONFIG_CONTROL_DESIGN.CURVE_BOUNDED_SURFACE' IN TYPEOF(sf)
            THEN
        IF SIZEOF (['CONFIG_CONTROL_DESIGN.CONICAL_SURFACE' ,
            'CONFIG_CONTROL_DESIGN.CYLINDRICAL_SURFACE' ,
            'CONFIG_CONTROL_DESIGN.PLANE'] * TYPEOF(sf\
            curve_bounded_surface.basis_surface)) = 1 THEN
          RETURN (SIZEOF(QUERY ( bcurve <* sf\curve_bounded_surface.
                boundaries | (NOT gbsf_check_curve(bcurve)) )) = 0);
        ELSE
          IF gbsf_check_surface (sf\curve_bounded_surface.basis_surface)
                THEN
            RETURN(SIZEOF(QUERY ( bcurve <* sf\curve_bounded_surface.
                  boundaries | (NOT gbsf_check_curve(bcurve)) )) = 0);
          END_IF ;
        END_IF ;
      ELSE
```

```
            IF (( 'CONFIG_CONTROL_DESIGN.OFFSET_SURFACE' IN TYPEOF(sf)) AND (
                  sf\offset_surface.self_intersect = FALSE)) OR (sf\
                  offset_surface.self_intersect = UNKNOWN) THEN
               RETURN(gbsf_check_surface(sf\offset_surface.basis_surface)) ;
            ELSE
               IF 'CONFIG_CONTROL_DESIGN.RECTANGULAR_COMPOSITE_SURFACE' IN
                    TYPEOF(sf) THEN
                  REPEAT i := 1 TO SIZEOF(sf\rectangular_composite_surface.
                        segments) BY 1;
                    REPEAT j := 1 TO SIZEOF(sf\rectangular_composite_surface.
                          segments[i]) BY 1;
                       IF NOT gbsf_check_surface(sf\
                          rectangular_composite_surface.segments[i] [j].
                          parent_surface) THEN RETURN(FALSE);
                       END_IF ;
                    END_REPEAT ;
                  END_REPEAT ;
                  RETURN (TRUE) ;
               ELSE
                  IF 'CONFIG_CONTROL_DESIGN.RECTANGULAR_TRIMMED_SURFACE' IN
                       TYPEOF(sf) THEN
                    IF SIZEOF(['CONFIG_CONTROL_DESIGN.CONICAL_SURFACE' ,
                      'CONFIG_CONTROL_DESIGN.CYLINDRICAL_SURFACE' ,
                      'CONFIG_CONTROL_DESIGN.PLANE'] * TYPEOF(sf\
                      rectangular_trimmed_surface.basis_surface)) = 1 THEN
                    RETURN (TRUE) ;
                  ELSE
                    RETURN (gbsf_check_surface (sf\rectangular_trimmed_surface
                          .basis_surface)) ;
                  END_IF ;
                 ELSE
                  IF 'CONFIG_CONTROL_DESIGN.SURFACE_REPLICA' IN TYPEOF(sf)
                       THEN
                    RETURN (gbsf_check_surface (sf\surface_repiica.
                      parent_surface)) ;
                  ELSE
                    IF 'CONFIG_CONTROL_DESIGN.SWEPT_SURFACE' IN TYPEOF(sf)
                         THEN
                      RETURN (gbsf_check_curve(sf\swept_surface.swept_curve)) ;
                  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.segments);
      surfs := get_basis_surface(c\composite_curve.segments[1].
          parent_curve) ;
      IF n > 1 THEN
        REPEAT i := 2 TO n BY 1;
          surfs := surfs * get_basis_surface(c\composite_curve.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 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
        res := [lis[1], n];
        REPEAT i := 2 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(1: 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 + 1 [i] ;
    END_REPEAT ;
    RETURN (s) ;
END_FUNCTION; - - list_to_set

FUNCTION make_array_of_array(lis: LIST [1:?] OF LIST [1:?] OF GENERIC:t;
                low1, u1, low2, u2: INTEGER): ARRAY OF ARRAY OF GENERIC:t;
    LOCAL
        res : ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;
    END_LOCAL ;
    IF ((u1 - low1) + 1) < > SIZEOF(lis) THEN RETURN (?);
    END_IF ;
    IF ((u2 - low2) + 1) < > SIZEOF(lis[1]) THEN RETURN(?);
    END_IF;
    res := [list_to_array(lis[1], low2, u2),(u1 - low1) + 1];
    REPEAT i := 2 TO HIINDEX(lis) BY 1;
        IF ((u2 - low2) + 1) < > SIZEOF (lis[i]) THEN RETURN (?);
        END_IF ;
        res[(low1 +i) - 1] := list_to_array(lis[i], low2, u2);
    END_REPEAT;
    RETURN (res) ;
END_FUNCTION; - - make_array_of_array

FUNCTION mixed_loop_type_set(l: SET [0:?] OF loop): LOGICAL;
    LOCAL
```

```
        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
        ): SET [1:?] OF closed_shell;
    IF SIZEOF (QUERY ( msbtype <* TYPEOF(brep) | (msbtype LIKE
            '*BREP_WITH_VOIDS') )) >= 1 THEN
        RETURN(brep\brep_with_voids.voids + brep.outer);
    ELSE
        RETURN ([brep.outer]) ;
    END_IF;
END_FUNCTION; - - msb_shells

FUNCTION msf_curve_check(cv: curve): BOOLEAN;
    IF SIZEOF (['CONFIG_CONTROL_DESIGN.BOUNDED_CURVE' ,
            'CONFIG_CONTROL_DESIGN.CONIC',
            'CONFIG_CONTROL_DESIGN.CURVE_REPLICA',
            'CONFIG_CONTROL_DESIGN.LINE' ,
            'CONFIG_CONTROL_DESIGN.OFFSET_CURVE_3D'] * TYPEOF(cv)) > 1 THEN
        RETURN(FALSE) ;
    ELSE
    IF (('CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE' IN TYPEOF(cv)) AND (cv\
            b_spline_curve.self_intersect = FALSE)) OR (cv\b_spline_curve.
            self_intersect = UNKNOWN) THEN RETURN (TRUE);
    ELSE
        IF SIZEOF(['CONFIG_CONTROL_DESIGN.CONIC' ,
                'CONFIG_CONTROL_DESIGN.LINE'] * TYPEOF(cv)) = 1 THEN
            RETURN (TRUE) ;
        ELSE
            IF 'CONFIG_CONTROL_DESIGN.CURVE_REPLICA' IN TYPEOF(cv) THEN
                RETURN(msf_curve_check(cv\curve_replica.parent_curve));
            ELSE
                IF ('CONFIG_CONTROL_DESIGN.OFFSET_CURVE_3D' IN TYPEOF(cv)) AND
                        ((cv\offset_curve_3d.self_intersect = FALSE) OR (cv\
                    offset_curve_3d.self_intersect = UNKNOWN)) AND (NOT (
                        'CONFIG_CONTROL_DESIGN.POLYLINE'
                        IN TYPEOF(cv.basis_curve))) THEN
                    RETURN(msf_curve_check(cv\offset_curve_3d.basis_curve)) ;
                ELSE
                    IF 'CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(cv) THEN
                        RETURN (msf_curve_check(cv\pcurve.reference_to_curve\
                                representation.items[1]) AND msf_surface_check(cv\
                                pcurve.basis_surface)) ;
                    ELSE
                        IF 'CONFIG_CONTROL_DESIGN.SURFACE_CURVE' IN TYPEOF(cv)
                                THEN
                            IF msf_curve_check(cv\surface_curve.curve_3d) THEN
                                REPEAT i := 1 TO SIZEOF(cv\surface_curve.
                                        associated_geometry) BY 1;
                                    IF 'CONFIG_CONTROL_DESIGN.SURFACE' IN TYPEOF(cv\
```

```
                                surface_curve.associated_geometry [i]) THEN
                              IF NOT msf_surface_check(cv\surface_curve.
                                  associated_geometry[i]) THEN
                                RETURN (FALSE) ;
                              END_IF ;
                            ELSE
                              IF 'CONFIG_CONTROL_DESIGN.PCURVE' IN TYPEOF(cv\
                                  surface_curve.associated_geometry[i]) THEN
                                IF NOT msf_curve_check(cv\surface_curve.
                                    associated_geometry [i]) THEN
                                  RETURN (FALSE) ;
                                END_IF;
                              END_IF;
                            END_IF ;
                          END_REPEAT;
                          RETURN (TRUE) ;
                        END_IF;
                      ELSE
                        IF 'CONFIG_CONTROL_DESIGN.POLYLINE' IN TYPEOF(cv) THEN
                          IF SIZEOF(cv\polyline.points) >= 3 THEN
                            RETURN (TRUE) ;
                          END_IF ;
                        END_IF ;
                      END_IF ;
                    END_IF ;
                  END_IF ;
                END_IF;
              END_IF;
            END_IF;
          END_IF ;
          RETURN (FALSE) ;
        END_FUNCTION; - - msf_curve_check


        FUNCTION msf_surface_check(surf: surface): BOOLEAN;
          IF 'CONFIG_CONTROL_DESIGN.ELEMENTARY_SURFACE' IN TYPEOF (surf) THEN
              RETURN (TRUE) ;
          ELSE
              IF 'CONFIG_CONTROL_DESIGN.SWEPT_SURFACE' IN TYPEOF (surf) THEN
                  RETURN(msf_curve_check(surf\swept_surface.swept_curve)) ;
              ELSE
                  IF (('CONFIG_CONTROL_DESIGN.OFFSET_SURFACE' IN TYPEOF (surf)) AND (
                       surf\offset_surface.self_intersect = FALSE)) OR (surf\
                       offset_surface.self_intersect = UNKNOWN) THEN
                      RETURN (msf_surface_check(surf\offset_surface.basis_surface)) ;
                  ELSE
                      IF 'CONFIG_CONTROL_DESIGN.SURFACE_REPLICA' IN TYPEOF (surf) THEN
                          RETURN (msf_surface_check(surf\surface_replica.parent_surface)) ;
                      ELSE
                          IF (('CONFIG_CONTROL_DESIGN.B_SPLINE_SURFACE' IN TYPEOF (surf))
                               AND (surf\b_spline_surface.self_intersect = FALSE)) OR (
                               surf\b_spline_surface.self_intersect = UNKNOWN) THEN
                            RETURN (TRUE) ;
                          END_IF ;
                      END_IF ;
                  END_IF ;
              END_IF ;
          END_IF ;
          RETURN (FALSE) ;
        END_FUNCTION; - - msf_surface_check
```

```
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
                v := dummy_gri || direction(arg.orientation.direction_ratios);
                IF arg.magnitude = 0 THEN RETURN (?);
                ELSE
                    vec := dummy_gri || vector(v, 1);
                END_IF ;
            END;
        ELSE
            v := dummy_gri || direction(arg.direction_ratios);
        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 open_shell_reversed(a_shell: open_shell): oriented_open_shell;
    LOCAL
        the_reverse : oriented_open_shell;
    END_LOCAL ;
    IF 'CONFIG_CONTROL_DESIGN.ORIENTED_OPEN_SHELL' IN TYPEOF(a_shell)
            THEN
        the_reverse := dummy_tri || connected_face_set(a_shell\
            connected_face_set.cfs_faces) || open_shell( ) ||
            oriented_open_shell (a_shell\oriented_open_shell,
            open_shell_element, NOT a_shell\oriented_open_shell.orientation) ;
    ELSE
        the_reverse := dummy_tri || connected_face_set(a_shell\
            connected_face_set.cfs_faces) || open_shell( ) ||
            oriented_open_shell (a_shell, FALSE) ;
    END_IF;
    RETURN (the_reverse) ;
END_FUNCTION; - - open_shell_reversed
```

```
FUNCTION orthogonal_complement(vec: direction): direction;
    LOCAL
        result : direction;
    END_LOCAL ;
    IF (vec.dim < > 2) OR (NOT EXISTS(vec)) THEN RETURN (?);
    ELSE
        result   := dummy_gri || direction ([-vec.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  :  BOOLEAN : = 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): oriented_path;
    LOCAL
        the_reverse : oriented_path;
    END_LOCAL ;
    IF 'CONFIG_CONTROL_DESIGN.ORIENTED_PATH' IN TYPEOF(a_path) THEN
        the_reverse :  = dummy_tri || path(list_of_topology_reversed(a_path.
            edge_list)) || oriented_path(a_path\oriented_path.path_element,
            NOT a_path\oriented_path.orientation) ;
    ELSE
        the_reverse := dummy_tri || path(list_of_topology_reversed(a_path.
            edge_list)) || oriented_path(a_path, FALSE);
    END_IF ;
    RETURN (the_reverse) ;
END_FUNCTION; - - path_reversed

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 RETURN(?);
    ELSE
        IF 'CONFIG_CONTROL_DESIGN.VECTOR' IN TYPEOF (vec) THEN
            v := dummy_gri || direction(vec.orientation.direction_ratios);
            mag := scalar * vec.magnitude;
        ELSE
            v := dummy_gri || direction(vec.direction_ratios);
            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 := dummy_gri || vector(normalise(v), 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 : = dummy_gri || 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;
    IF 'CONFIG_CONTROL_DESIGN.OPEN_SHELL' IN TYPEOF(a_shell) THEN
        RETURN(open_shell_reversed(a_shell)) ;
    ELSE
        IF 'CONFIG_CONTROL_DESIGN.CLOSED_SHELL' IN TYPEOF(a_shell) THEN
            RETURN (closed_shell_reversed(a_shell)) ;
        ELSE
            RETURN (?) ;
        END_IF;
    END_IF;
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 ;
    REPEAT i := 1 TO SIZEOF(ords.requests) BY 1;
        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 ;
    REPEAT k := 1 TO SIZEOF(assign) BY 1;
        versions := versions + assign[k].items;
    END_REPEAT ;
    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_items (item: founded_item_select;
                      checked_items: SET OF founded_item_select
        ): SET OF founded_item_select;
    LOCAL
        next_items      : SET OF founded_item_select;
        new_check_items : SET OF founded_item_select;
        result_items    : SET OF founded_item_select;
    END_LOCAL ;
```

```
    result_items := [ ];
    new_check_items := checked_items + item;
    next_items := QUERY ( z <* bag_to_set(USEDIN(item,' ')) | ((
          'CONFIG_CONTROL_DESIGN.REPRESENTATION_ITEM' IN TYPEOF(z)) OR (
          'CONFIG_CONTROL_DESIGN.FOUNDED_ITEM' IN TYPEOF(z))) );
    IF SIZEOF(next_items) > 0 THEN
       REPEAT i := 1 TO HIINDEX(next_items) BY 1;
          IF NOT (next_items[i] IN new_check_items) THEN
            result_items := result_items + next_items[i] + using_items(
                  next_items[i], new_check_items) ;
          END_IF ;
       END_REPEAT ;
    END_IF;
    RETURN (result_items) ;
END_FUNCTION; - - using_items

FUNCTION using_representations (item: founded_item_select
        ) : SET OF representation;
    LOCAL
        results                 : SET OF representation;
        intermediate_items      : SET OF founded_item_select;
        result_bag              : BAG OF representation;
    END_LOCAL ;
    results := [ ];
    result_bag := USEDIN (item,
    'CONFIG_CONTROL_DESIGN.REPRESENTATION.ITEMS') ;
    IF SIZEOF (result_bag) > 0 THEN
        REPEAT i := 1 TO HIINDEX (result_bag) BY 1;
       results := results + result_bag[i];
        END_REPEAT ;
    END_IF ;
    intermediate_items := using_items( item, [ ]);
    IF SIZEOF (intermediate_items) > 0 THEN
        REPEAT i := 1 TO HIINDEX (intermediate_items) BY 1;
          result_bag := USEDIN (intermediate_items [i],
                'CONFIG_CONTROL_DESIGN.REPRESENTATION.ITEMS') ;
          IF SIZEOF (result_bag) > 0 THEN
             REPEAT j := 1 TO HIINDEX (result_bag) BY 1;
                results := results + result_bag [j];
             END_REPEAT ;
          END_IF;
        END_REPEAT ;
    END_IF ;
    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;
    OTHERWISE  :  RETURN(TRUE) ;
    END_CASE ;
END_FUNCTION; - - valid_calendar_date

FUNCTION valid_geometrically_bounded_wf_curve(crv: curve): BOOLEAN;
    IF SIZEOF (['CONFIG_CONTROL_DESIGN.POLYLINE',
        'CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE' ,
        'CONFIG_CONTROL_DESIGN.ELLIPSE', 'CONFIG_CONTROL_DESIGN.CIRCLE'] *
        TYPEOF(crv)) = 1 THEN RETURN(TRUE);
    ELSE
      IF 'CONFIG_CONTROL_DESIGN.TRIMMED_CURVE' IN TYPEOF(crv) THEN
        IF SIZEOF (['CONFIG_CONTROL_DESIGN.LINE' ,
            'CONFIG_CONTROL_DESIGN.PARABOLA' ,
            'CONFIG_CONTROL_DESIGN.HYPERBOLA'] * TYPEOF (crv\trimmed_curve.
            basis_curve)) = 1 THEN RETURN(TRUE);
        ELSE
          RETURN (valid_geometrically_bounded_wf_curve(crv\trimmed_curve.
              basis_curve)) ;
        END_IF;
      ELSE
        IF 'CONFIG_CONTROL_DESIGN.OFFSET_CURVE_3D' IN TYPEOF(crv) THEN
          RETURN (valid_geometrically_bounded_wf_curve(crv\offset_curve_3d.
              basis_curve)) ;
        ELSE
          IF 'CONFIG_CONTROL_DESIGN.CURVE_REPLICA' IN TYPEOF(crv) THEN
            RETURN (valid_geometrically_bounded_wf_curve(crv\curve_replica.
                parent_curve)) ;
          ELSE
            IF 'CONFIG_CONTROL_DESIGN.COMPOSITE_CURVE' IN TYPEOF(crv)
                THEN
              RETURN (SIZEOF(QUERY ( ccs <* crv\composite_curve.segments |
                (NOT valid_geometrically_bounded_wf_curve(ccs.
                parent_curve)) )) = 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): BOOLEAN;
    IF 'CONFIG_CONTROL_DESIGN.CARTESIAN_POINT' IN TYPEOF(pnt) THEN
      RETURN (TRUE) ;
    ELSE
      IF 'CONFIG_CONTROL_DESIGN.POINT_ON_CURVE' IN TYPEOF(pnt) THEN
        RETURN (valid_geometrically_bounded_wf_curve(pnt\point_on_curve.
            basis_curve)) ;
      ELSE
        IF 'CONFIG_CONTROL_DESIGN.POINT_REPLICA' IN TYPEOF(pnt) THEN
          RETURN (valid_geometrically_bounded_wf_point(pnt\point_replica.
              parent_pt)) ;
```

```
        END_IF ;
      END_IF ;
  END_IF ;
  RETURN (FALSE) ;
END_FUNCTION; - - valid_geometrically_bounded_wf_point

FUNCTION valid_measure_value(m: measure_value): BOOLEAN;
  IF 'REAL' IN TYPEOF(m) THEN
      RETURN(m > 0);
  ELSE
      IF 'INTEGER' IN TYPEOF(m) THEN RETURN (m > 0);
      ELSE
          RETURN (TRUE) ;
      END_IF;
  END_IF ;
END_FUNCTION; - - valid_measure_value

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): BOOLEAN;
    IF SIZEOF(['CONFIG_CONTROL_DESIGN.LINE', 'CONFIG_CONTROL_DESIGN.CONIC',
            'CONFIG_CONTROL_DESIGN.B_SPLINE_CURVE',
            'CONFIG_CONTROL_DESIGN.POLYLINE'] * TYPEOF(crv)) = 1 THEN
        RETURN (TRUE) ;
    ELSE
```

```
        IF 'CONFIG_CONTROL_DESIGN.CURVE_REPLICA' IN TYPEOF(crv) THEN
            RETURN (valid_wireframe_edge_curve(crv\curve_replica.parent_curve));
        ELSE
            IF 'CONFIG_CONTROL_DESIGN.OFFSET_CURVE_3D' IN TYPEOF(crv) THEN
                RETURN (valid_wireframe_edge_curve(crv\offset_curve_3d.
                    basis_curve));
            END_IF ;
        END_IF ;
    END_IF ;
    RETURN (FALSE) ;
END_FUNCTION; - - valid_wireframe_edge_curve

FUNCTION valid_wireframe_vertex_point(pnt: point): BOOLEAN;
    IF 'CONFIG_CONTROL_DESIGN.CARTESIAN_POINT' IN TYPEOF (pnt) THEN
        RETURN (TRUE) ;
    ELSE
        IF 'CONFIG_CONTROL_DESIGN.POINT_REPLICA' IN TYPEOF(pnt) THEN
            RETURN (valid_wireframe_vertex_point(pnt\point_replica.parent_pt));
        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 RETURN (?) ;
    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 ;
                res     := dummy_gri || direction(vec1.direction_ratios);
            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 := dummy_gri || vector(res, SQRT(mag));
            ELSE
              result := dummy_gri || vector(vec1, 0);
            END_IF;
          END;
        END_IF;
        RETURN (result) ;
    END_FUNCTION; - - vector_difference

END_SCHEMA; - - config_control_design
(*
```

ПРИЛОЖЕНИЕ В
(обязательное)

Сокращенные наименования объектов прикладной интерпретированной модели (ПИМ)

В настоящем приложении представлены индивидуальные сокращения наименований каждого объекта, рассмотренного в настоящем стандарте. Полные наименования объектов взяты из EXPRESS-спецификации листинга ПИМ, приведенного в приложении А.

В таблице В.1 приведены сокращенные наименования объектов (сокращения), представленных в ПИМ из настоящего стандарта. Требования по использованию сокращенных наименований этих объектов установлены в методах реализации, описанных в соответствующих стандартах серии ГОСТ Р ИСО 10303.

Содержание настоящего приложения, представленное в электронном виде, может быть получено по электронной почте (см. приложение J).

Таблица В.1 — Сокращенные наименования объектов ПИМ

Полное наименование объекта	Сокращенное наименование
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
AREA_UNIT	ARUNT
ASSEMBLY_COMPONENT_USAGE	ASCMUS
ASSEMBLY_COMPONENT_USAGE_SUBSTITUTE	ACUS
AXIS1_PLACEMENT	AX1PLC
AXIS2_PLACEMENT_2D	A2PL2D

Продолжение таблицы В.1

Полное наименование объекта	Сокращенное наименование
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_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
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

Продолжение таблицы В.1

Полное наименование объекта	Сокращенное наименование
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
DEGENERATE_TOROIDAL_SURFACE	DGTRSR
DESIGN_CONTEXT	DSGCNT
DESIGN_MAKE_FROM_RELATIONSHIP	DMFR
DIMENSIONAL_EXPONENTS	DMNEXP
DIRECTED_ACTION	DRCACT
DIRECTION	DRCTN
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

Продолжение таблицы В.1

Полное наименование объекта	Сокращенное наименование
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
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_3D	OFCR3D
OFFSET_SURFACE	OFFSRF
OPEN_SHELL	OPNSHL
ORDINAL_DATE	ORDDT
ORGANIZATION	ORGNZT
ORGANIZATIONAL_ADDRESS	ORGADD
ORGANIZATIONAL_PROJECT	ORGPRJ
ORGANIZATION_RELATIONSHIP	ORGRLT

Продолжение таблицы В.1

Полное наименование объекта	Сокращенное наименование
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
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

Продолжение таблицы В.1

Полное наименование объекта	Сокращенное наименование
PRODUCT_RELATED_PRODUCT_CATEGORY	PRPC
PROMISSORY_USAGE_OCCURRENCE	PRUSOC
PROPERTY_DEFINITION	PRPDFN
PROPERTY_DEFINITION_REPRESENTATION	PRDFRP
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
START_WORK	STRWRK
SUPPLIED_PART_RELATIONSHIP	SPPRRL

Окончание таблицы В.1

Полное наименование объекта	Сокращенное наименование
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

ПРИЛОЖЕНИЕ С
(обязательное)

Форма заявки о соответствии реализации протоколу

Общие положения:

Форма заявки о соответствии реализации протоколу (ЗСРП) обеспечивает проведение оценки соответствия реализаций настоящему стандарту. Данная форма ЗСРП включает ряд вопросов, охватывающих статическую информацию о тестируемой реализации (ТР). Эта информация используется при статистической оценке выбранных вариантов (опций) и конфигурировании сеанса соответствующих аттестационных тестов (динамической оценке).

Ряд вариантов (опций) определен в настоящем стандарте с целью использования их в соответствующих реализациях. Некоторые из этих опций могут быть использованы (или не использованы) при динамическом выборе (для прогона), например, в качестве атрибутов OPTIONAL соответствующего объекта. Другие опции могут быть выбраны статически (в пределах времени формирования соответствующей конфигурации), например конкретный стиль геометрии, заданный в классе соответствия.

Вопросы

Необходимо упростить ссылку на идентификатор (обозначение) реализации изделия или системы, тестируемых на соответствие стандартам серии ГОСТ Р ИСО 10303.

1 Обозначение (или наименование) изделия/системы _____

Конкретная реализация должна, по крайней мере, обеспечивать функциональные возможности класса соответствия 1a, определенного в настоящем стандарте. В стандарте определены 12 классов соответствия. Каждый из этих классов определяет подмножество конструктивов ПИМ, описанной в настоящем стандарте. Данные классы описаны в разделе 6 настоящего стандарта.

2 Требуемые классы соответствия (функциональные возможности) — область выбора:

1a — обозначение изделия без формы;
1b — информация о проекте с управляемой конфигурацией без формы;
2a — обозначение изделия & поверхность & каркас & без топологии;
2b — информация о проекте с управляемой конфигурацией & поверхность & каркас & без топологии;
3a — обозначение изделия & каркас & топология;
3b — информация о проекте с управляемой конфигурацией & каркас & топология;
4a — информация о проекте с управляемой конфигурацией & множественные поверхности & топология;
4b — информация о проекте с управляемой конфигурацией & множественные поверхности & топология;
5a — обозначение изделия & фасеточные BREP;
5b — информация о проекте с управляемой конфигурацией & фасеточные BREP;
6a — обозначение изделия & усовершенствованные BREP;
6b — информация о проекте с управляемой конфигурацией & усовершенствованные BREP.

Соответствие настоящему стандарту может быть выполнено посредством одного или нескольких различных методов реализации. Конкретные методы реализации определяют типы выполнения обмена данными в соответствии с настоящим стандартом.

3 Требуемые виды (формы) реализации — структура обмена (по ГОСТ Р 10303-21).

Если эта реализация получает данные, не удовлетворяющие требованиям настоящего стандарта в части выбора класса(ов) соответствия или стандартов серии ГОСТ Р ИСО 10303 группы 20 (по методам реализации), должен быть выдан отрицательный ответ. Ответ должен содержать набор соответствующих сведений.

4 Ответ по умолчанию _____

Соответствующая реализация должна обеспечивать опции, выбранные посредством последующей динамической оценки (тестирования) без внесения изменений. В среде пользователя данная реализация должна постоянно обеспечивать выполнение выбранных опций, авторизованный контроль пользователем изменений и определений выбранных опций или того и другого (в зависимости от специфики данной опции).

5 Позволяет ли ТР обеспечить пользователю свободу при изменении и определении выбранных опций? Да или нет.

6 Если да, то в какой степени?

a) Класс(ы) соответствия: _____

b) Ответ по умолчанию: _____

Формулировка соответствия должна включать обозначение по крайней мере одной стороны, заявляющей о соответствии данной реализации.

7 Эксперт(ы) (испытатель/сертификатор/аккредитатор) _____

ПРИЛОЖЕНИЕ D
(обязательное)

Специальные требования к методам реализации

Соответствие настоящему стандарту должно быть реализовано посредством одного или нескольких методов реализации. Конкретные методы реализации определяют, какой тип обмена предпочтителен в данном стандарте. В настоящем стандарте определен один метод реализации — по ГОСТ Р ИСО 10303-21.

В структуре обмена файловый формат этой структуры должен быть закодирован в соответствии с синтаксисом и отображением языка EXPRESS, определенного в ГОСТ Р ИСО 10303-21, и ПИМ, описанной в приложении А к настоящему стандарту. Заголовок структуры обмена должен быть задан посредством определенной в настоящем стандарте схемы "config_control_design".

ПРИЛОЖЕНИЕ Е
(обязательное)

Регистрация информационного объекта

Е.1 Обозначение документа

Для обеспечения однозначного обозначения информационного объекта в открытой системе настоящему стандарту присвоен следующий идентификатор объекта:

{ iso standard 10303 part(203) version(3) }

Смысл данного обозначения установлен в ГОСТ Р ИСО/МЭК 8824-1 и описан в ГОСТ Р ИСО 10303-1.

Е.2 Обозначение схемы

В ГОСТ Р ИСО 10303-1 описано использование ГОСТ Р ИСО/МЭК 8824-1 для идентификации конкретных схем. Настоящий стандарт содержит две схемы, при этом каждой из них присвоен собственный идентификатор для обеспечения однозначного обозначения схемы в открытой системе.

Е.2.1 Обозначение полной схемы config_control_design

Полной схеме **config_control_design_schema** (см. приложение А) присвоен следующий идентификатор объекта:

{ iso standard 10303 part(203) version(2) object(1) config-control-design-schema (1) }

Е.2.2 Обозначение сокращенной схемы config_control_design

Сокращенной схеме **config_control_design** (см. 5.2) присвоен следующий идентификатор объекта:

{ iso standard 10303 part(203) version(2) object(1) config-control-design-schema (2) }

ПРИЛОЖЕНИЕ F
(справочное)

Прикладная функциональная модель

Прикладная функциональная модель (ПФМ) предназначена для облегчения понимания области действия и информационных требований, определяемых конкретным прикладным протоколом. Данная модель представлена в виде набора определений видов деятельности и данных и набора диаграмм видов деятельности. Данная модель охватывает виды деятельности вне рамок данного прикладного протокола. Определения, содержащиеся в настоящем приложении, не заменяют приведенных в основной части настоящего стандарта. В диаграммах использована нотация IDEF0 (см. приложение L — [3], [4]). На рисунке F.1 приведено краткое пояснение по интерпретации стрелок, присоединенных к каждому функциональному блоку, описывающему вид деятельности. Каждый вид деятельности может быть декомпозирован с целью более детального рассмотрения. Если отдельный блок был декомпозирован, то он иллюстрируется отдельным рисунком.

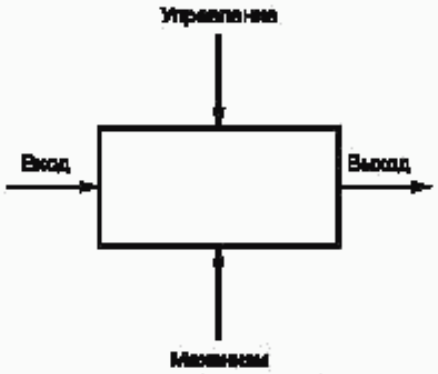


Рисунок F.1 — Основная нотация IDEF0

Как и любая IDEF0 модель, ПФМ зависит от применяемой точки зрения и поставленной цели. ПФМ построена с точки зрения проектировщика. Целью ПФМ является пояснение контекста и области действия конкретного прикладного протокола.

F.1 Терминология прикладной функциональной модели (ПФМ)

Ниже перечислены термины, используемые в диаграммах описания прикладной деятельности. Термины, отмеченные знаком «*», не входят в область действия настоящего прикладного протокола.

F.1.1 **административные данные** (administrative data): Данные, относящиеся к конструкции изделия, полученные при управлении проектом (конструированием). Примерами данных этого типа являются наименование компании, код поставщика, гриф секретности и обозначение (номер) контракта.

F.1.2 **данные анализа и тестирования** (analysis and test data)*: Данные, созданные в процессе проектирования (конструирования), используемые для оценки конструкции (проекта).

F.1.3 **модели анализа и тестирования** (analysis and test models)*: Представления данных о конструкции изделия, используемые для оценки конструкции.

F.1.4 **анализ и тестирование конструкции** (analysis and test design)*: Процесс оценки соответствия конструкции изделия установленным требованиям.

F.1.5 **схема сборочной единицы** (assembly layout): Представление того, как отдельные детали (компоненты) соединяются для образования сборочных единиц.

F.1.6 **требования к сборочной единице** (assembly requirements): Ограничения, наложенные на конструкцию изделия, относящиеся к способу соединения компонентов сборочной единицы.

F.1.7 **распоряжение по заявке (предложению) об изменении** (change request disposition): Документ, содержащий официальный технический ответ на заявку об изменении конструкции.

F.1.8 **концептуальная схема конструкции** (conceptual design layout): Представление того, как конструкторские идеи могут быть воплощены в изделии.

F.1.9 **схематизация конструкции изделия** (conceptualize product design)*: Процесс, при выполнении которого конструкторские идеи, требования и ограничения преобразуются в формализованное описание конструкции. Данное описание состоит из концептуальной схемы конструкции и требований к сборке.

F.1.10 **данные о проекте изделия с управляемой конфигурацией** (configuration controlled product design data): Данные о конструкции изделия, записанные в официальной системе управления конфигурацией.

F.1.11 **обратная связь при управлении конфигурацией** (configuration control feedback)*: Отклонения от стандартов, установленной практики и процедур, обнаруженные в процессе утверждения конструкции.

F.1.12 **требования контракта** (contractual requirements)*: Элементы конструкции изделия, установленные в юридическом документе.

F.1.13 **данные по управлению конструированием изделия** (control product design data)*: Формальные идентификаторы изделия, например, номер и версия детали, обеспечивающие соответствие установленным требованиям, стандартам, принятой практике и процедурам.

F.1.14 **определение и формулировка конструкции изделия** (define and formulate product design): Процесс синтеза конструкции изделия. Данный процесс, охватывающий эскизное (концептуальное), техническое (предварительное) и рабочее (детальное) проектирование, сопровождается теоретическим анализом и моделированием конструкции, а также оценкой альтернативных вариантов конструкции.

F.1.15 **заявка об изменении конструкции** (design change request): Официальная заявка об изменении любой формы, соединения или функции существующей конструкции.

F.1.16 **концепция конструкции** (design concept)*: Формализованное выражение конструкторской идеи изделия.

F.1.17 **конструкторские ограничения** (design constraint)*: Ограничения, связанные с конструированием, например технологические, а также по отношению цена/свойства.

F.1.18 **обратная связь конструкции (альтернативные конструкции)** [design feedback (design alternatives)])*: Анализ предшествующих или альтернативных конструкций, используемый для формирования новой конструкции.

F.1.19 **конструкторские идеи** (design ideas)*: Схематизация формы, монтажа и функциональных требований для предложенной конструкции изделия.

F.1.20 **структура интеграции конструкции** (design integration structure): Представление формирования функциональных систем посредством соединения ряда сборочных единиц и сборок.

F.1.21 **разработка конструкции изделия** (develop product design): Вид деятельности, при реализации которого создают, анализируют, тестируют и выпускают представление формы, монтажа и функций изделия.

F.1.22 **заявка на распространение** (distribution request)*: Официальный документ, определяющий количество копий конструкции изделия, направляемых определенным лицам в организации.

F.1.23 **информация о рассылке документации** (distribution tracking information): Данные, определяющие организации или лица, получающие отдельные конструкторские документы.

F.1.24 **изменения чертежей** (drawing corrections)*: Изменения чертежей, основанные на стандартной практике выпуска чертежей в конкретной организации.

F.1.25 **авторизация технического изменения** (engineering change authorization): Технический документ, определяющий выпущенную или измененную конструкторскую документацию. Данный документ определяет, что было сделано в процессе конструирования и утверждения изменений конструкторской документации.

F.1.26 **информация о существующих изделиях** (existing product information): Официальная конструкторская документация на изделие.

F.1.27 **требования к проверке, процессам и материалам** (inspection, process, material requirements)*: Требования к проверке, реализации процессов и характеристикам материалов, необходимые для создания формы, монтажа и функций изделия.

F.1.28 **требования к жизненному циклу** (life cycle requirements)*: Информация о требованиях к конструкции, обеспечивающая возможности ее сопровождения, технологичности и удобства обслуживания.

F.1.29 **управление конструированием** (manage design)*: Деятельность по разработке в процессе конструирования графиков, финансовых балансов и потребностей в персонале.

F.1.30 **управляющие директивы, график, бюджет** (management directives, schedule, budget)*: Результат деятельности по планированию управления конструкторской подготовкой изделия.

F.1.31 **идентификаторы конфигурации изделия** (product configuration identifiers): Обозначения (идентификаторы) утвержденных конструкторских документов, определяющие использование этих документов в структуре изделия, например обозначение детали или используемого элемента конфигурации.

F.1.32 **данные о конструкции изделия** (product design data): Документация, создаваемая в процессе конструирования, определяющая форму, монтаж и функции изделия.

F.1.33 **требования к изделию** (product requirements)*: Критерии, определяющие функциональные возможности и характеристики изделия, в т.ч. его назначение, стоимость, функции и конкурентоспособность.

F.1.34 **структура изделия** (product structure): Связь между элементами, образующими изделие.

F.1.35 **проектные директивы** (project directives)*: Документы, описывающие политику проекта, в т.ч. отклонения от стандартных методов, указания заказчику и требования по специальным вопросам.

F.1.36 **выпущенные технические данные о конструкции изделия** (realeased engineering product design data): Утвержденная соответствующими ответственными лицами конструкторская документация.

F.1.37 **статус утверждения** (release status): Информация, содержащая сведения о том, какие данные, когда и кем были выданы (распространены).

F.1.38 **отчеты об исследованиях** (research papers)*: Любая документация, описывающая процессы или изменения, внесенные при разработке конструкции изделия, например исследование производственной (торговой) деятельности, сделанные выводы, описание патентов.

F.1.39 **спецификации, стандарты, практика и процедуры** (specifications, standards, practices and procedures): Официальная документация, влияющая на процессы конструирования с точки зрения унификации изделий.

F.1.40 **информация о стандартных деталях** (standart part information): Данные о конструкции изделия от организаций по стандартизации.

F.1.41 **стандартные инструменты, оборудование** (standard tools, hardware)*: Существующие приспособления и механизмы (инструментарий), учитываемые при конструировании изделия.

F.1.42 **синтез и отработка конфигурации** (synthesize and refine configuration): Процесс проверки соответствия формы, монтажа и функции части изделия по отношению к изделию в целом.

F.1.43 **требования к инструментарию** (tools requirements)*: Инструментальные средства, необходимые для испытаний, обработки и изготовления сконструированного изделия.

F.2 Диаграммы прикладной функциональной модели (ПФМ)

На нижеприведенных диаграммах (рисунки F.2 и F.3) процессы и потоки данных, отмеченные знаком «*», не относятся к рассматриваемой области.

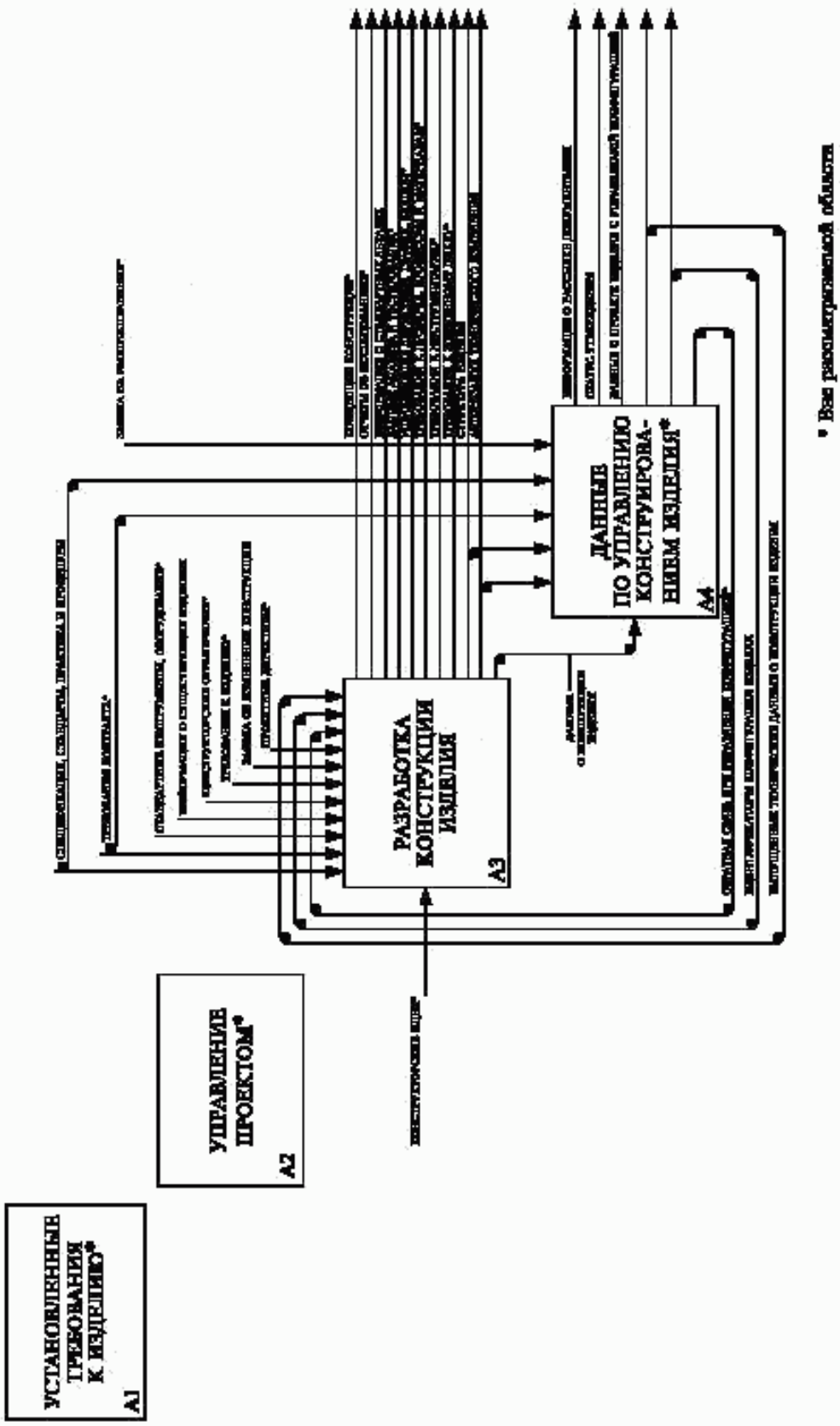


Рисунок F.2 — IDEF0-диаграмма A0. Управление проектированием изделия

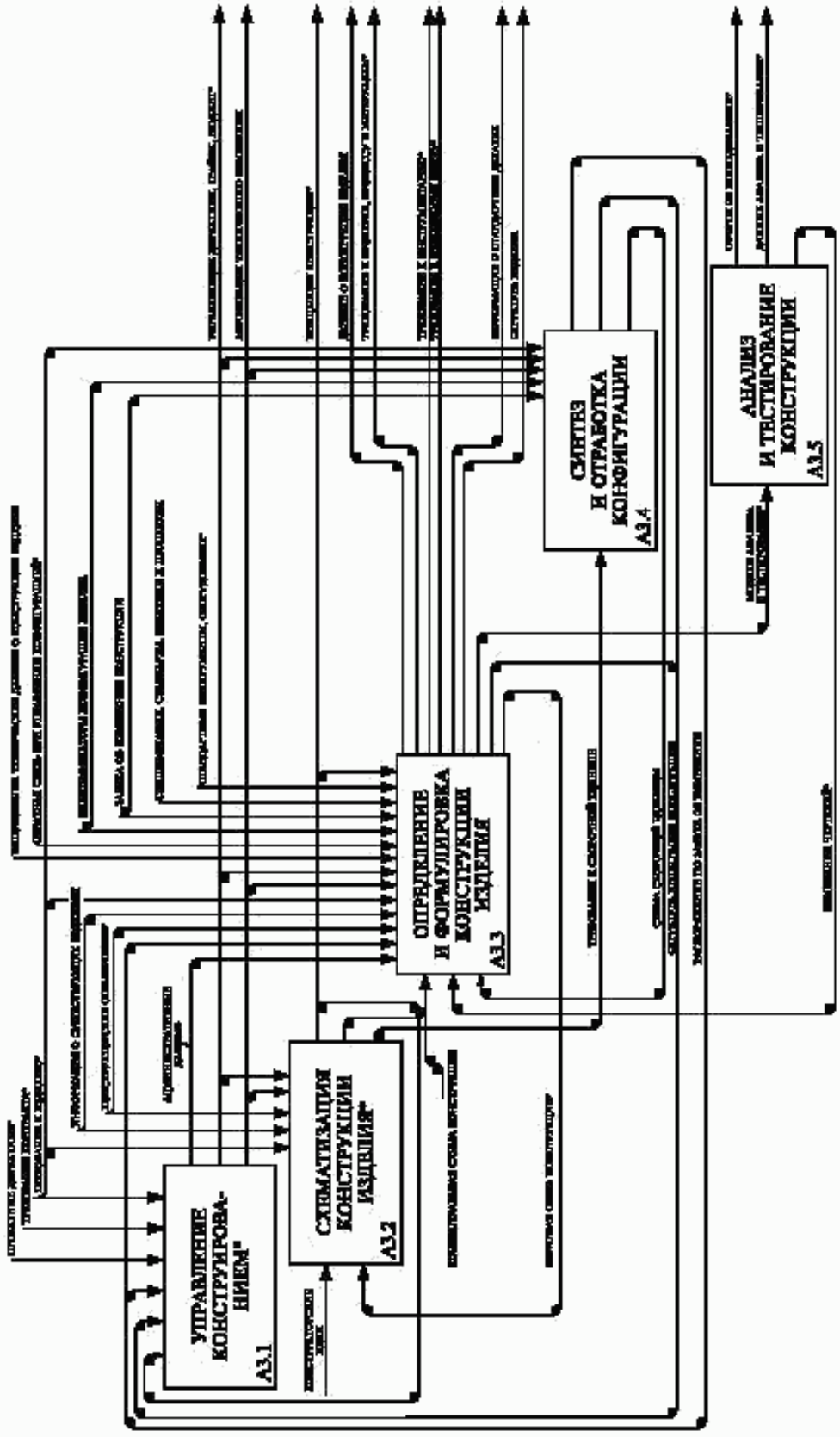


Рисунок F.3 — IDEF0-диаграмма A3. Разработка конструкции изделия

ПРИЛОЖЕНИЕ G
(справочное)

Прикладная эталонная модель

Настоящее приложение описывает прикладную эталонную модель (ПЭМ) прикладного протокола для проектов с управляемой конфигурацией. ПЭМ определяет графическое представление структуры и ограничений прикладных объектов, описанных в разделе 4 настоящего стандарта (см. рисунки G.1—G.7) в нотации IDEF1X (см. приложение L [5]). ПЭМ является инвариантной по отношению к любым методам реализации.

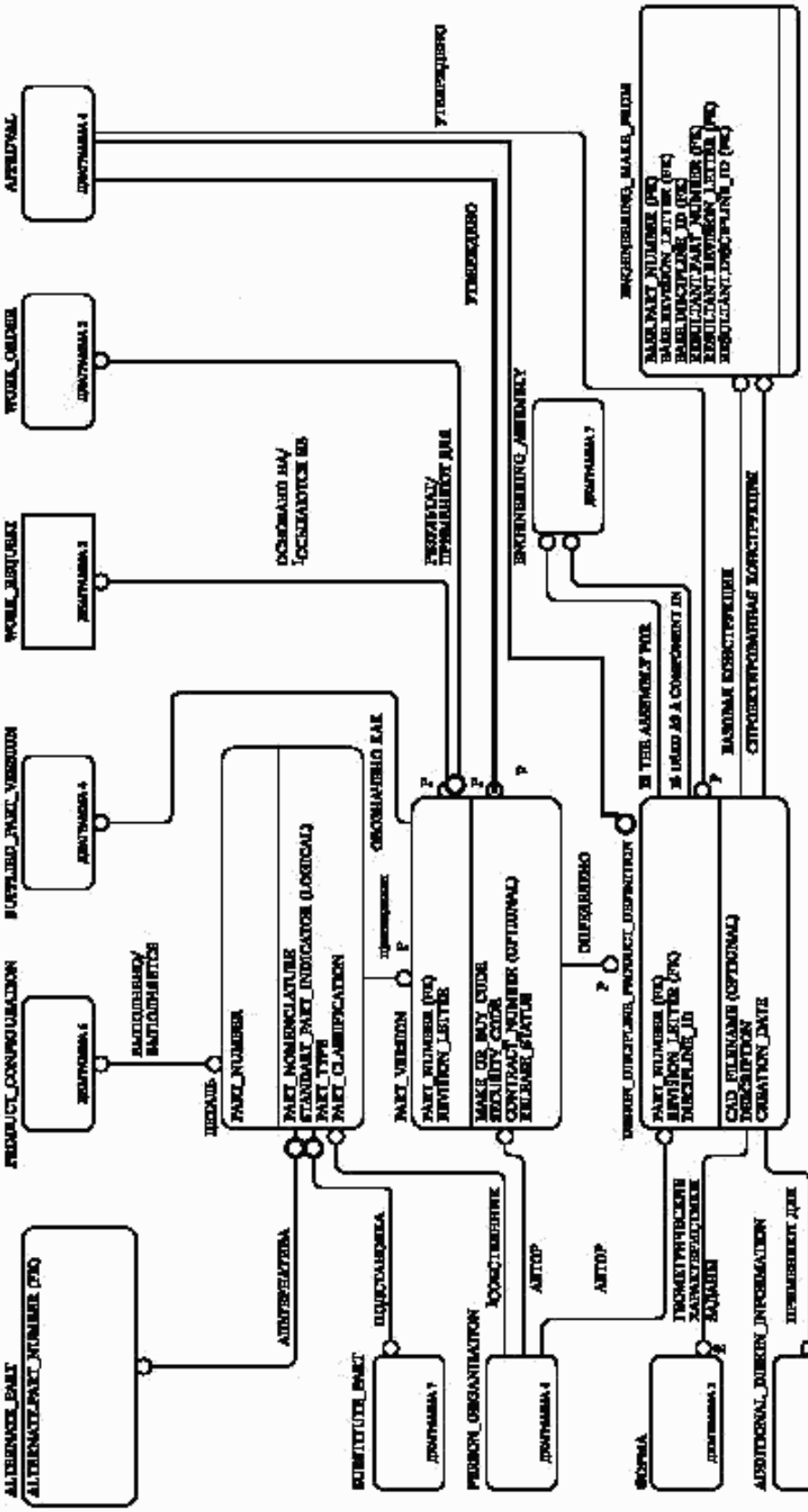


Рисунок G.1 — IDEF1X-диаграмма ПЭМ 1 из 7

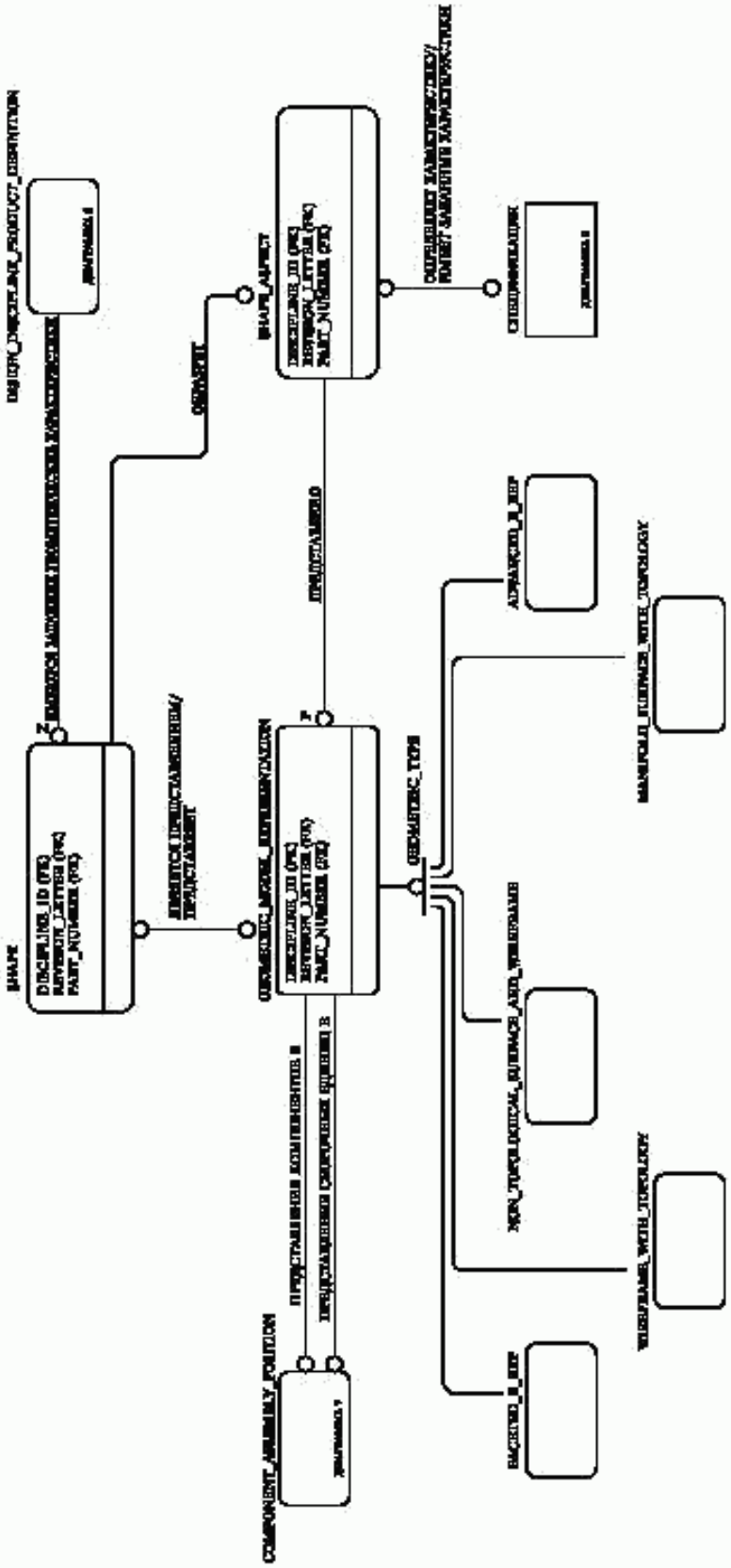


Рисунок G.2 — IDEF1X-диаграмма ПЭМ 2 из 7

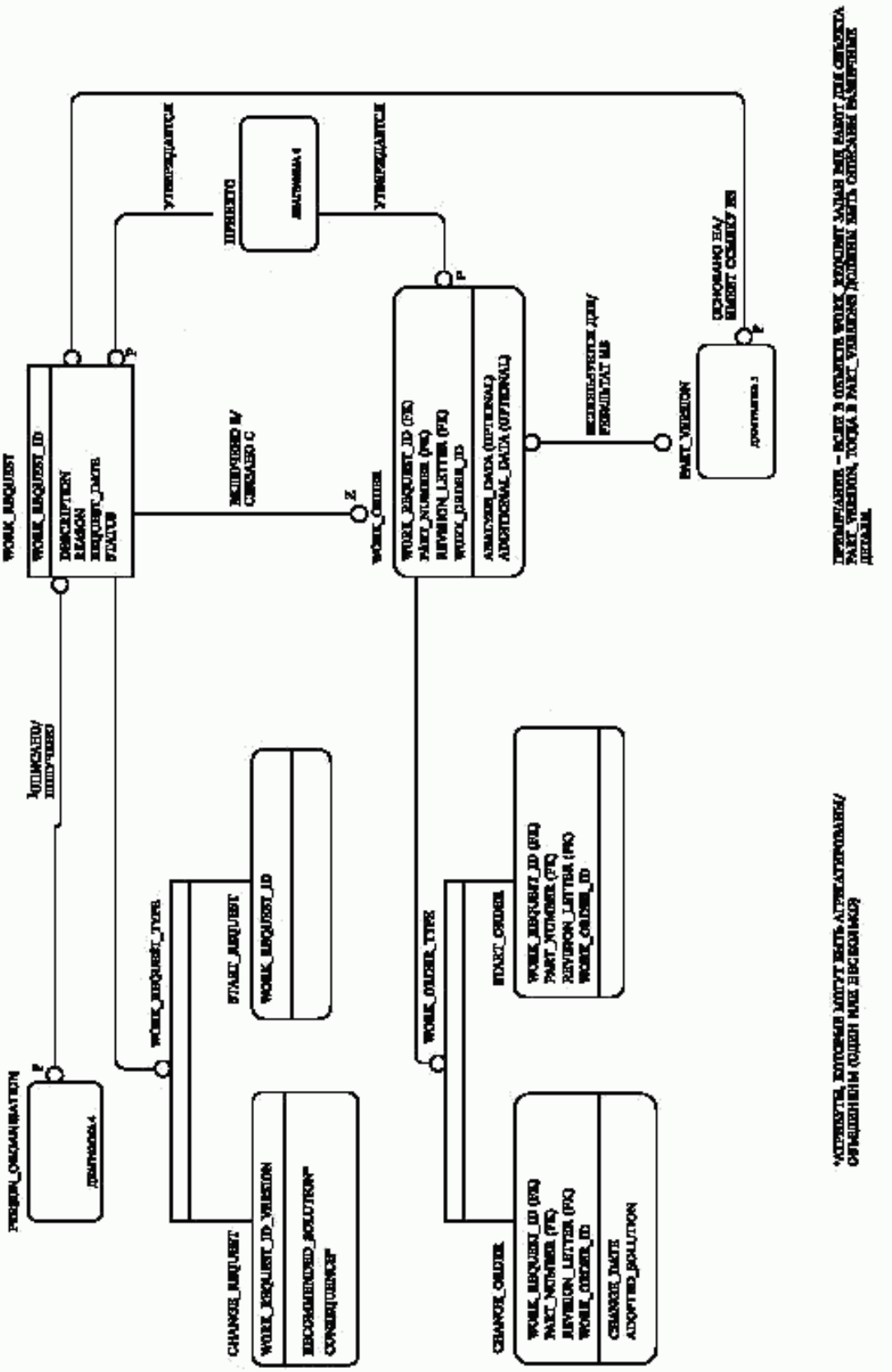


Рисунок G.3 — IDEF1X-диаграмма ПЭМ 3 из 7

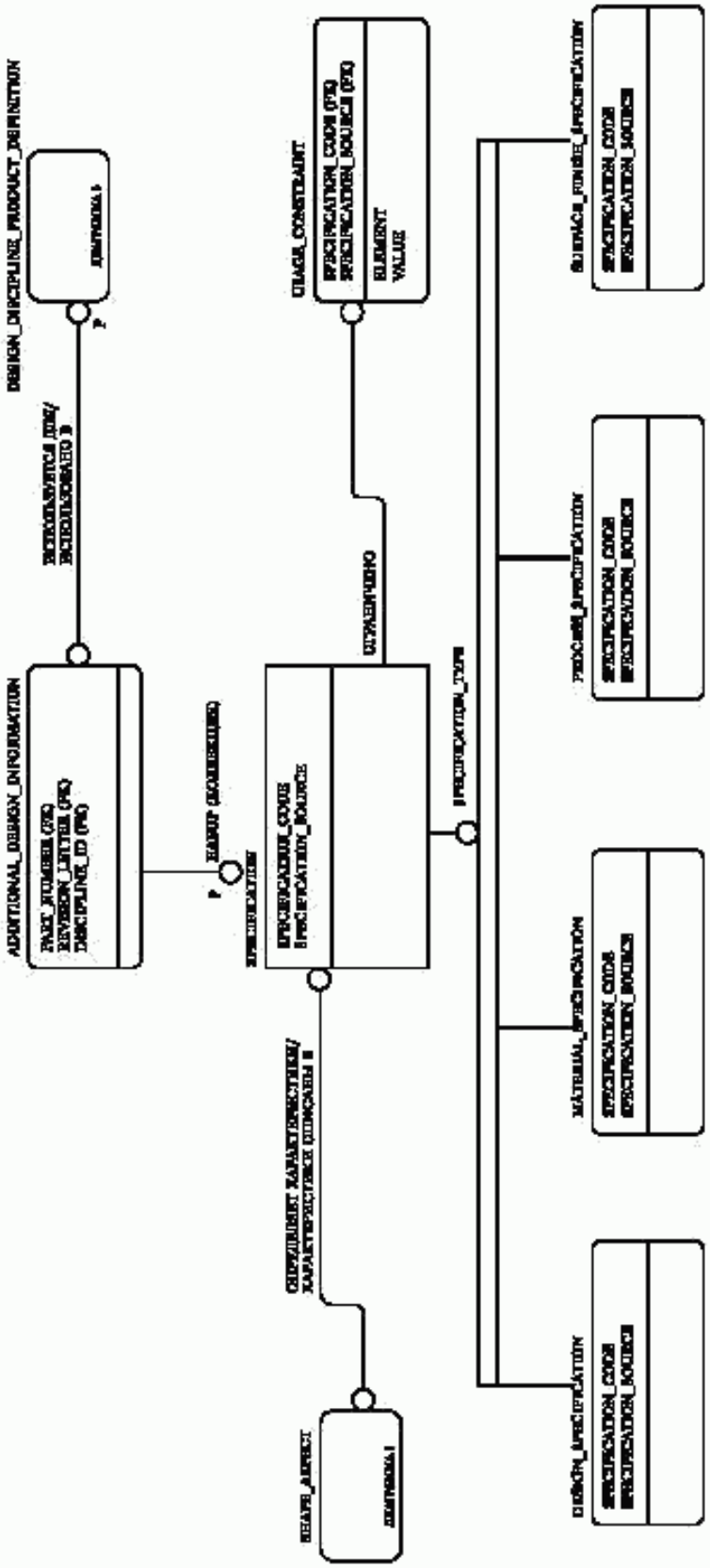


Рисунок G.4 — IDEF1X-диаграмма ПЭМ 4 из 7

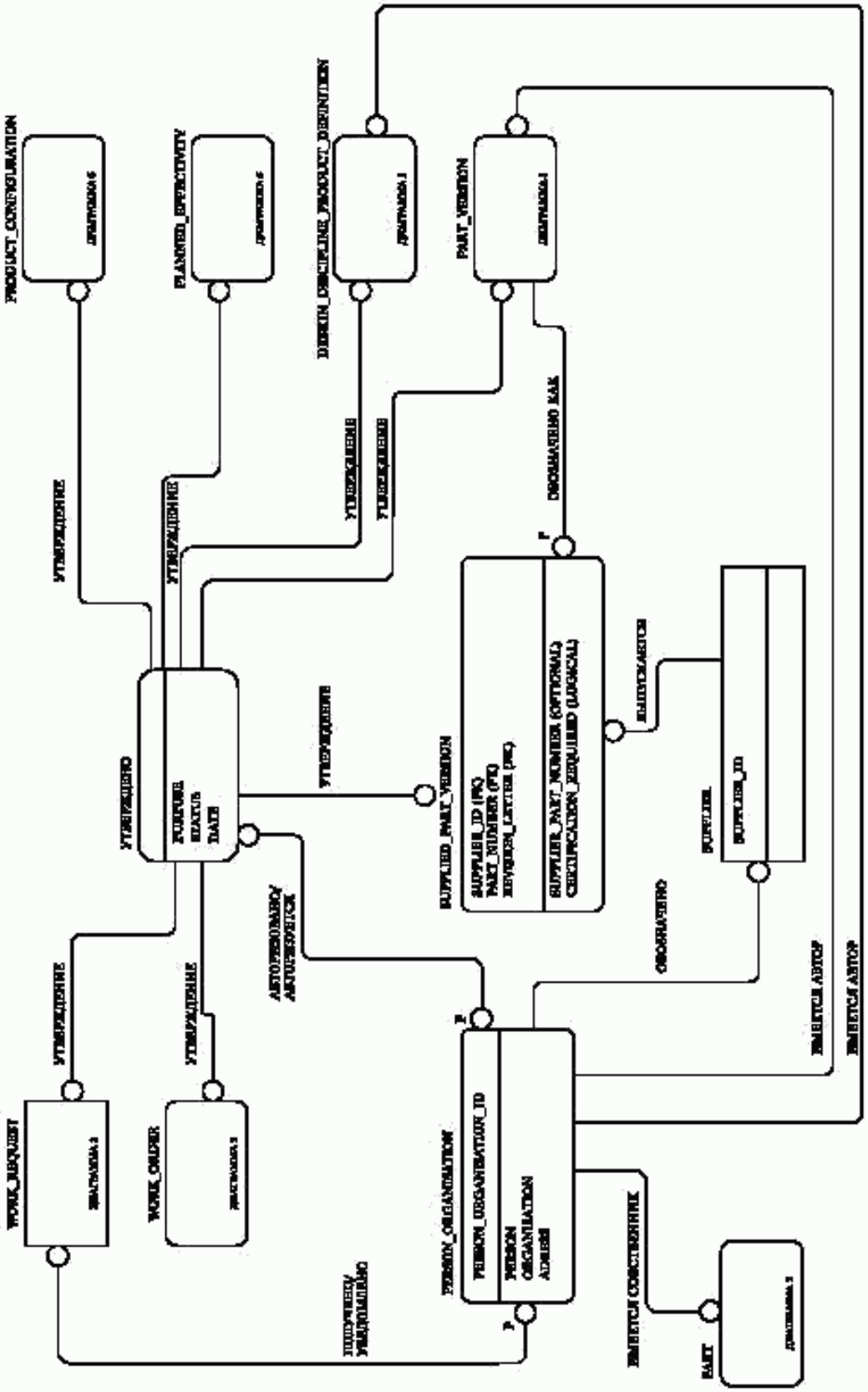


Рисунок G.5 — IDEF1X-диаграмма ПЭМ 5 из 7

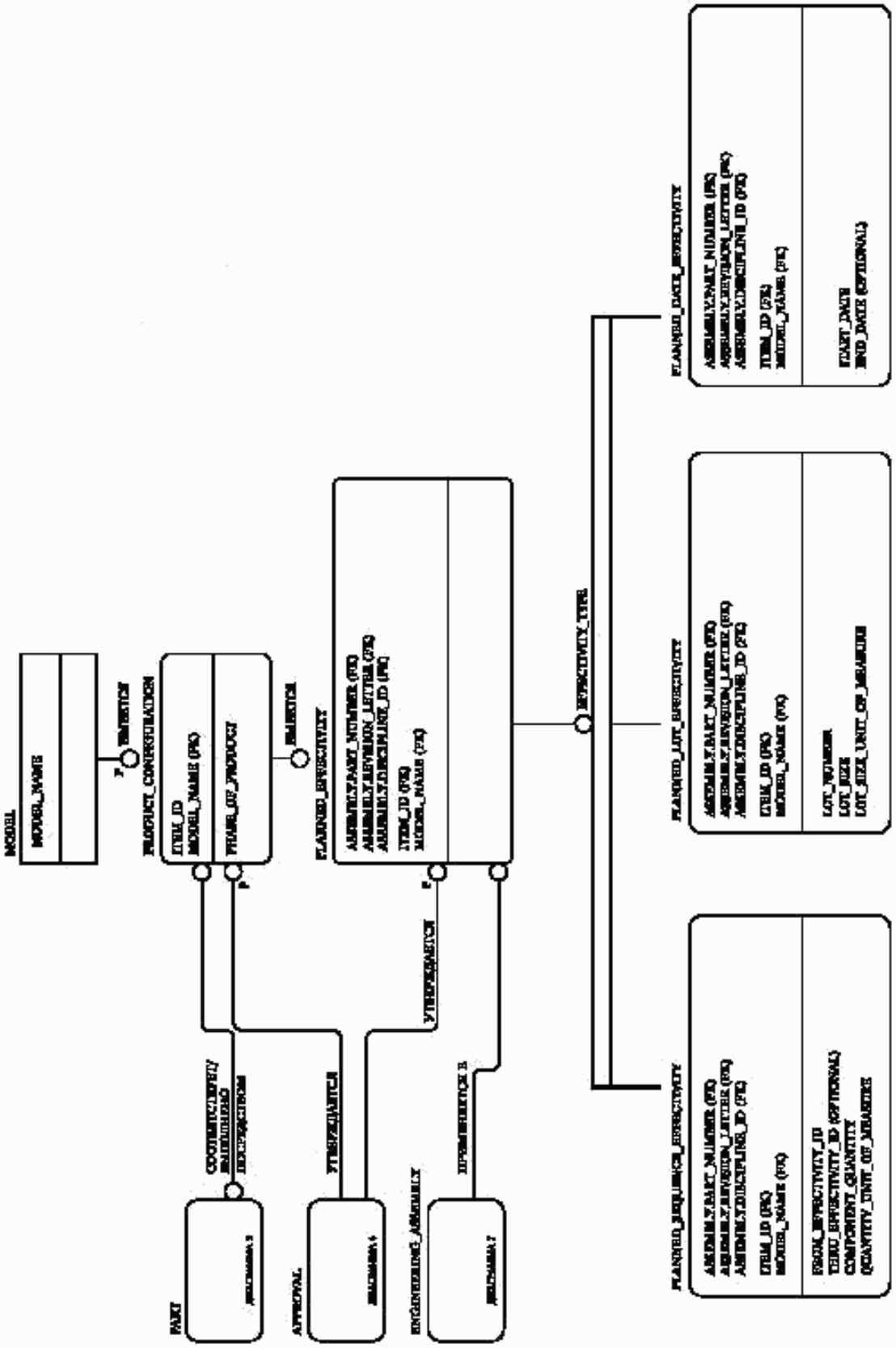


Рисунок G.6 — IDEF1X-диаграмма ПЭМ 6 из 7

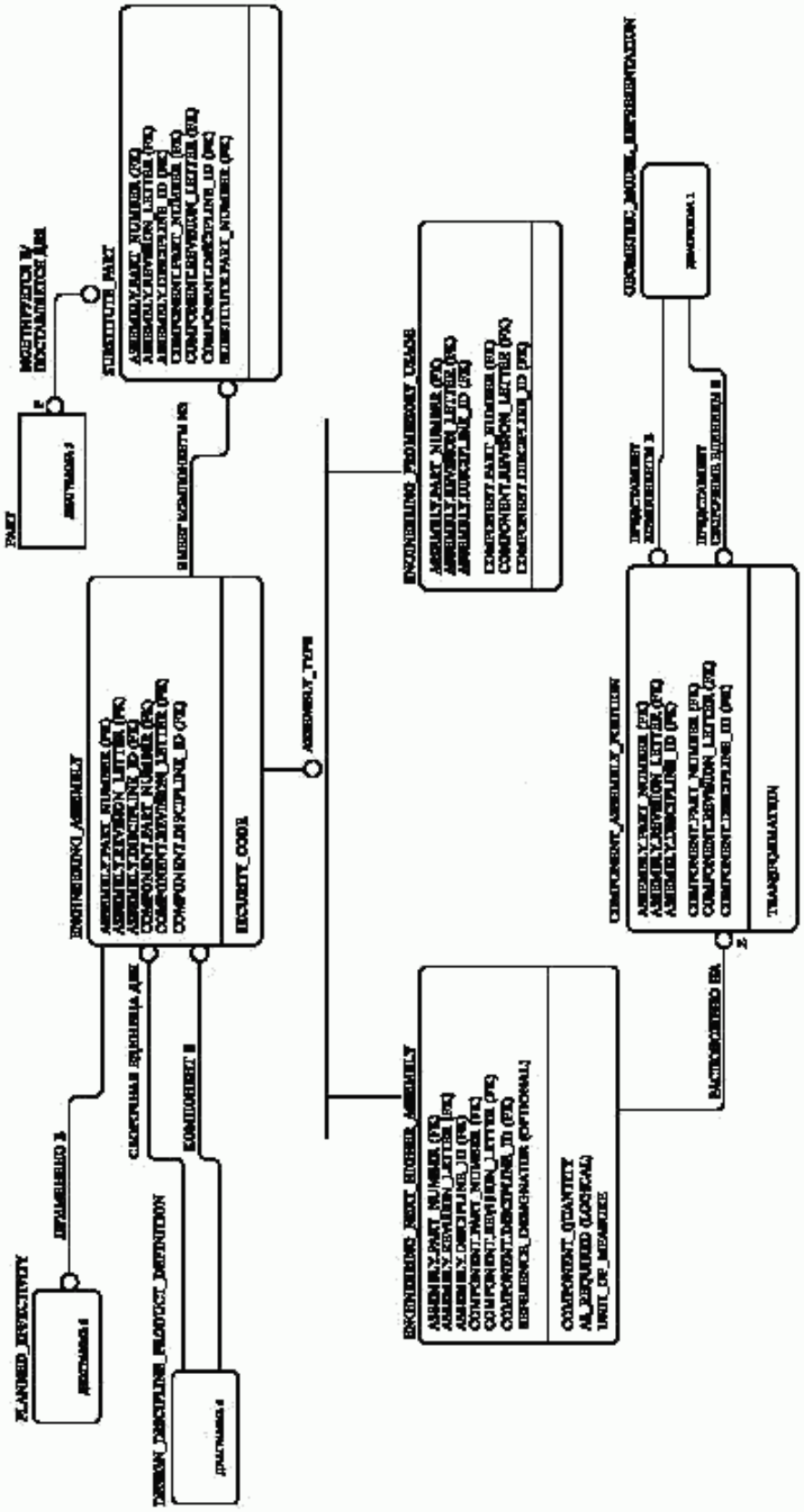


Рисунок G.7 — IDEF1X-диаграмма ПЭМ 7 из 7

ПРИЛОЖЕНИЕ Н
(справочное)

EXPRESS-G диаграммы

В настоящем приложении приведены диаграммы (см. рисунки Н.1 - Н.39), соответствующие развернутому EXPRESS-листингу прикладной интерпретированной модели (ПИМ), приведенному в приложении А. В этих диаграммах использована графическая нотация EXPRESS-G языка EXPRESS. Правила построения EXPRESS-G диаграмм установлены в приложении D ГОСТ Р ИСО 10303-11. Обратите внимание, что межстраничные ссылки между этими диаграммами заданы по номеру диаграммы, а не по номеру рисунка, например (2) или (39), а не (Н.2) или (Н.39).

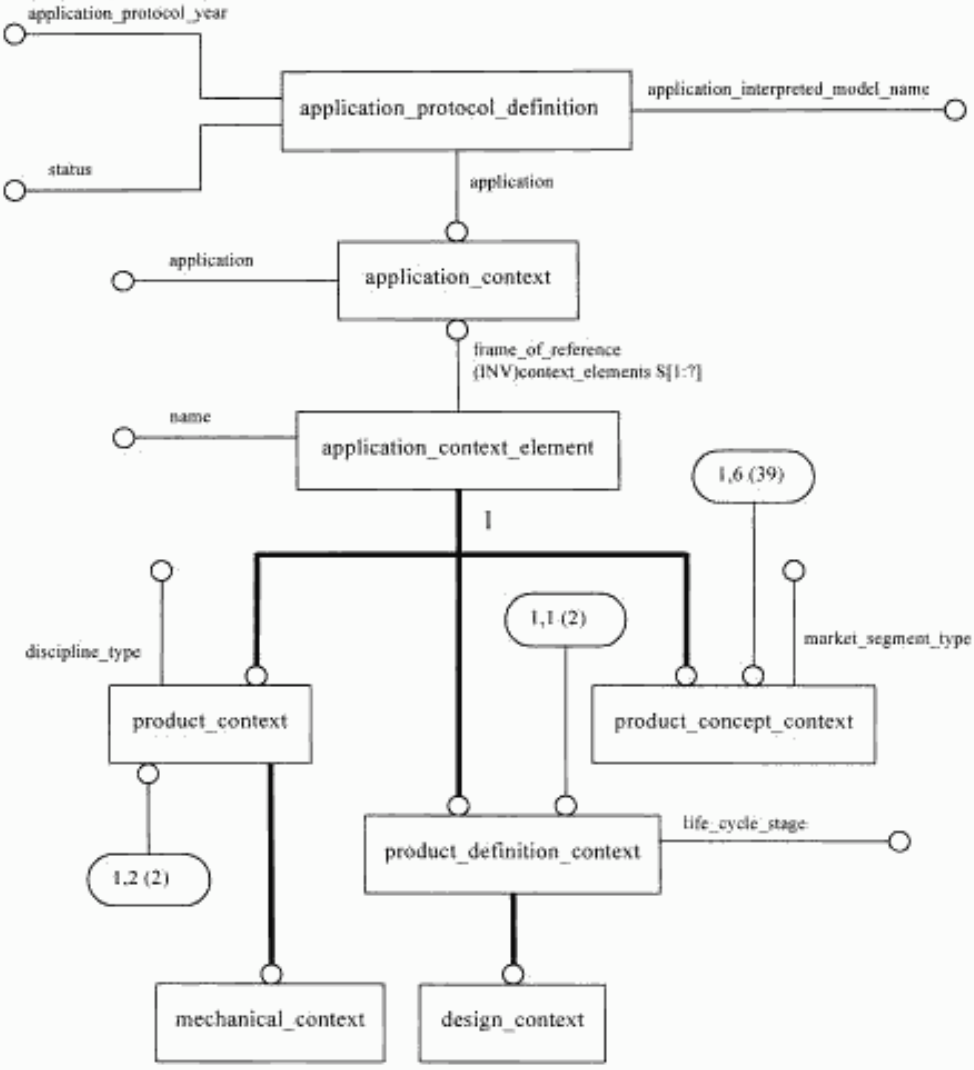


Рисунок Н.1 — **application_context** — ПИМ EXPRESS-G диаграмма 1 из 39

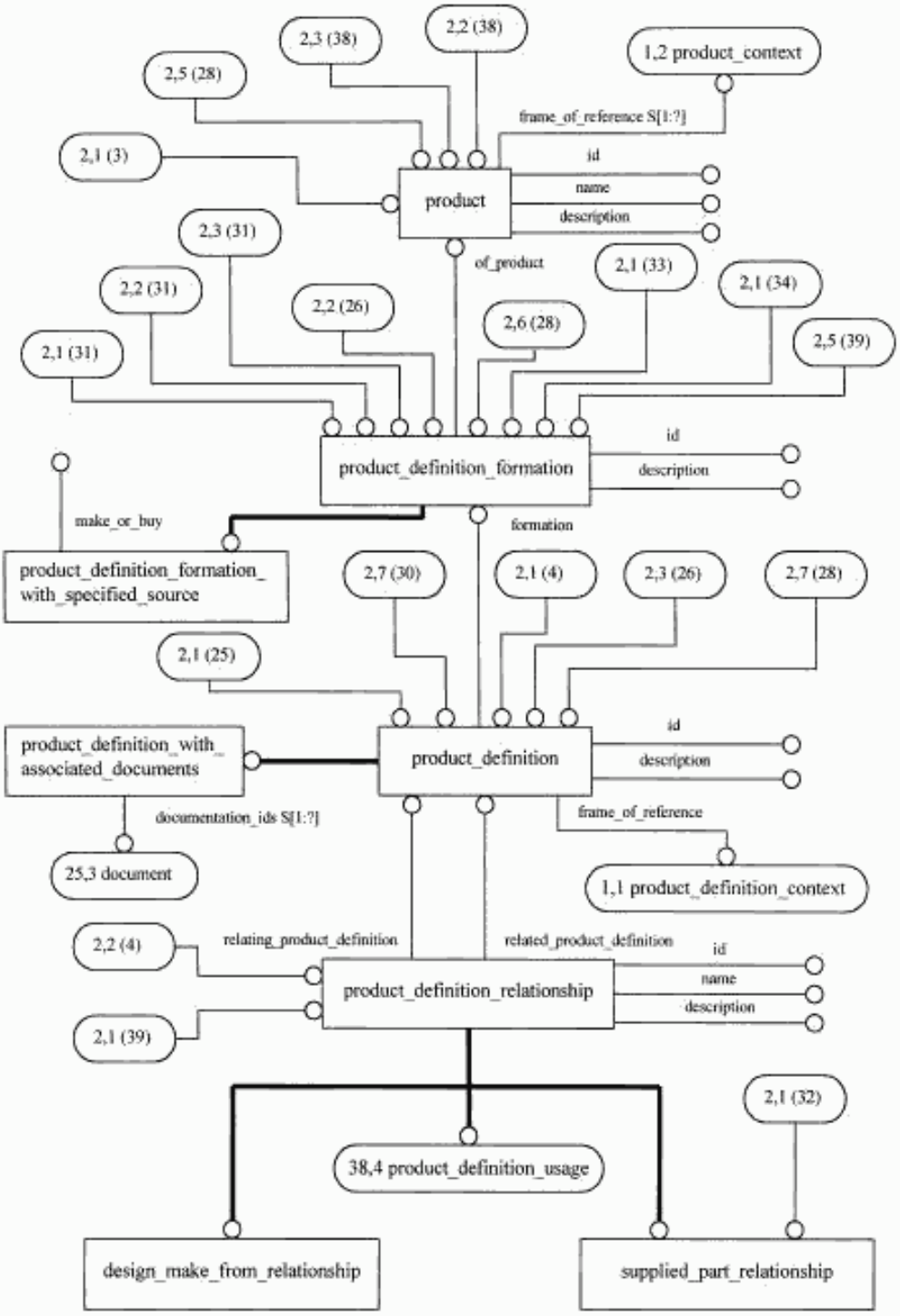


Рисунок H.2 — **product_definition** — ПИМ EXPRESS-G диаграмма 2 из 39

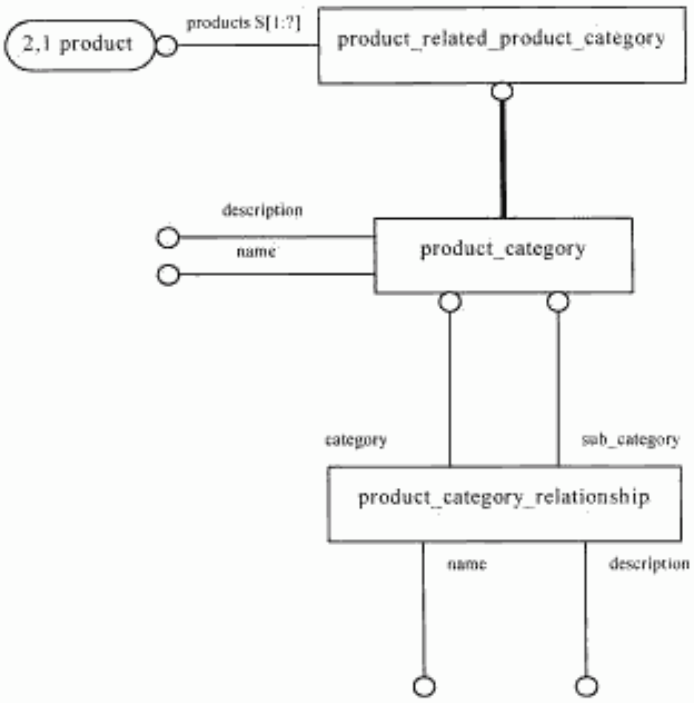


Рисунок H.3 — **product_category** — ПИМ EXPRESS-G диаграмма 3 из 39

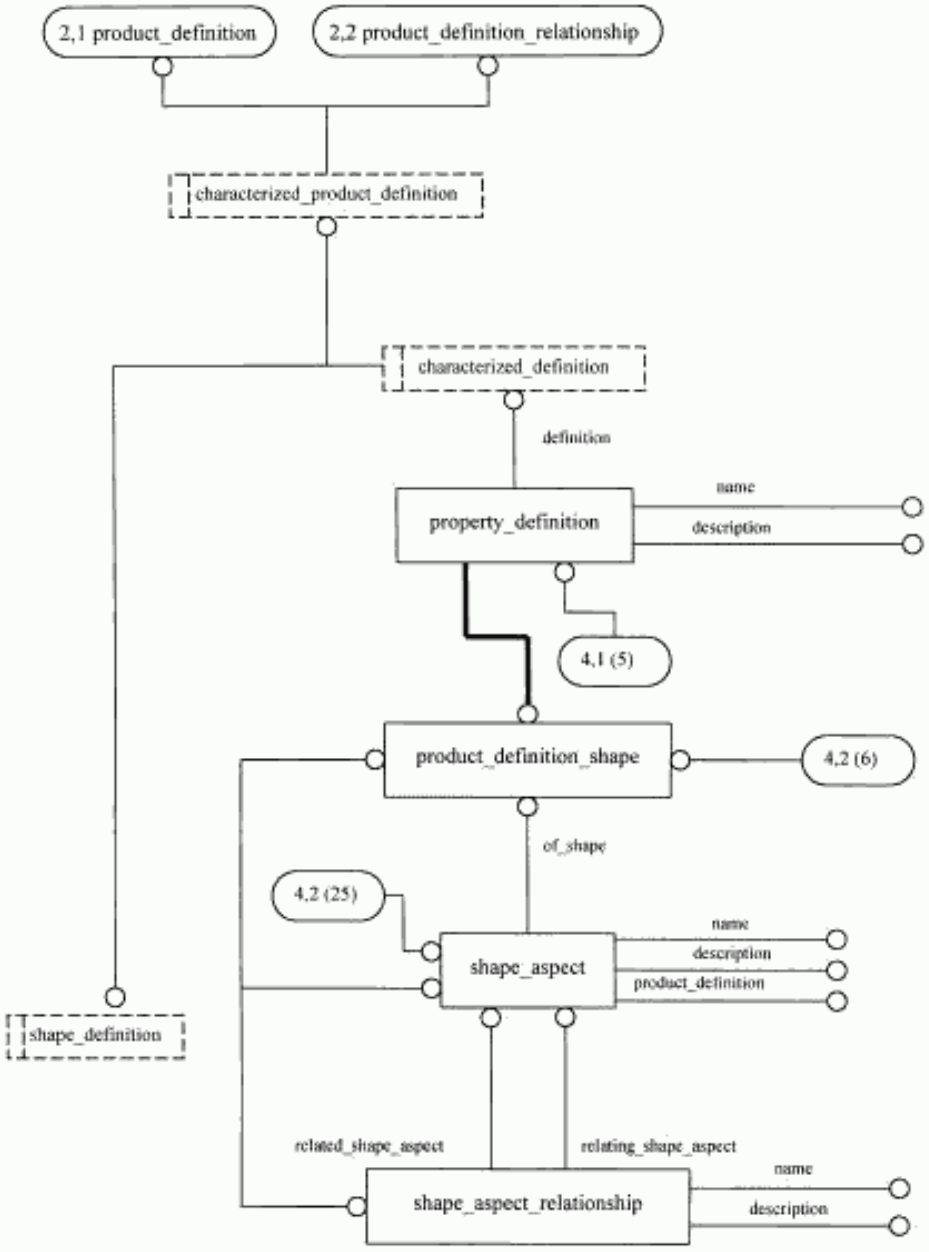


Рисунок Н.4 — **property_definition** — ПИМ EXPRESS-G диаграмма 4 из 39

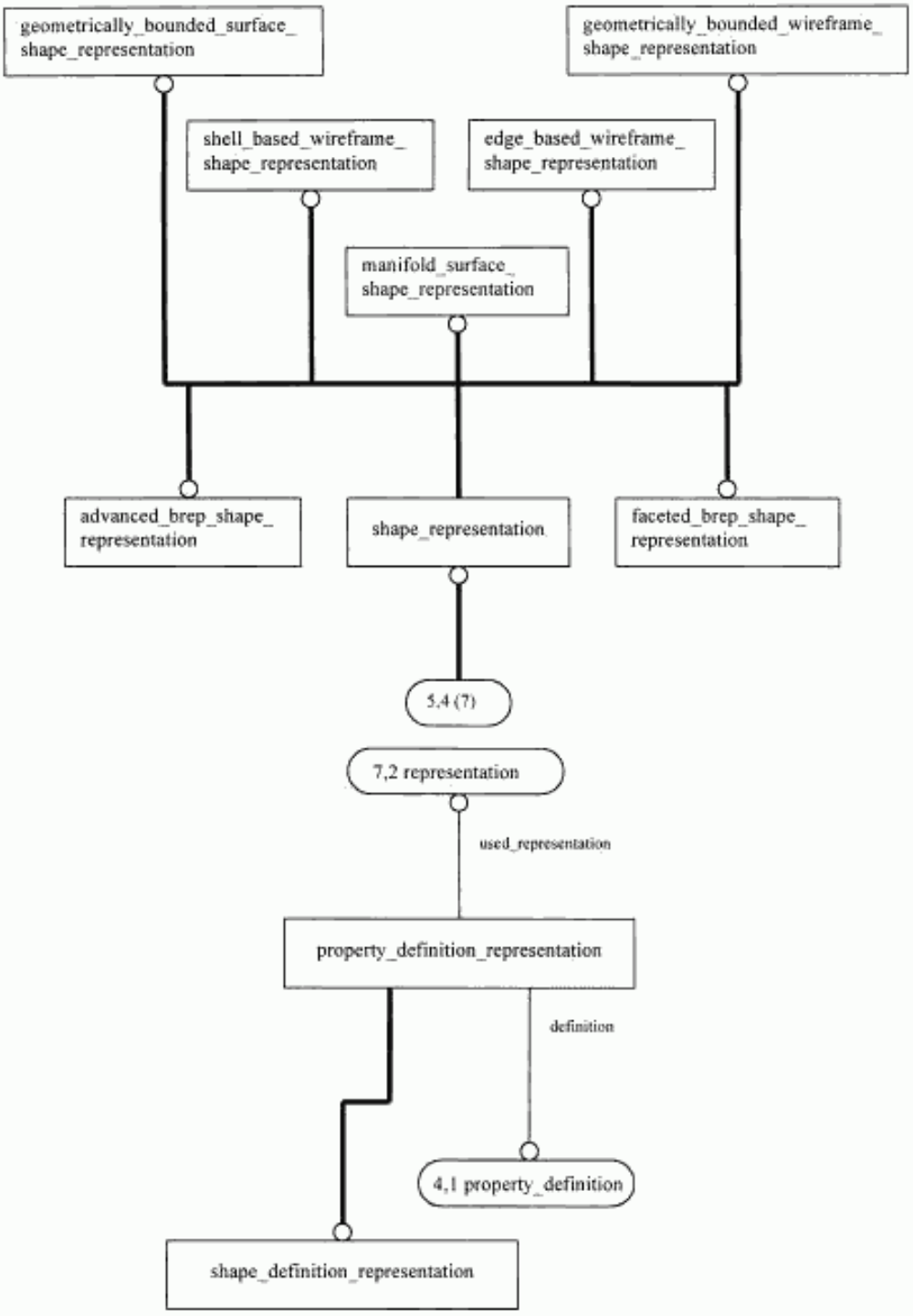


Рисунок H.5 — **property_representation** — ПИМ EXPRESS-G диаграмма 5 из 39

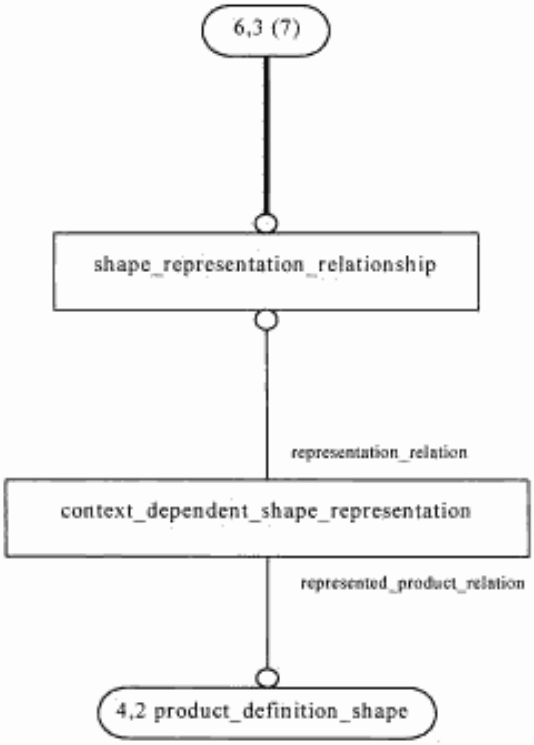


Рисунок Н.6 — **shape_representation_relationship** — ПИМ EXPRESS-G диаграмма 6 из 39

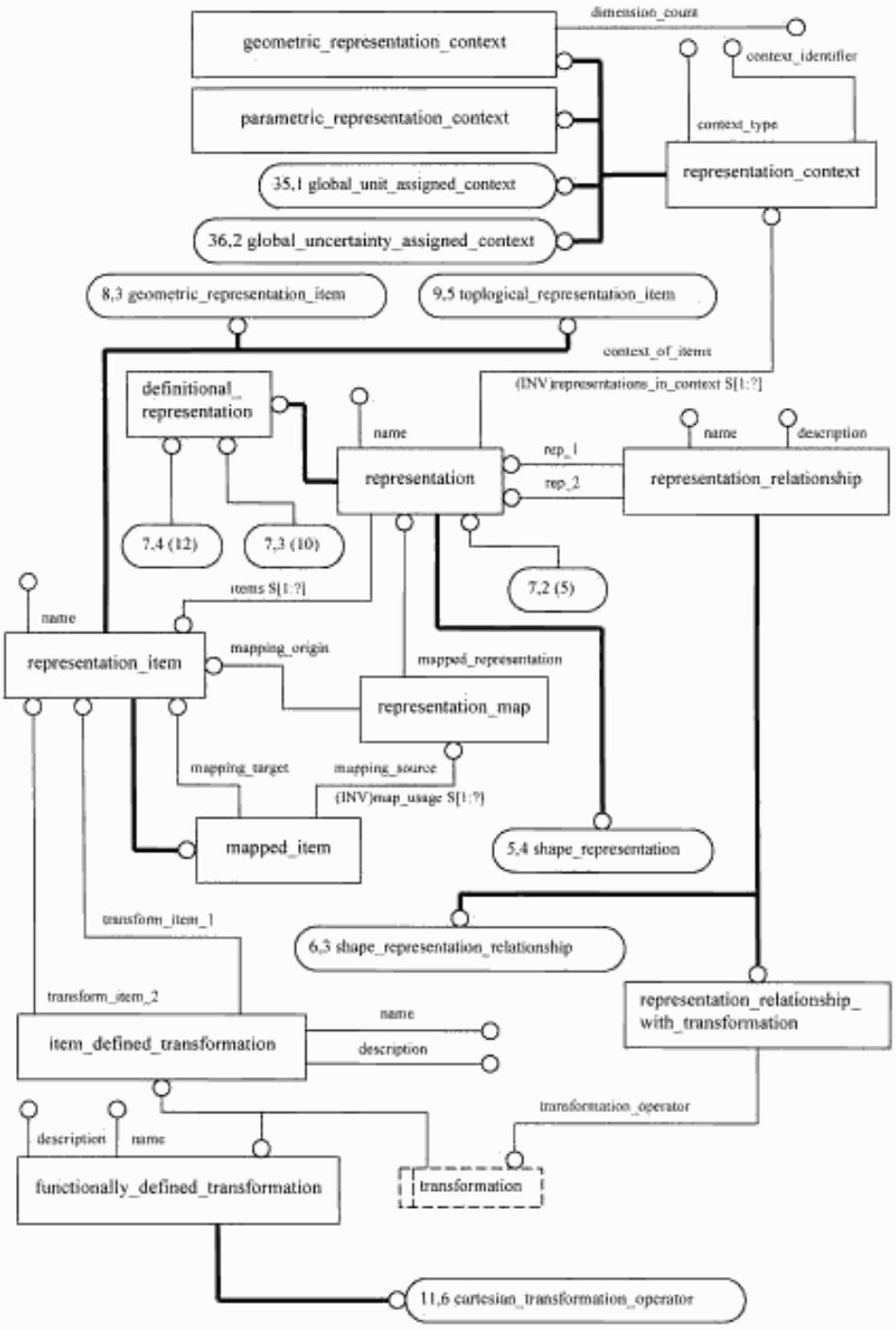


Рисунок Н.7 — **representation** — ПИМ EXPRESS-G диаграмма 7 из 39

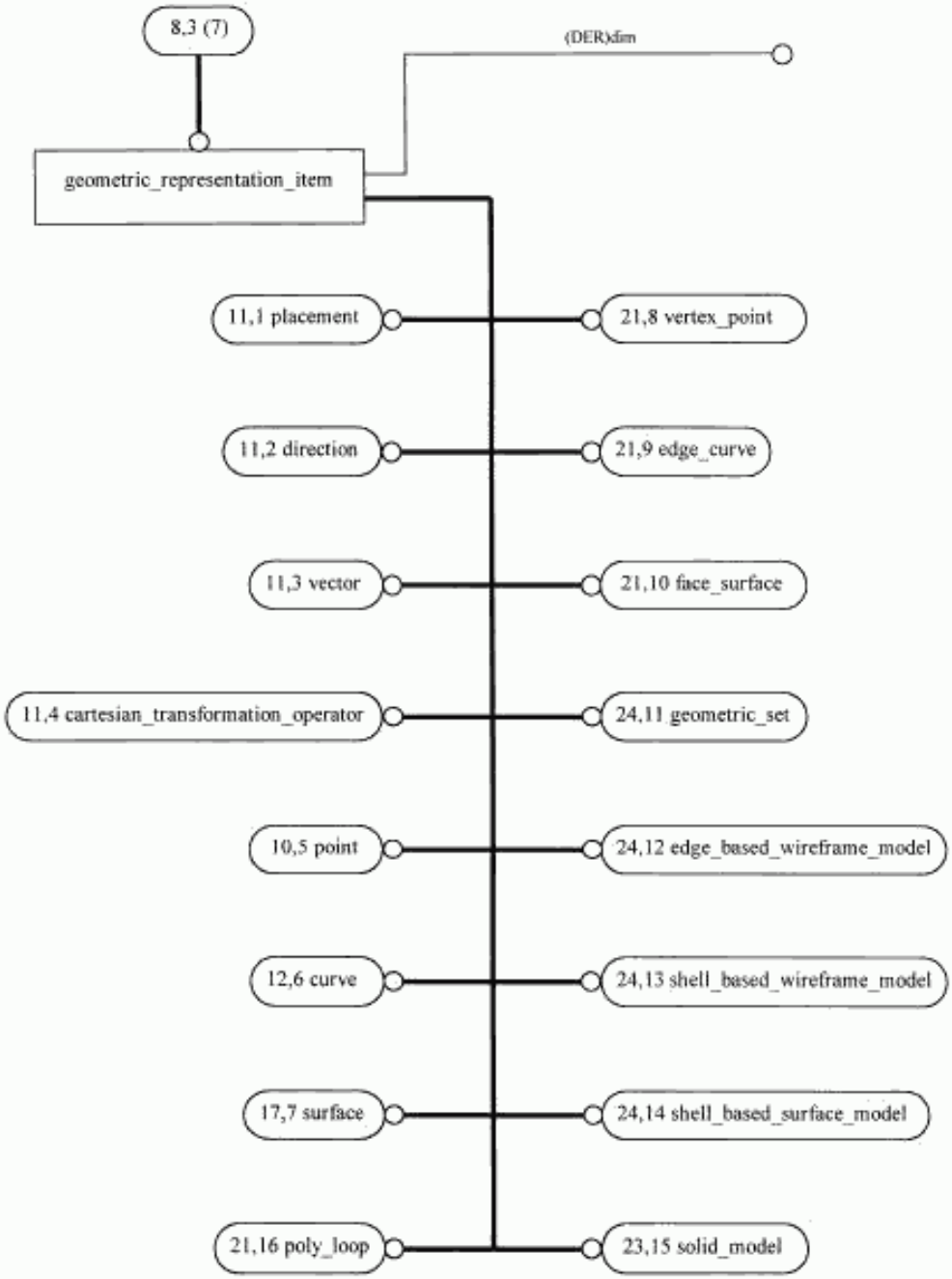
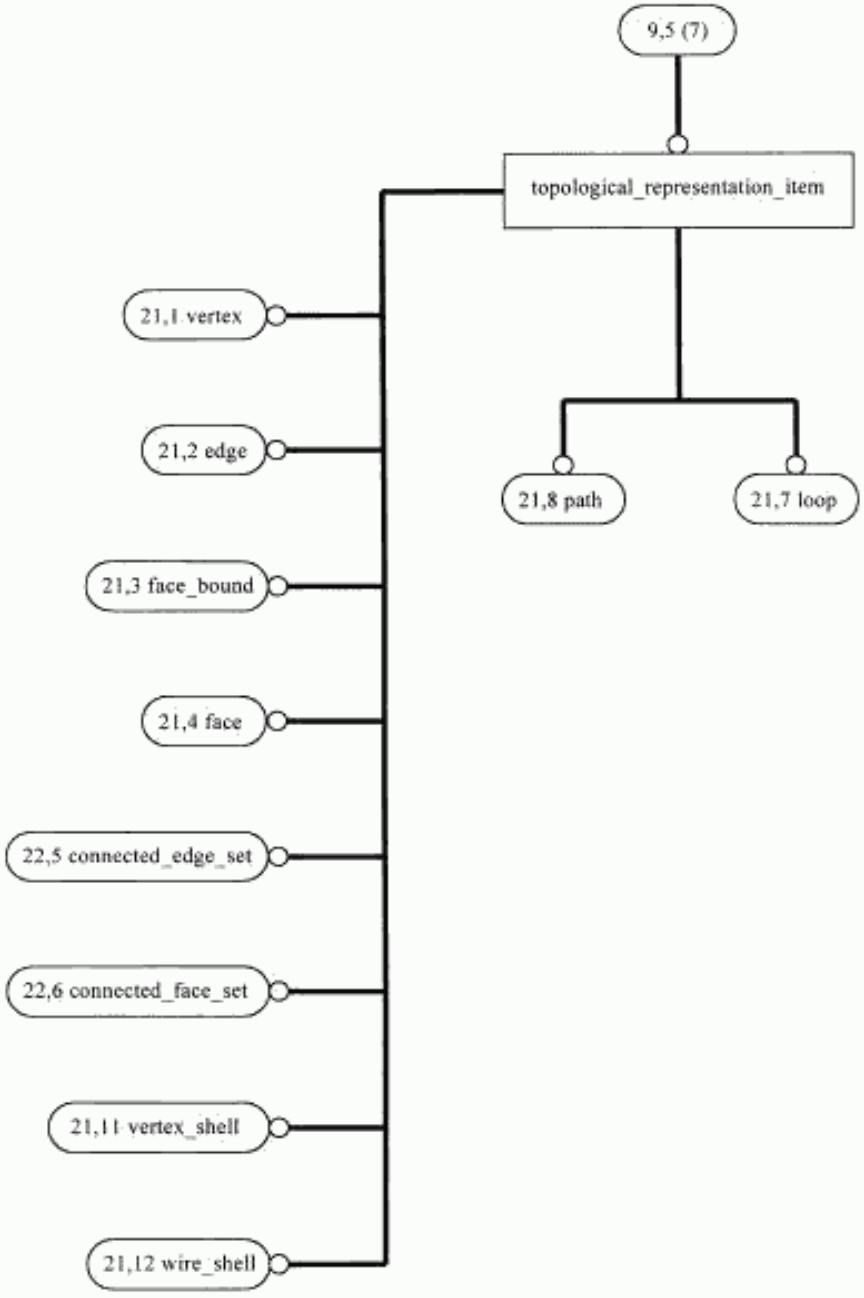


Рисунок Н.8 — **geometric_representation_items** — ПИМ EXPRESS-G диаграмма 8 из 39



Примечание — Данное "дерево" иллюстрирует цикл "ветви" оператора ANDOR, входящего в оператор ONEOF, связывающий объект в другом "дереве".

Рисунок Н.9 — **topological_representation_items** — ПИМ EXPRESS-G диаграмма 9 из 39

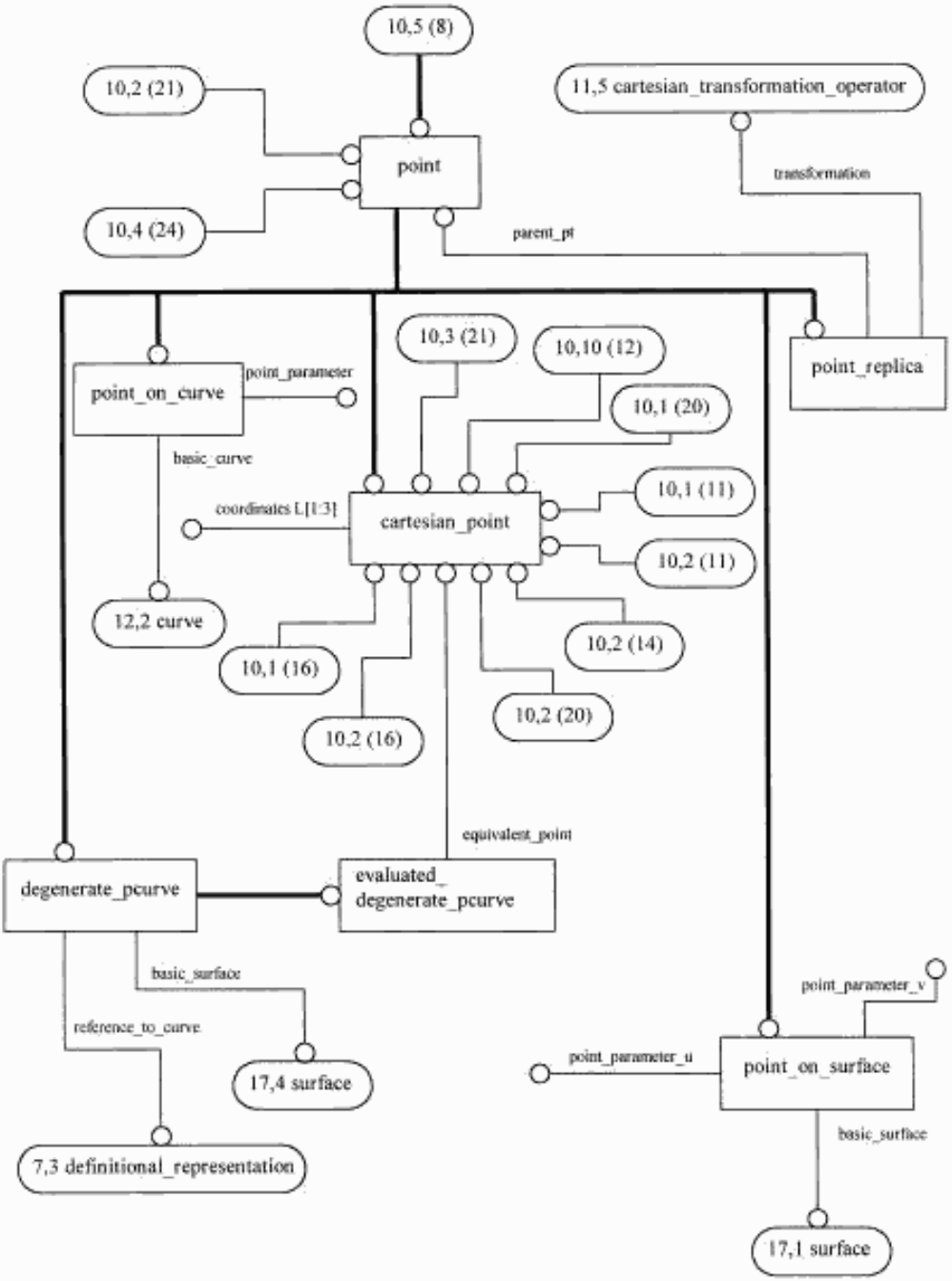


Рисунок H.10 — **point** — ПИМ EXPRESS-G диаграмма 10 из 39

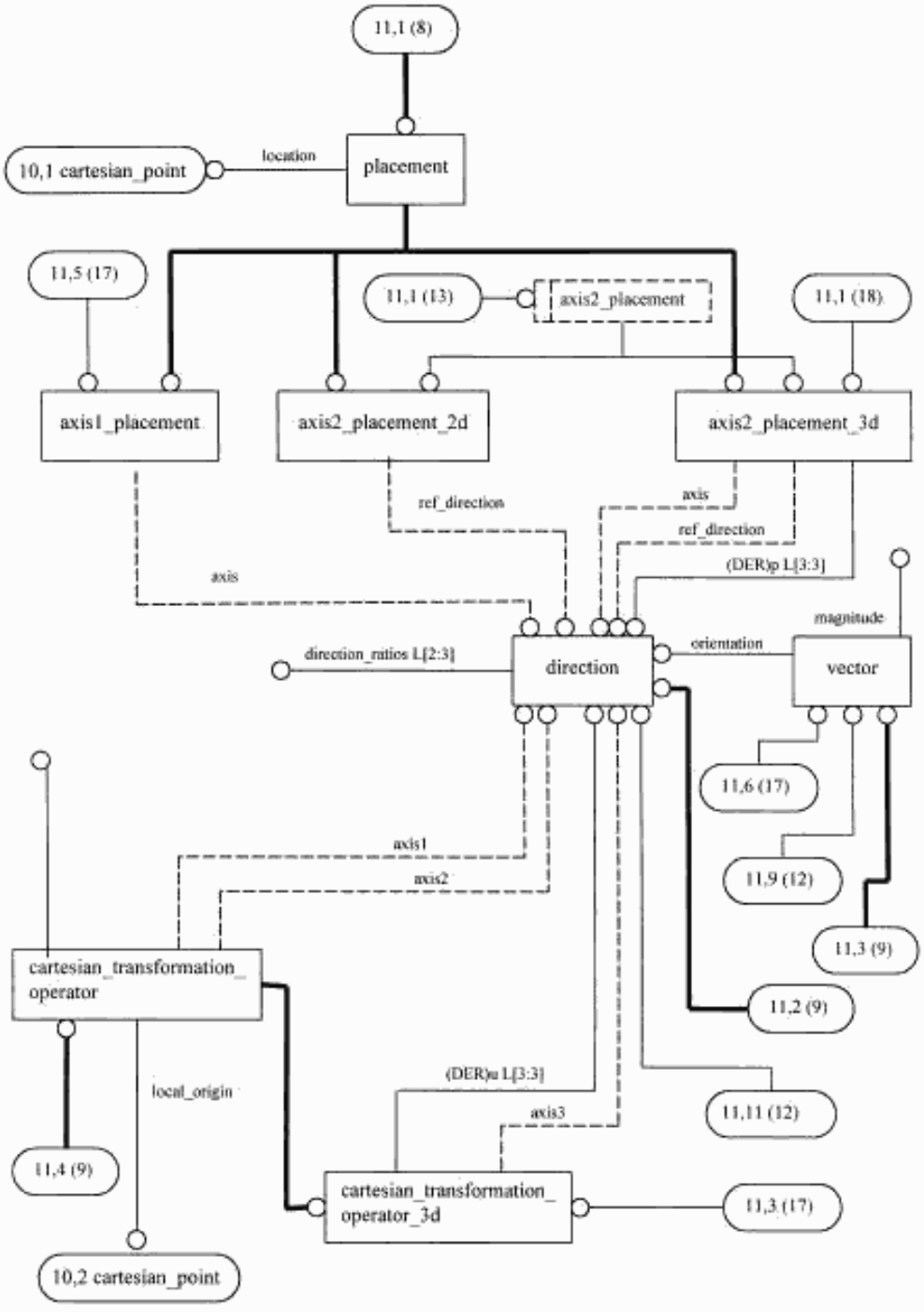


Рисунок Н.11 — **geometric_orientation** — ПИМ EXPRESS-G диаграмма 11 из 39

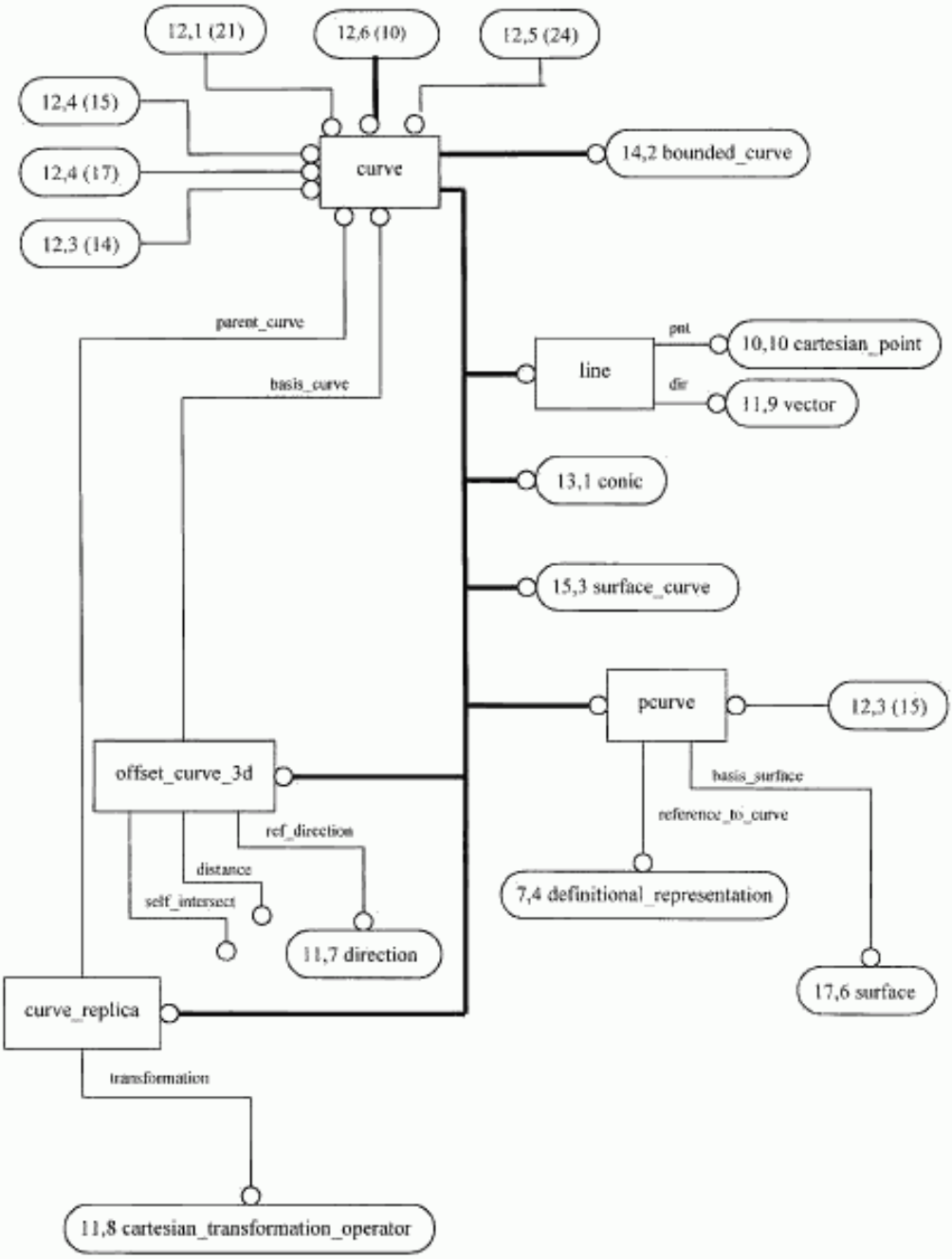


Рисунок H.12 — **curve** — ПИМ EXPRESS-G диаграмма 12 из 39

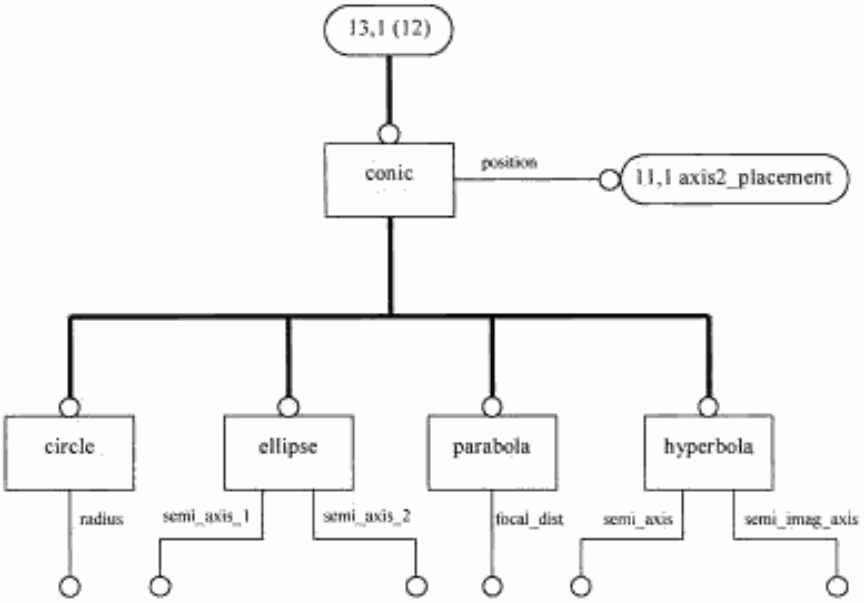


Рисунок Н.13 — **conic** — ПИМ EXPRESS-G диаграмма 13 из 39

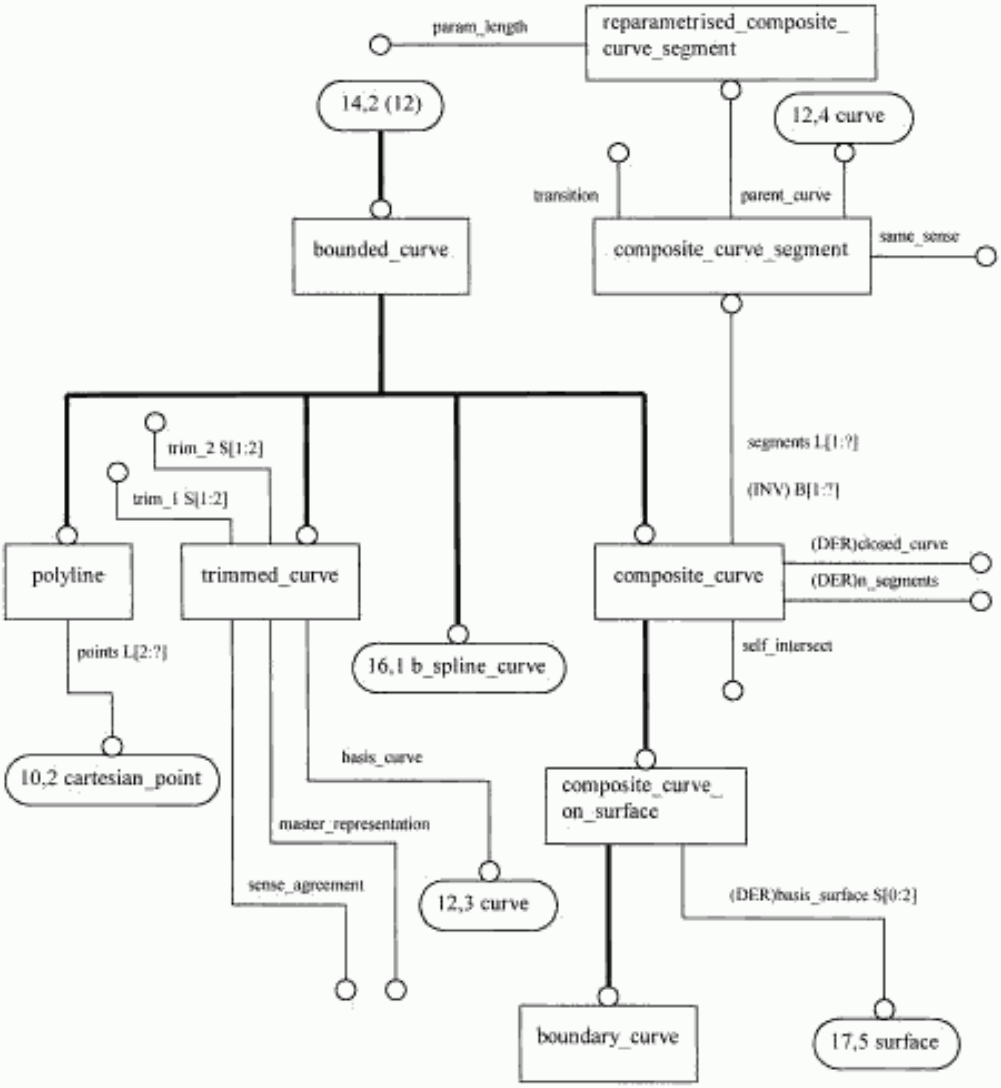


Рисунок H.14 — **bounded_curves** — ПИМ EXPRESS-G диаграмма 14 из 39

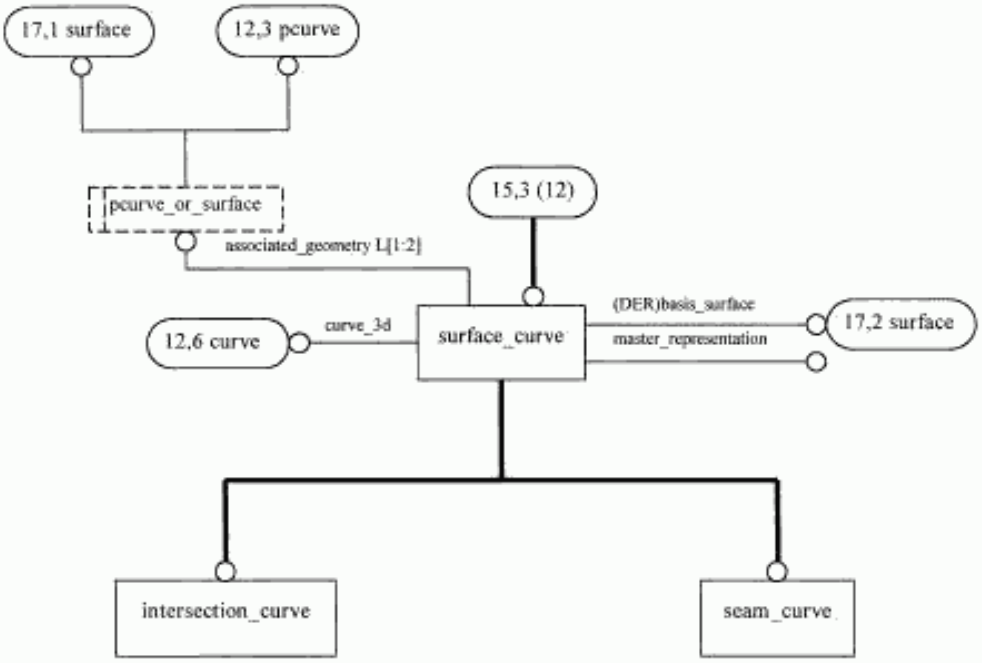


Рисунок H.15 — **surfase_curve** — ПИМ EXPRESS-G диаграмма 15 из 39

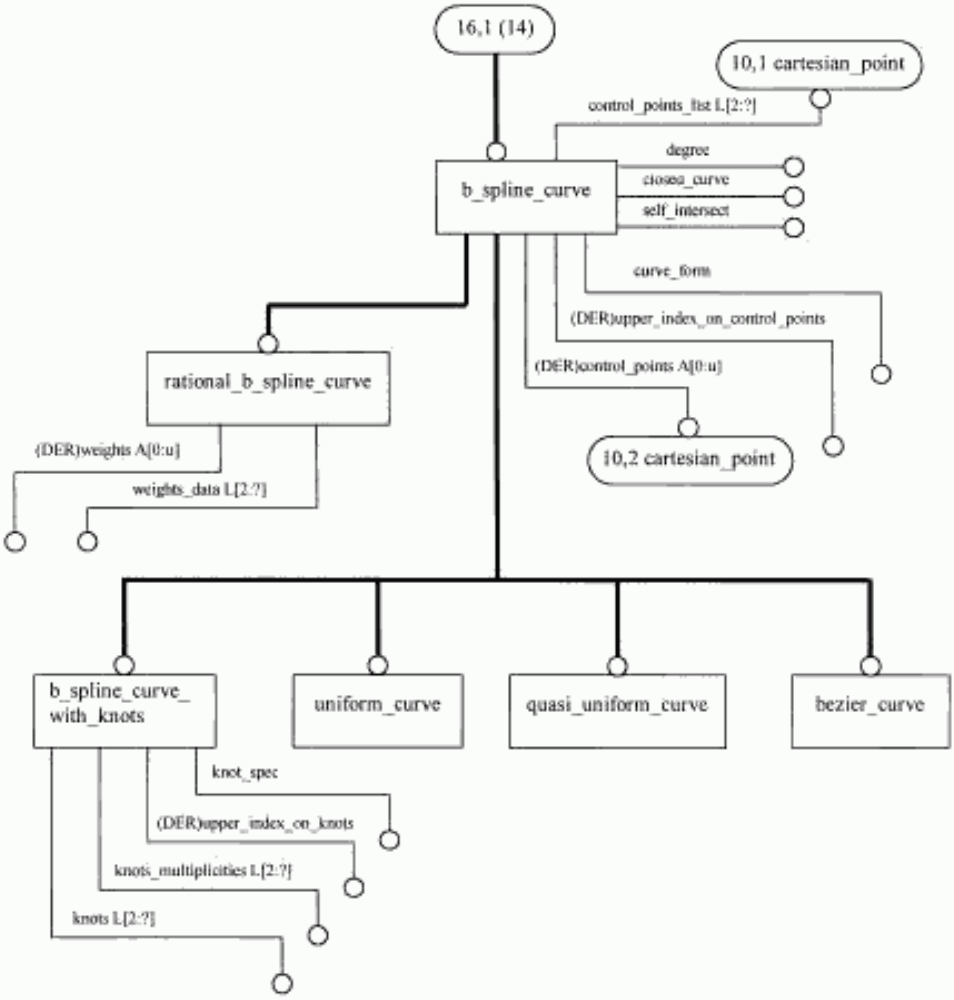


Рисунок H.16 — **b_spline_curve** — ПИМ EXPRESS-G диаграмма 16 из 39

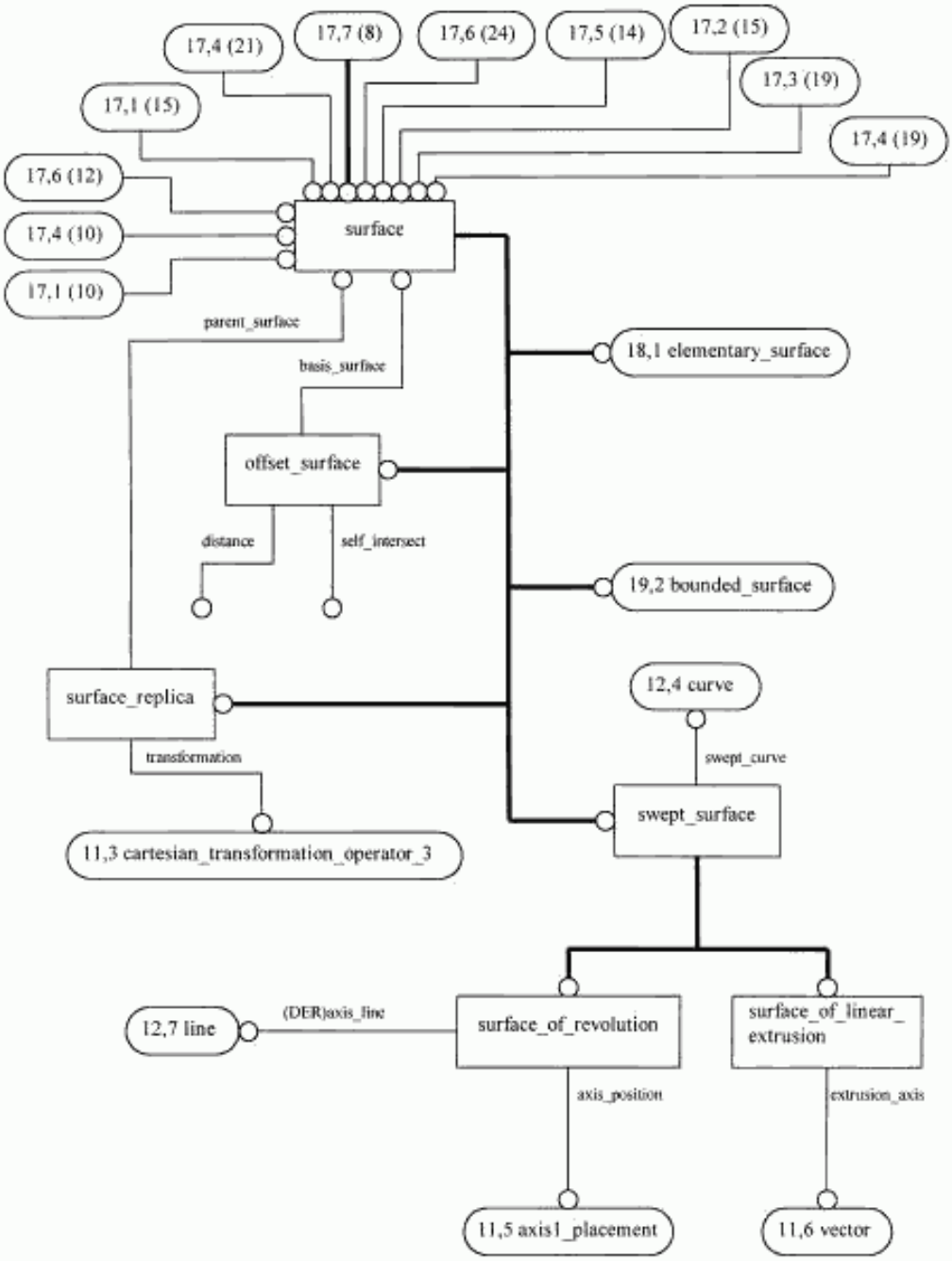


Рисунок H.17 — **surface** — ПИМ EXPRESS-G диаграмма 17 из 39

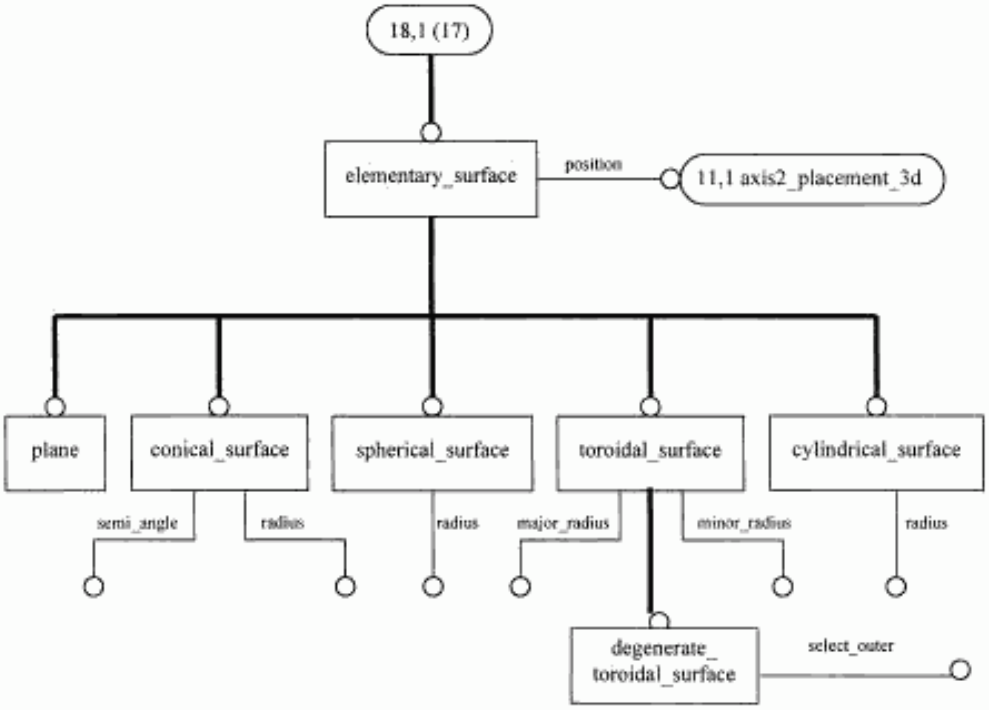


Рисунок H.18 — **elementary_surfaces** — ПИМ EXPRESS-G диаграмма 18 из 39

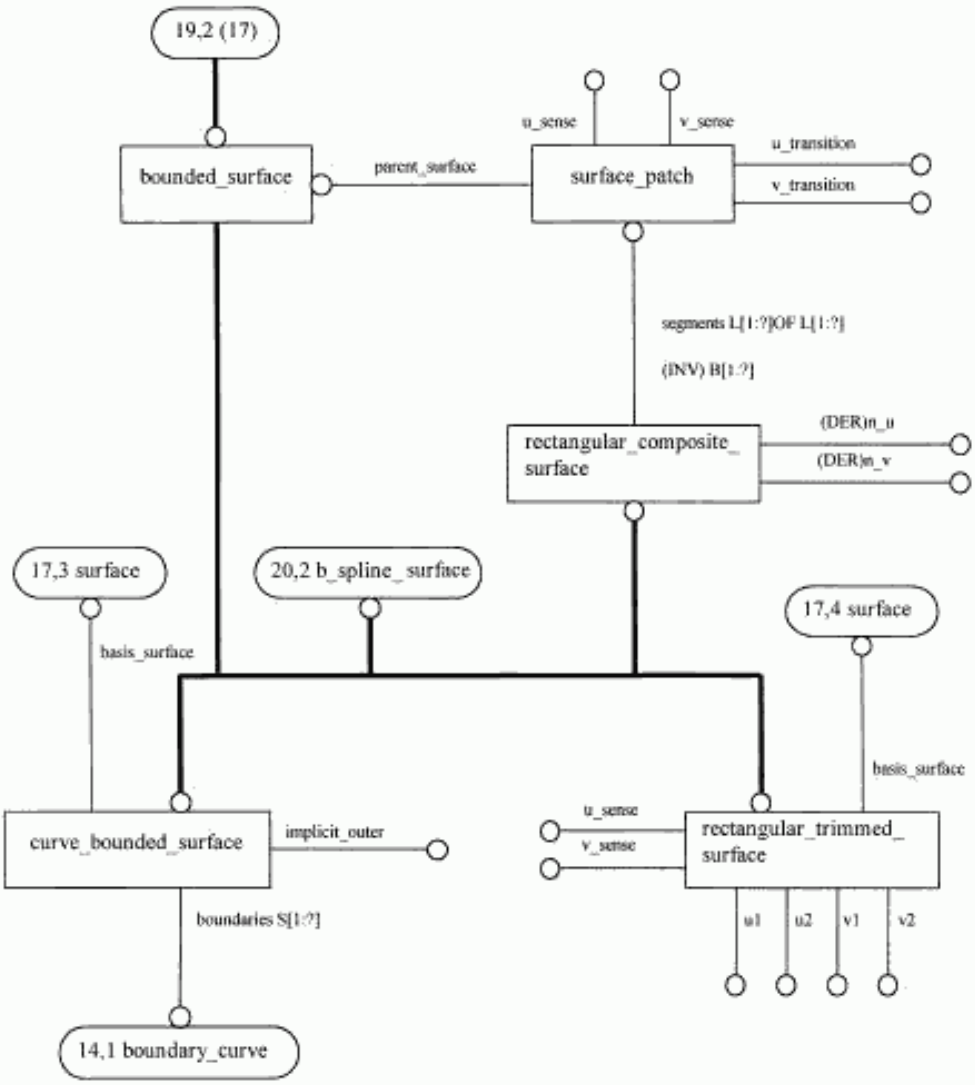


Рисунок H.19 — **bounded_surface** — ПИМ EXPRESS-G диаграмма 19 из 39

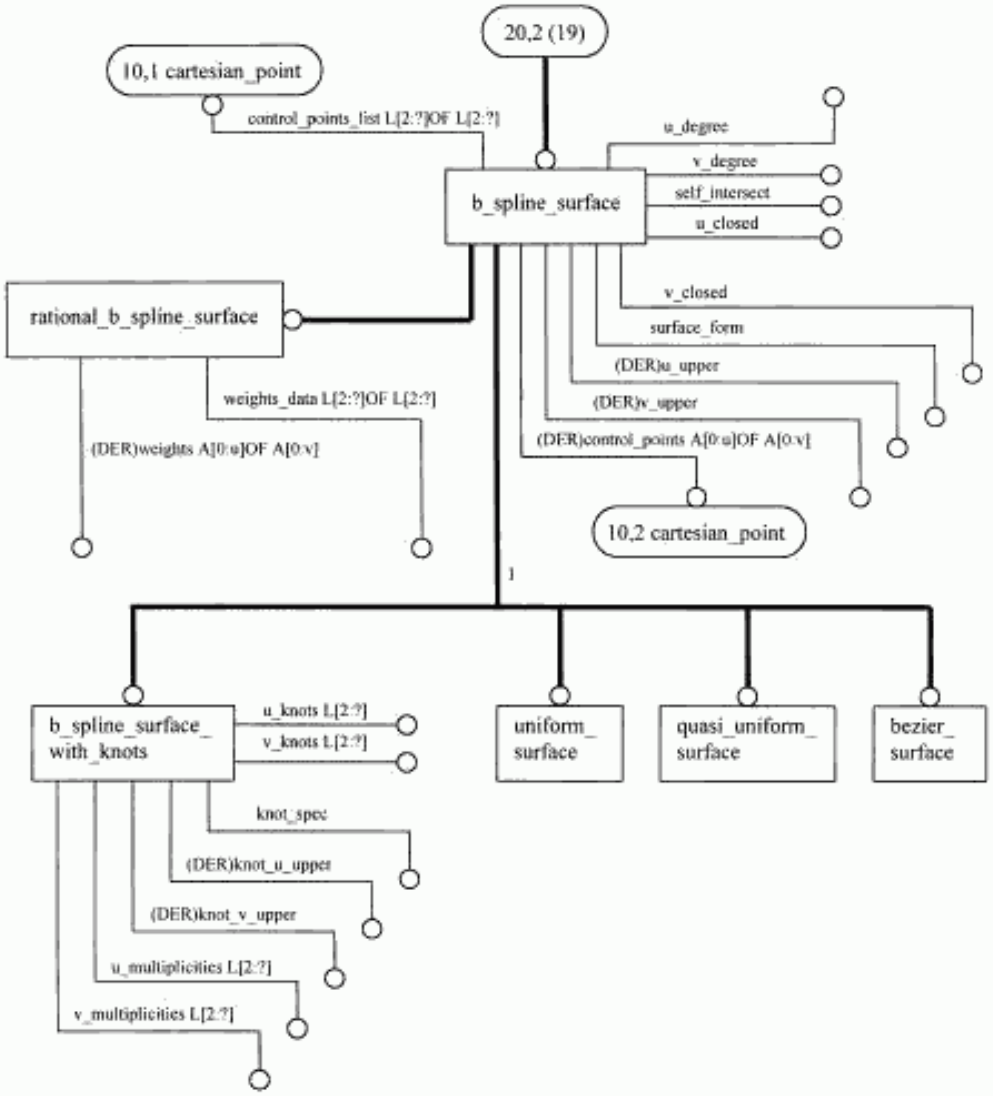


Рисунок H.20 — **b_spline_surface** — ПИМ EXPRESS-G диаграмма 20 из 39

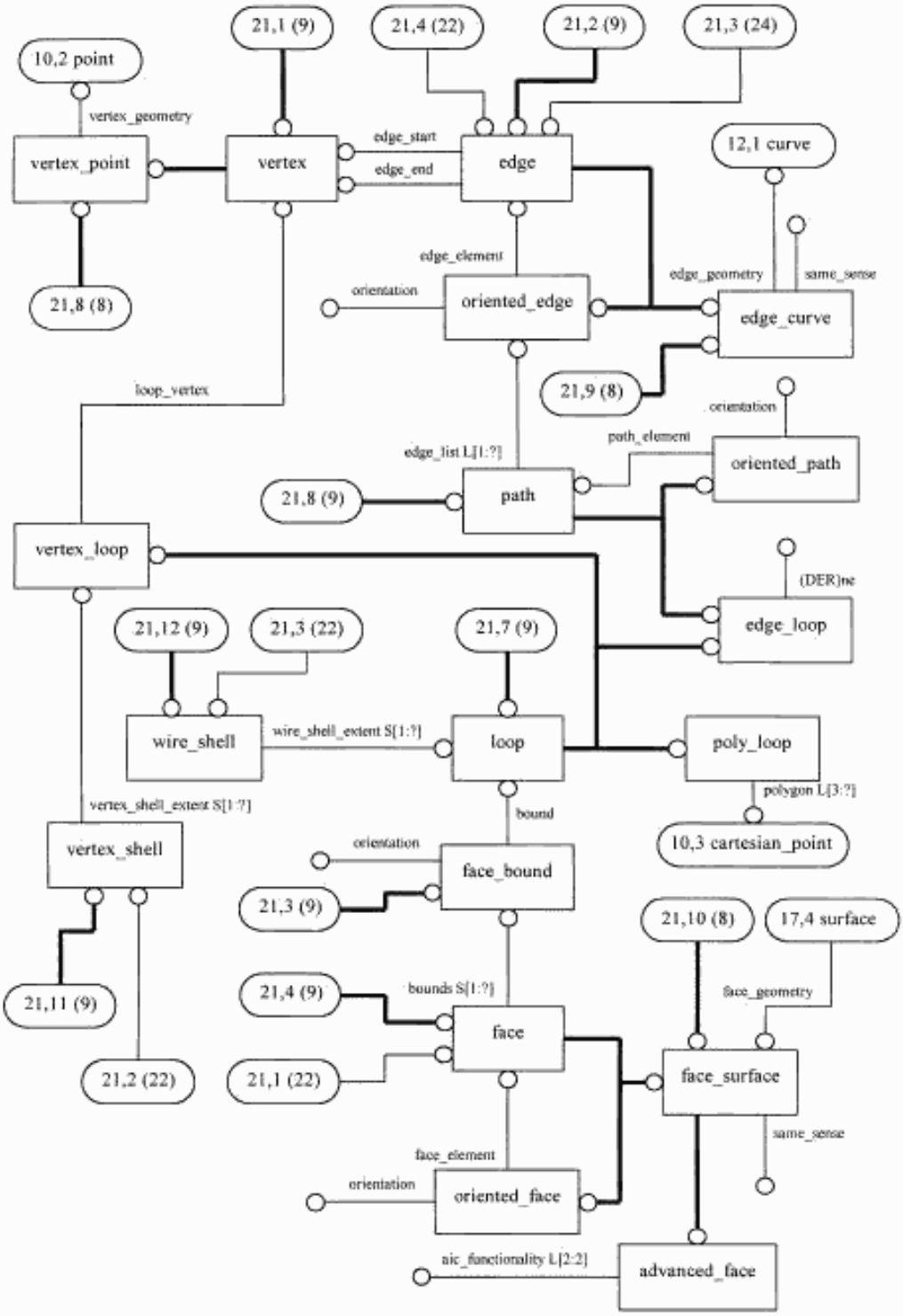


Рисунок H.21 — **topology** — ПИМ EXPRESS-G диаграмма 21 из 39

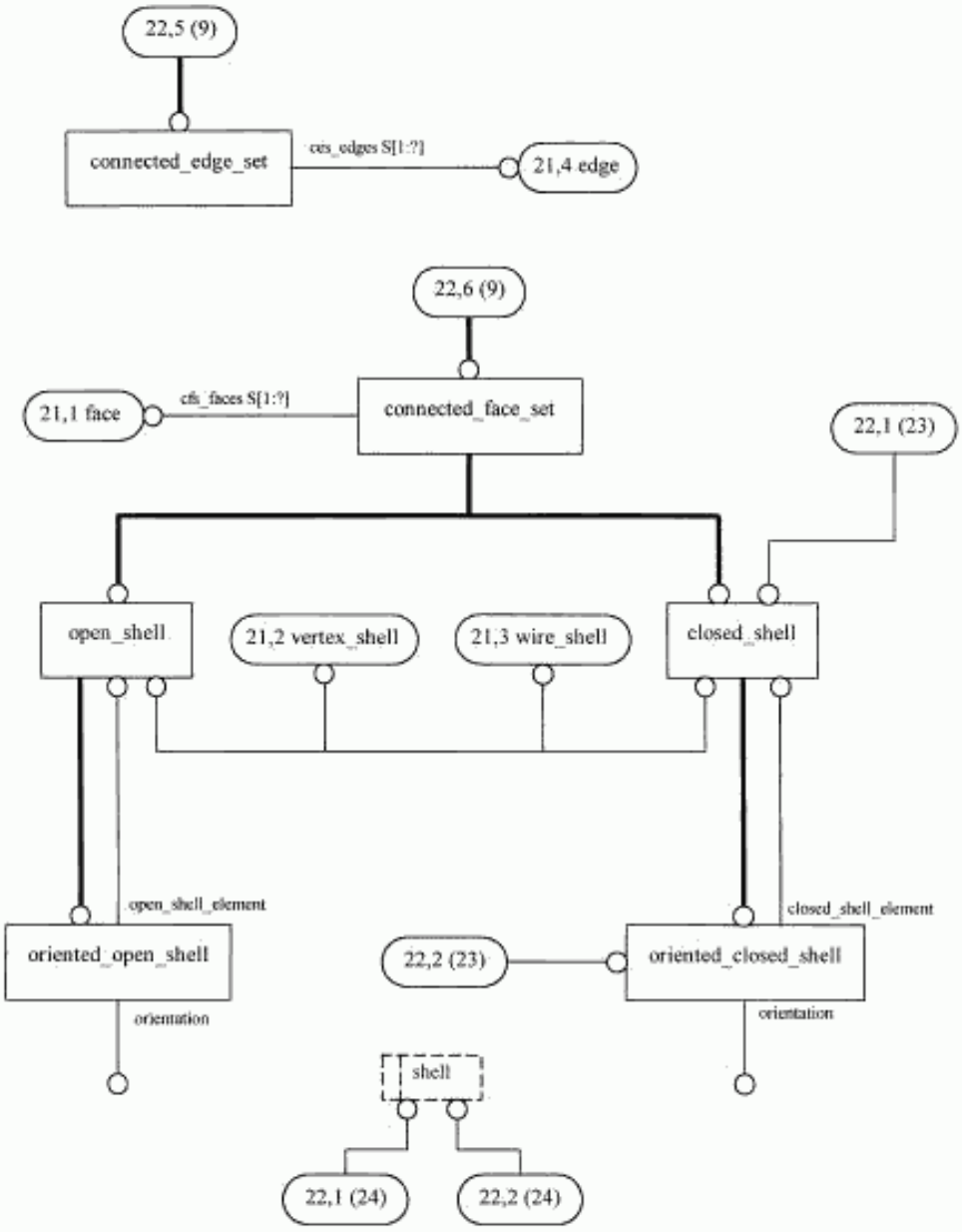


Рисунок H.22 — **shell** — ПИМ EXPRESS-G диаграмма 22 из 39

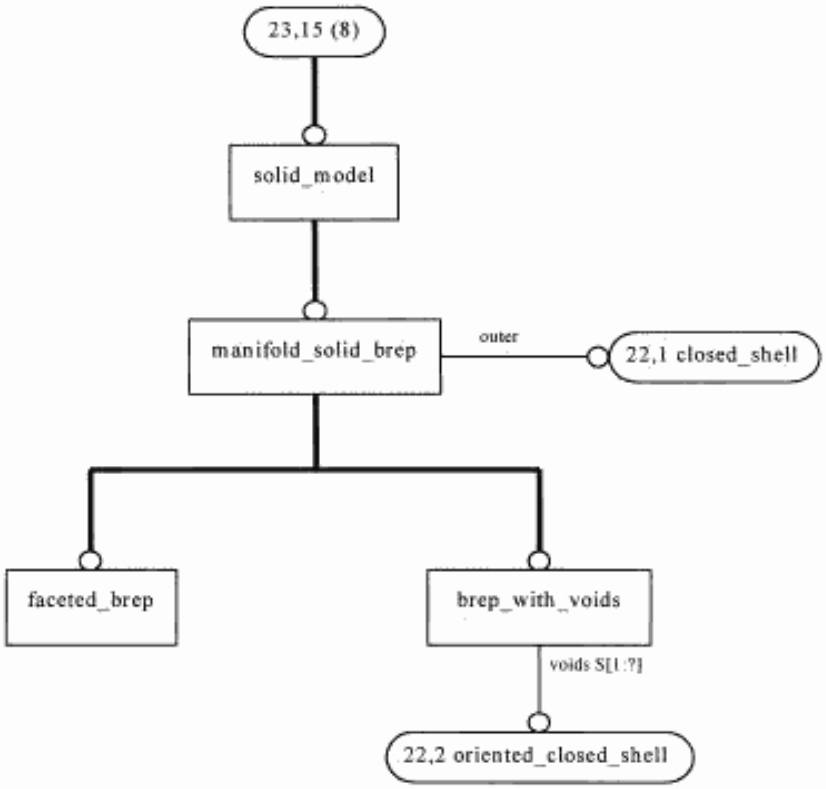


Рисунок H.23 — **solid_model** — ПИМ EXPRESS-G диаграмма 23 из 39

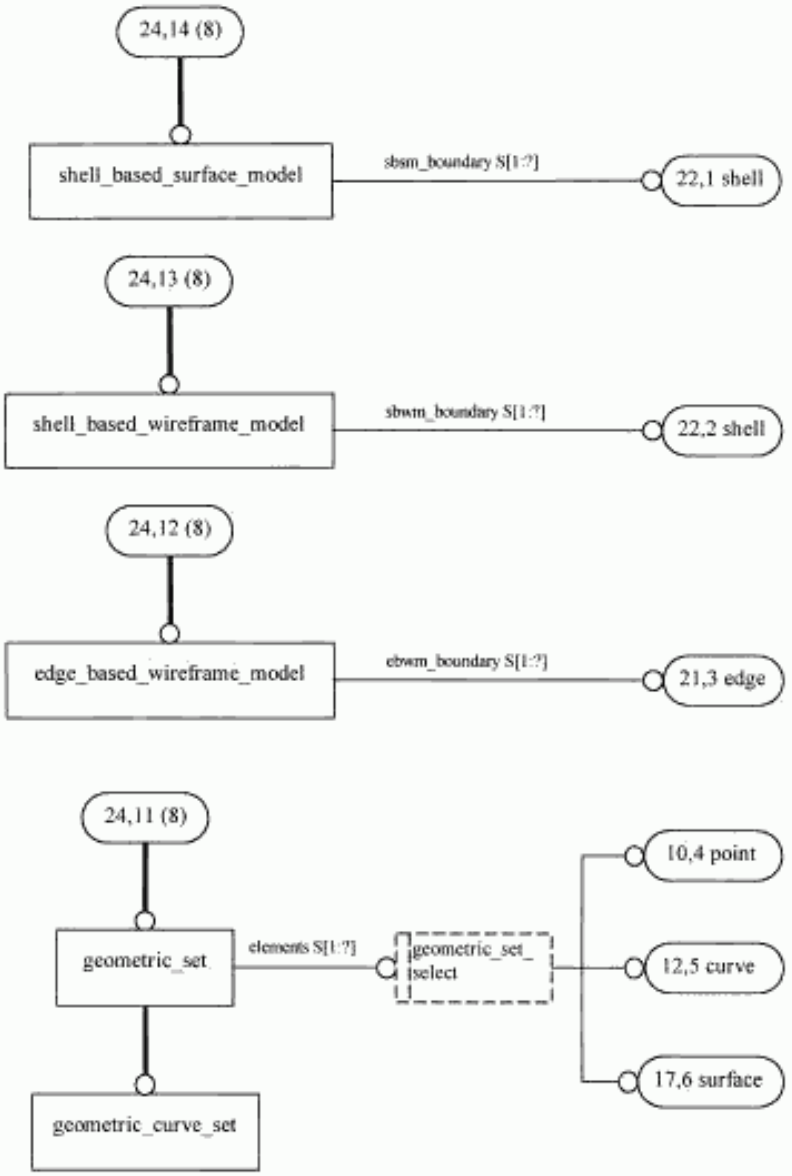


Рисунок H.24 — **surface_and_wireframe_models** — ПИМ EXPRESS-G диаграмма 24 из 39

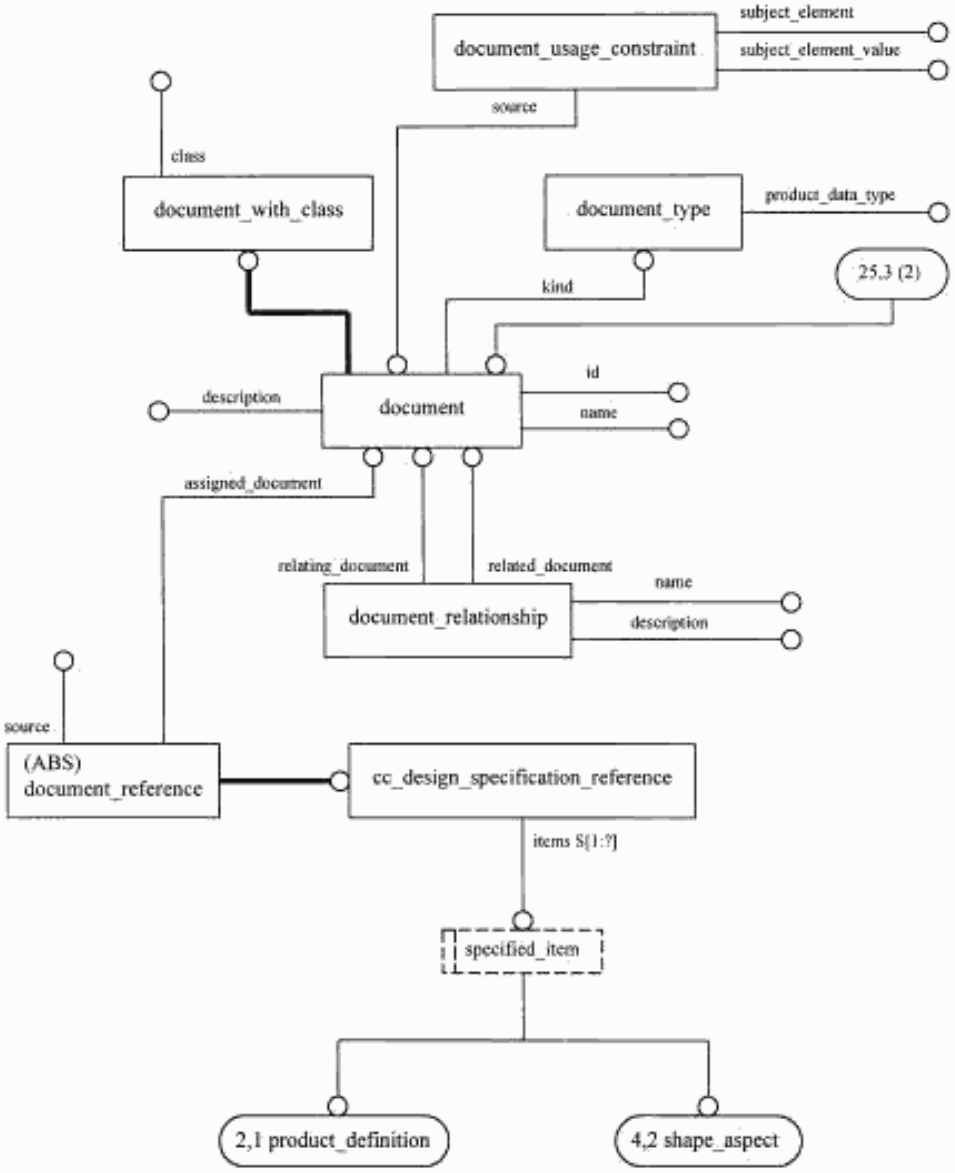


Рисунок H.25 — **document** — ПИМ EXPRESS-G диаграмма 25 из 39

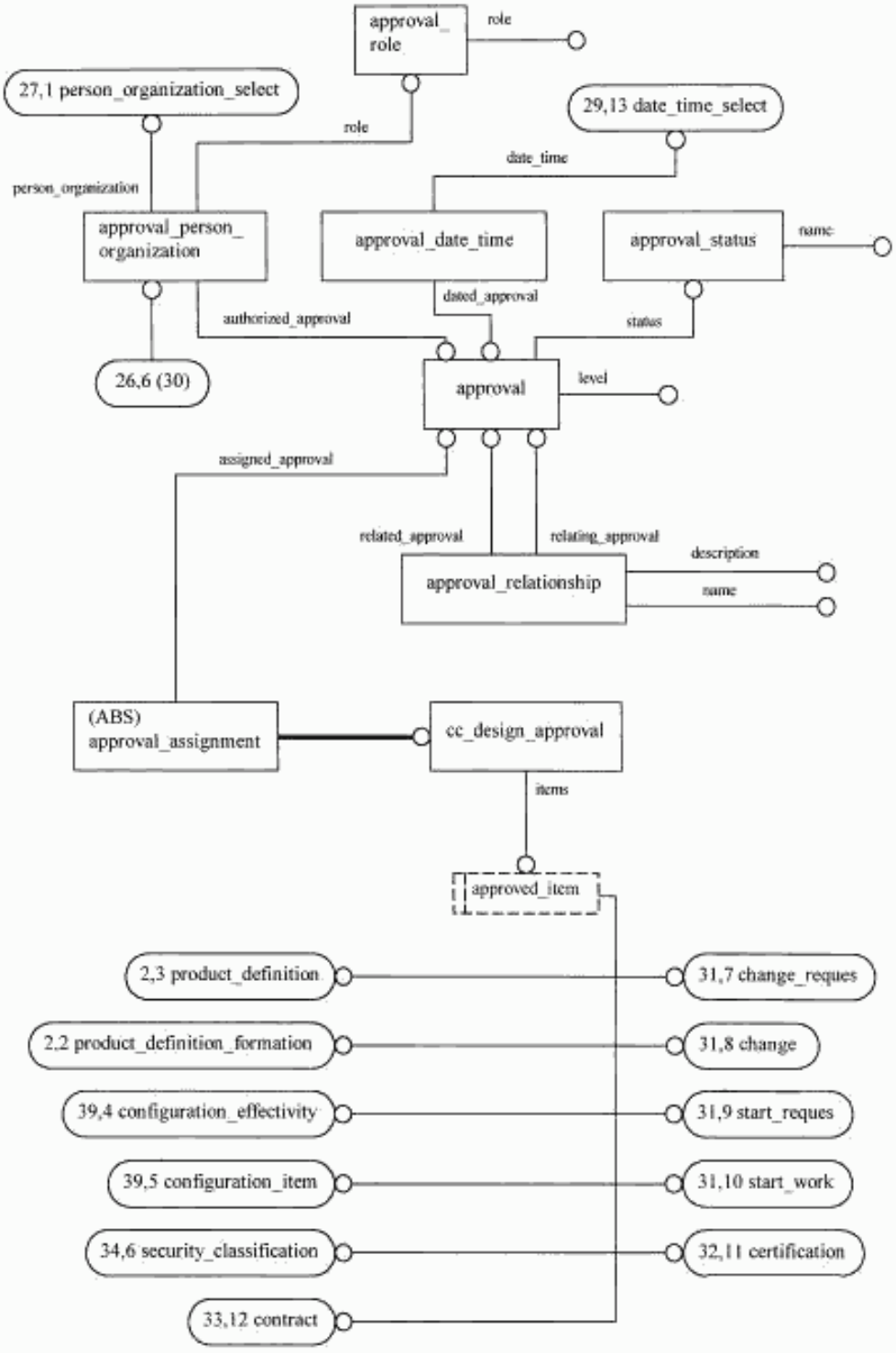


Рисунок Н.26 — **approval** — ПИМ EXPRESS-G диаграмма 26 из 39

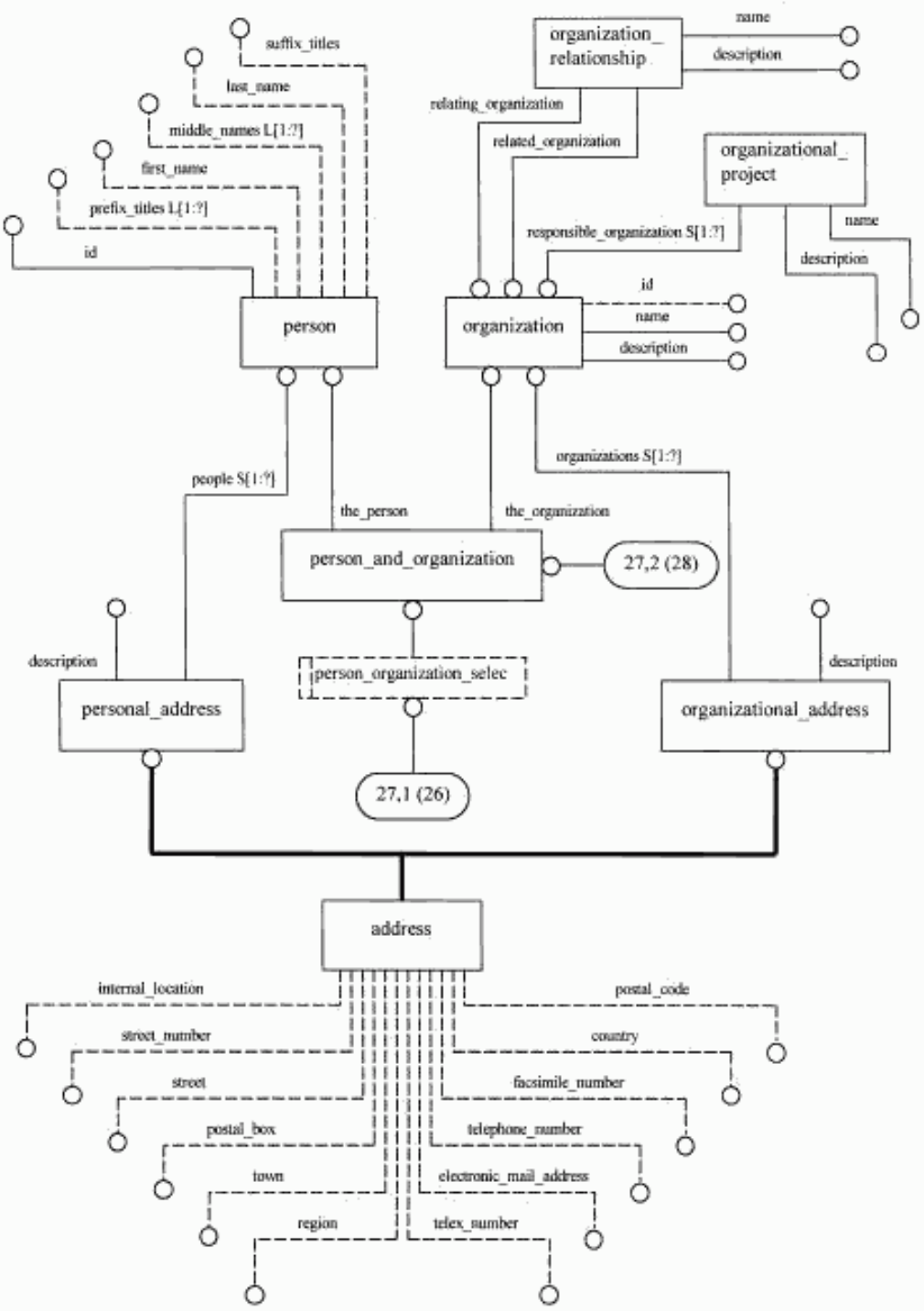


Рисунок H.27 — **person_and_organization** — ПИМ EXPRESS-G диаграмма 27 из 39

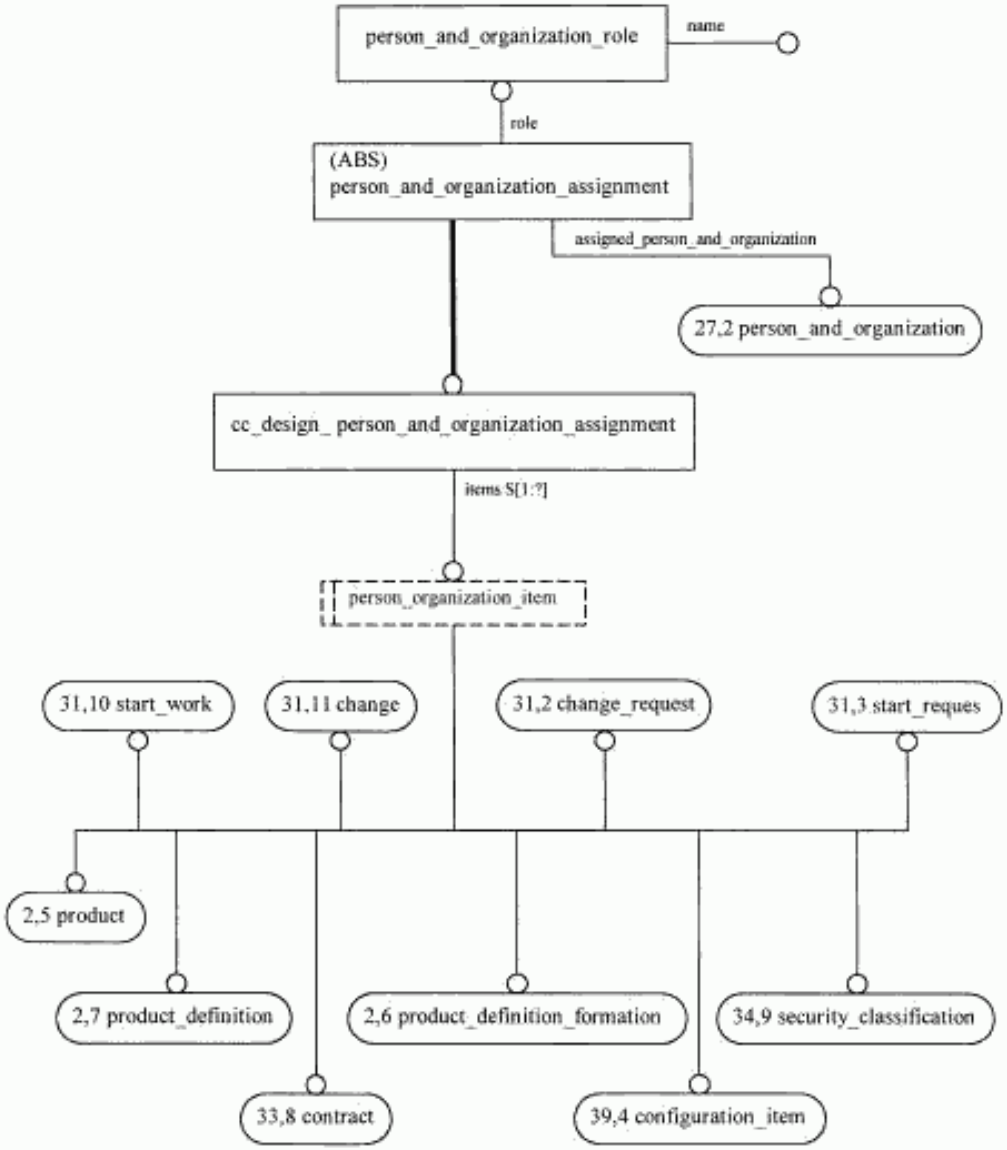


Рисунок H.28 — **person_and_organization_assignment** — ПИМ EXPRESS-G диаграмма 28 из 39

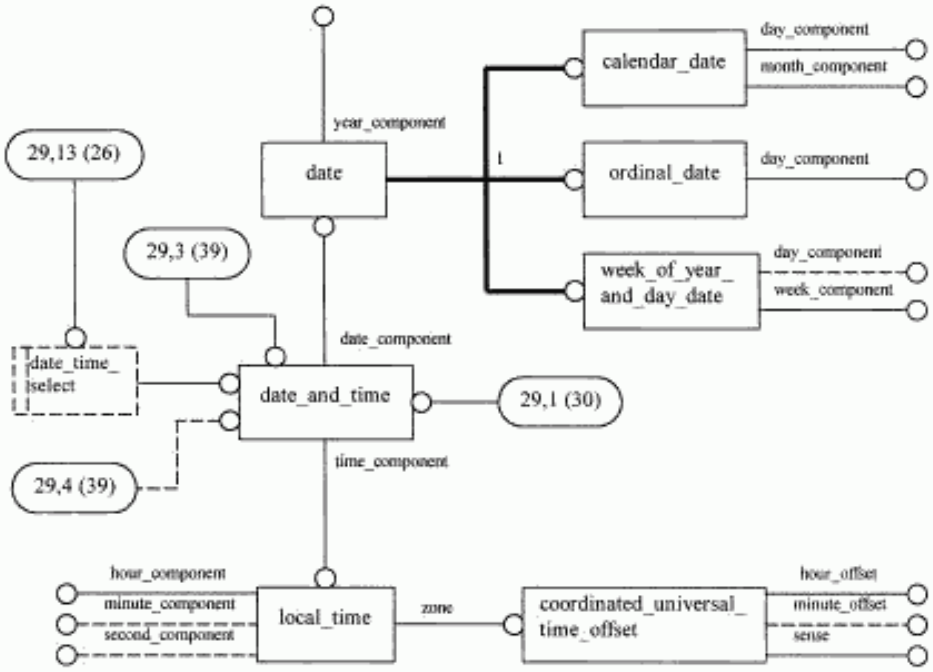


Рисунок H.29 — **date_and_time** — ПИМ EXPRESS-G диаграмма 29 из 39

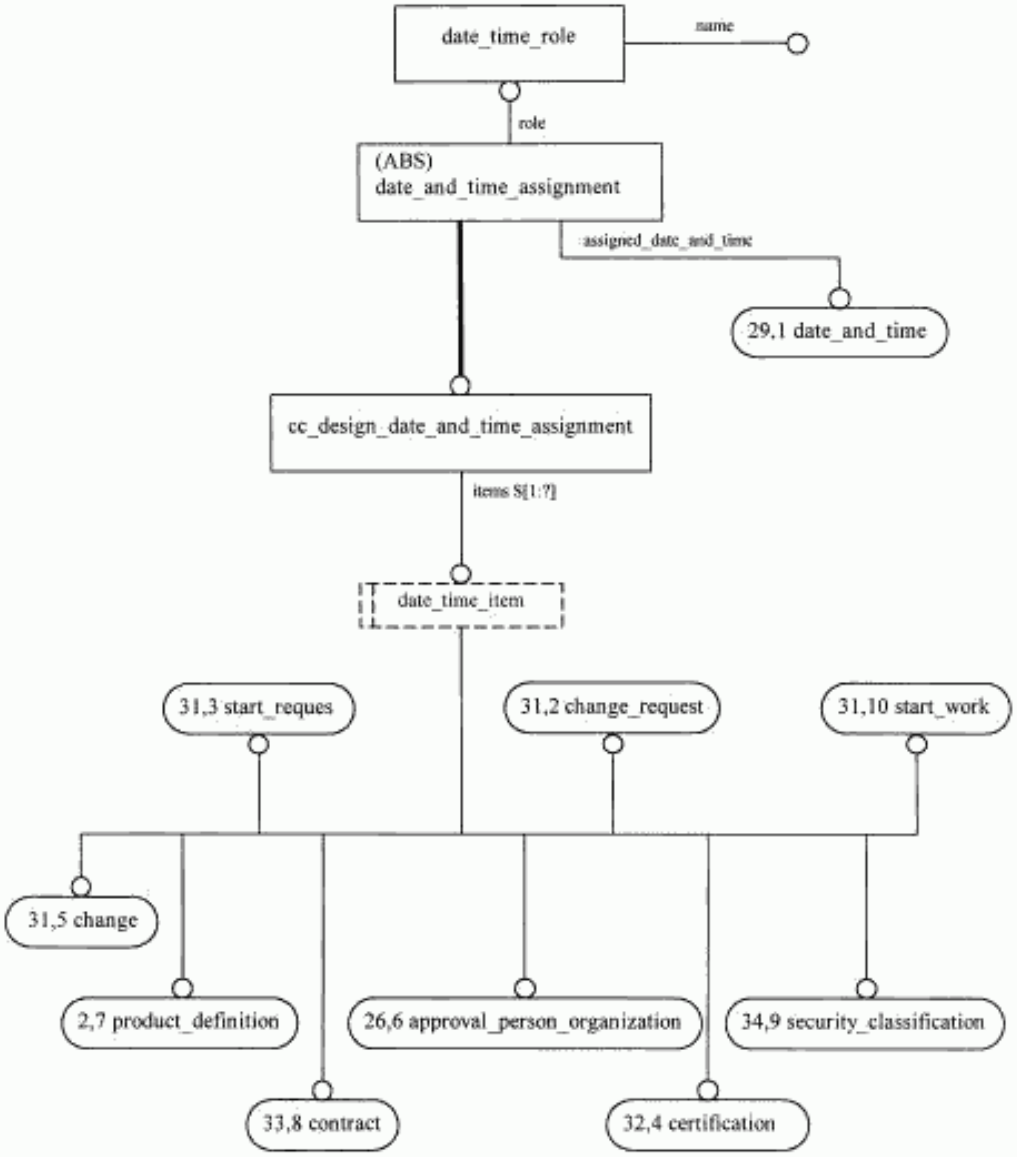


Рисунок Н.30 — **date_and_time_assignment** — ПИМ EXPRESS-G диаграмма 30 из 39

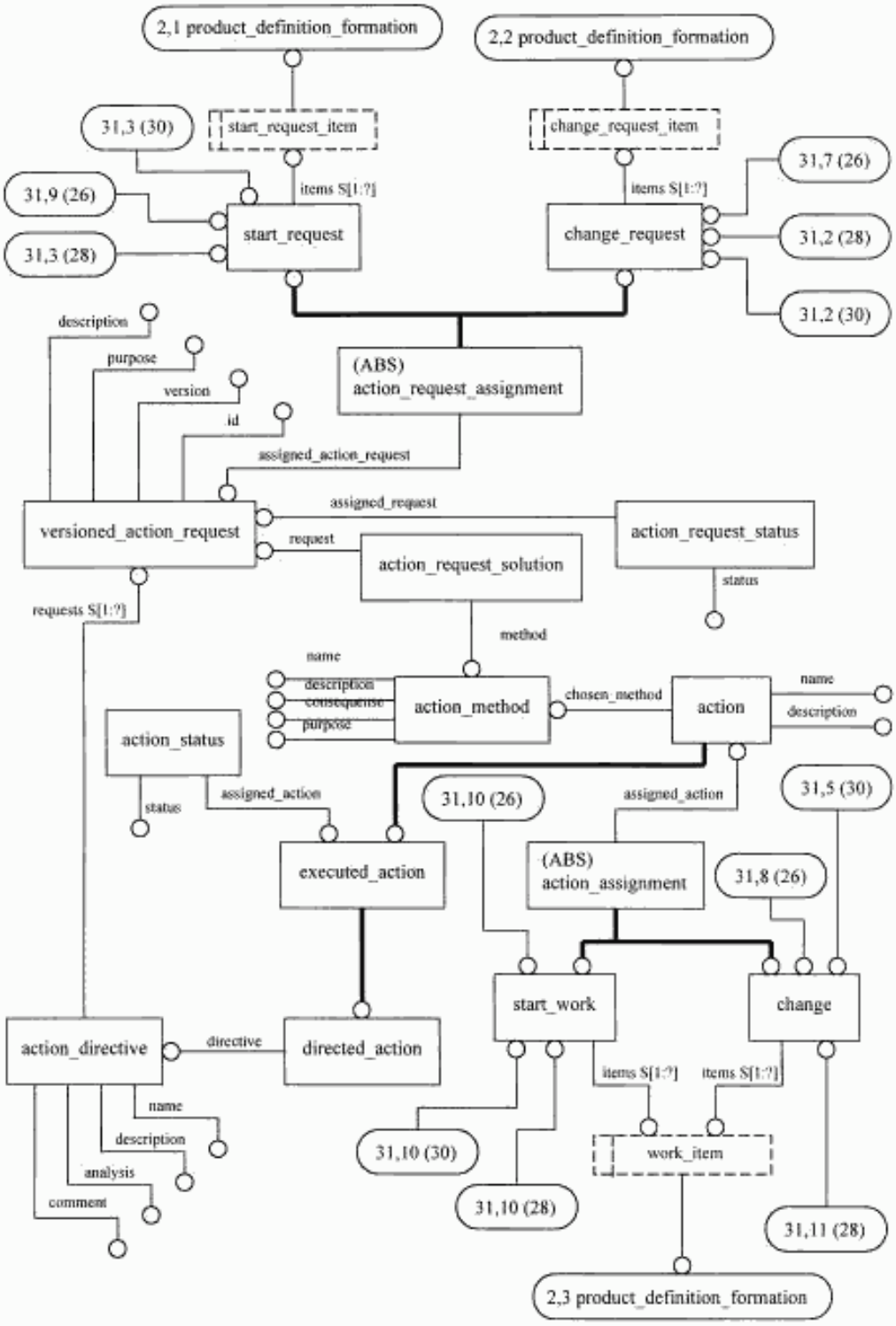


Рисунок H.31 — **work_and_change_documentation** — ПИМ EXPRESS-G диаграмма 31 из 39

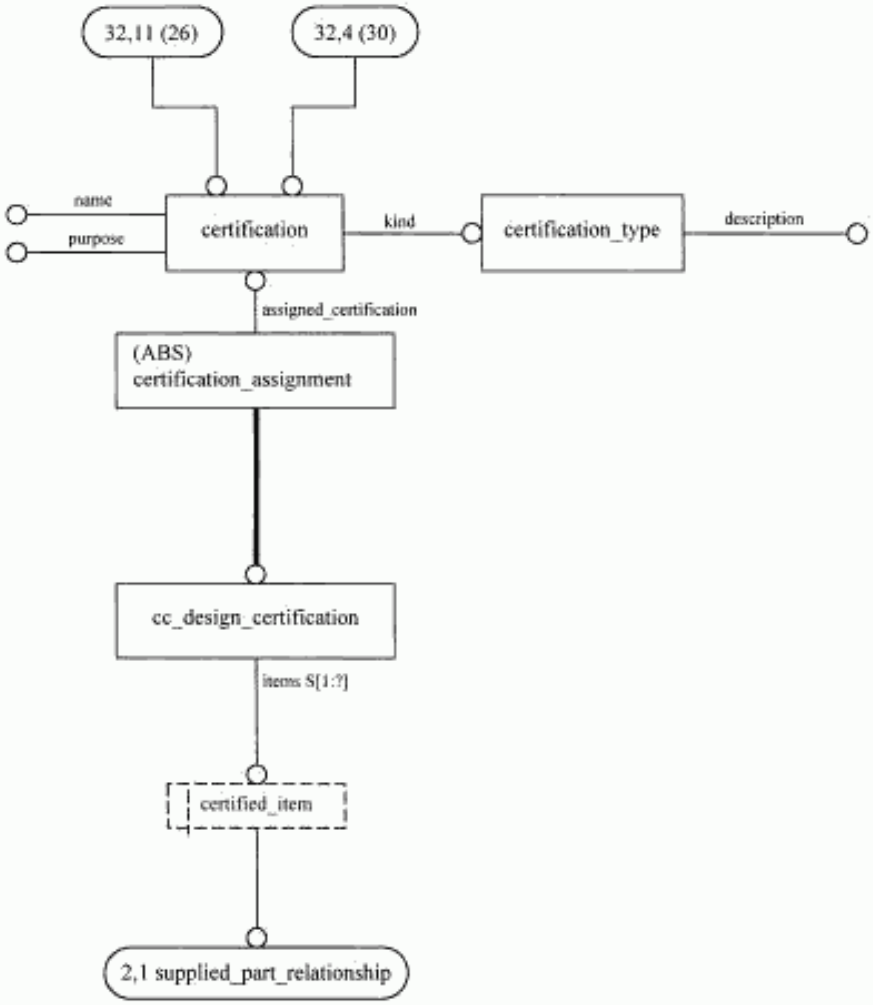


Рисунок H.32 — **certification** — ПИМ EXPRESS-G диаграмма 32 из 39

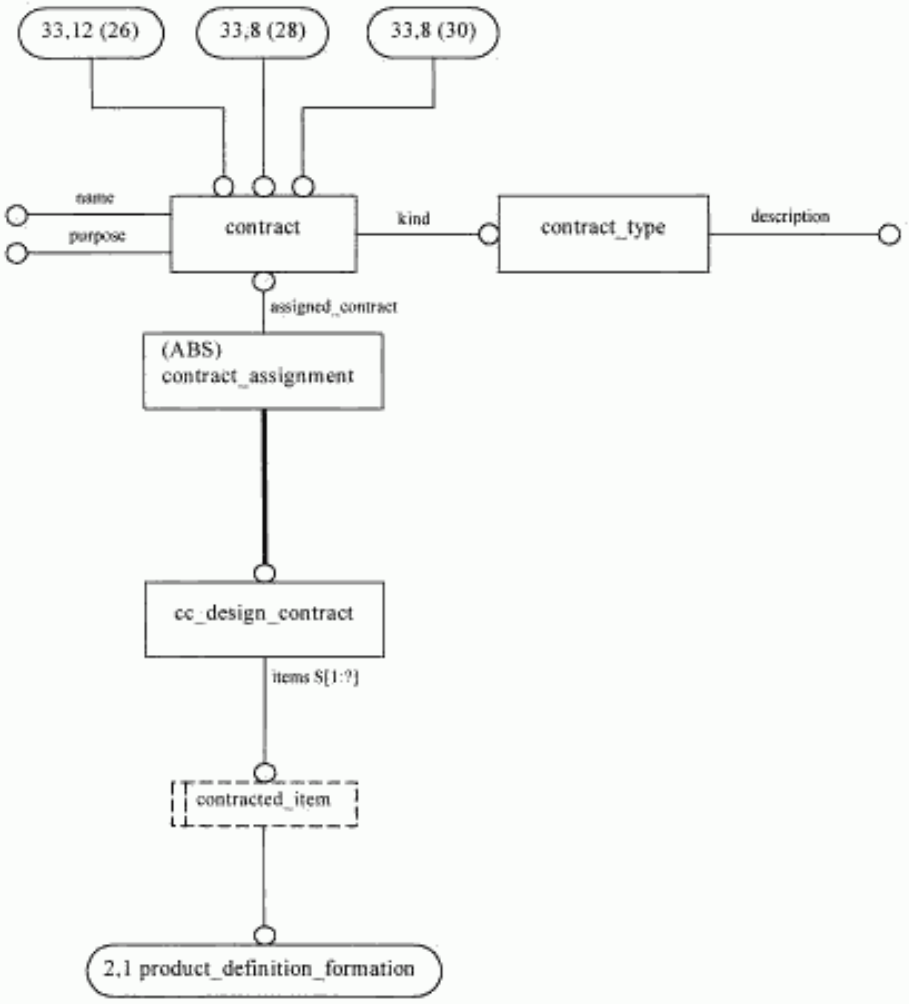


Рисунок H.33 — **contract** — ПИМ EXPRESS-G диаграмма 33 из 39

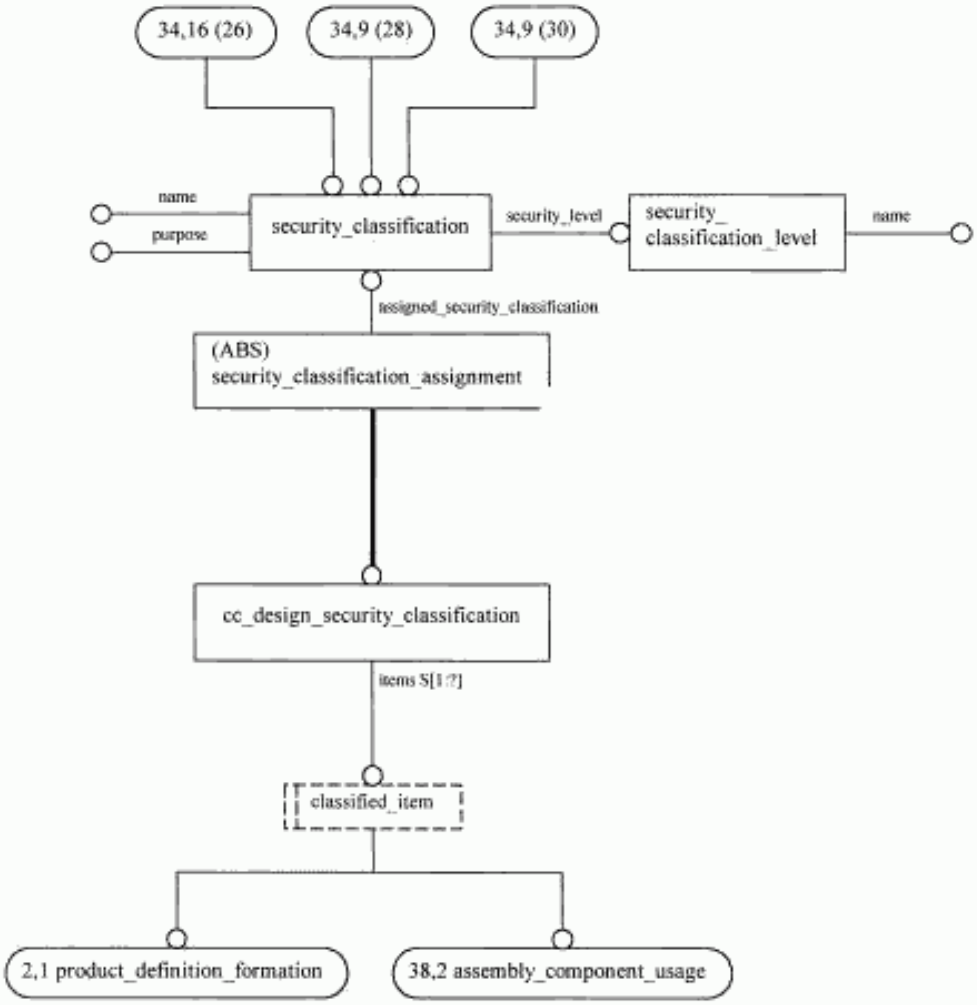


Рисунок H.34 — **security_classification** — ПИМ EXPRESS-G диаграмма 34 из 39

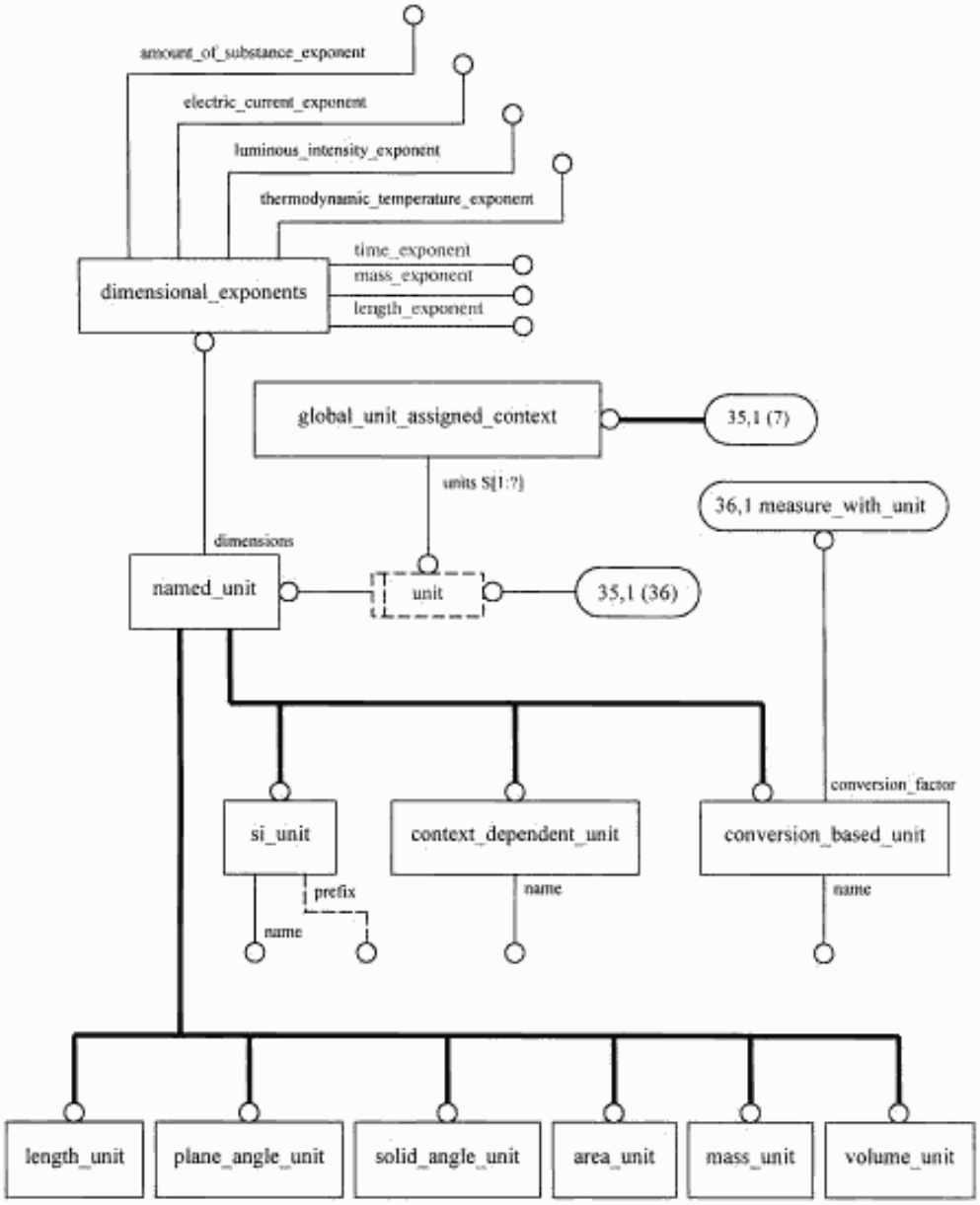


Рисунок H.35 — **units** — ПИМ EXPRESS-G диаграмма 35 из 39

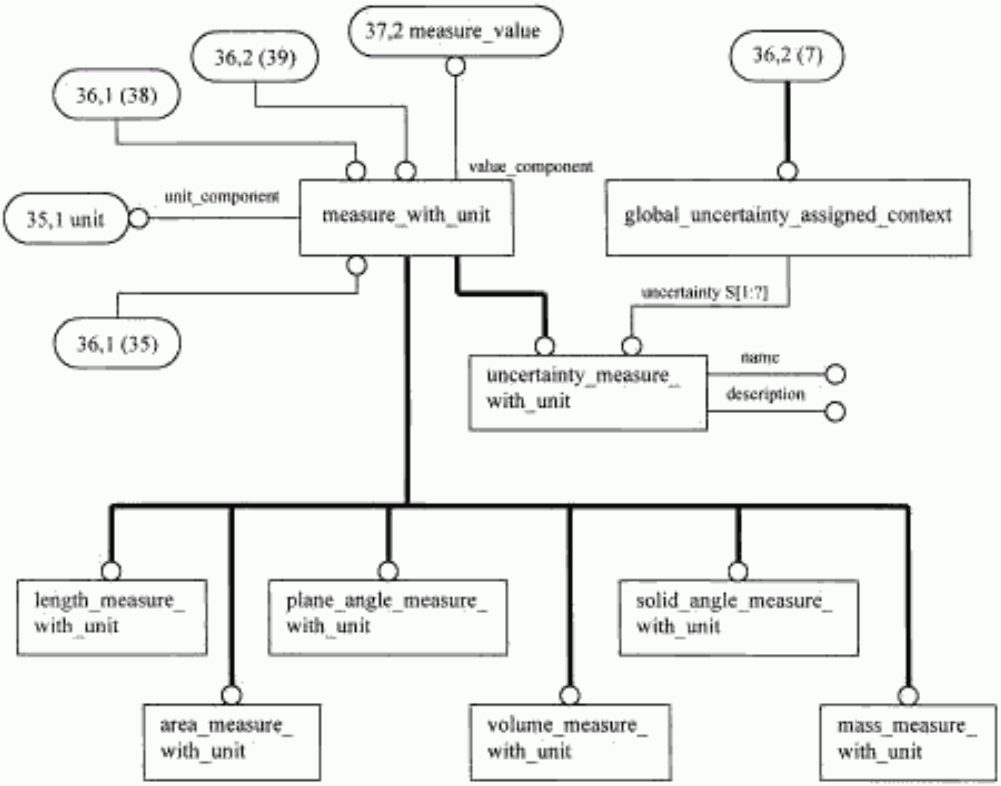


Рисунок H.36 — **measure_with_unit** — ПИМ EXPRESS-G диаграмма 36 из 39

Рисунок Н.37 — **measures** — ПИМ EXPRESS-G диаграмма 37 из 39

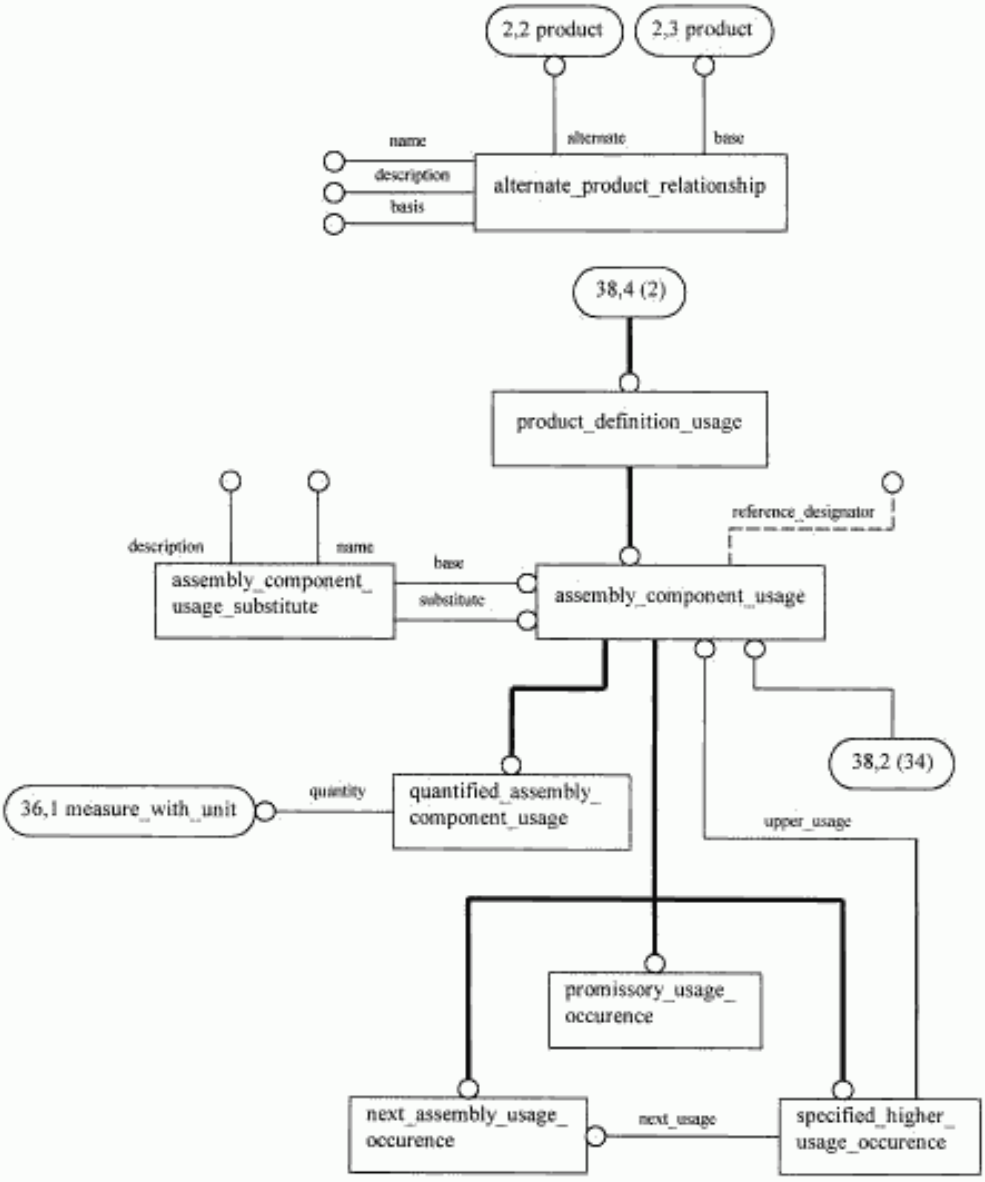


Рисунок H.38 — **product_structure** — ПИМ EXPRESS-G диаграмма 38 из 39

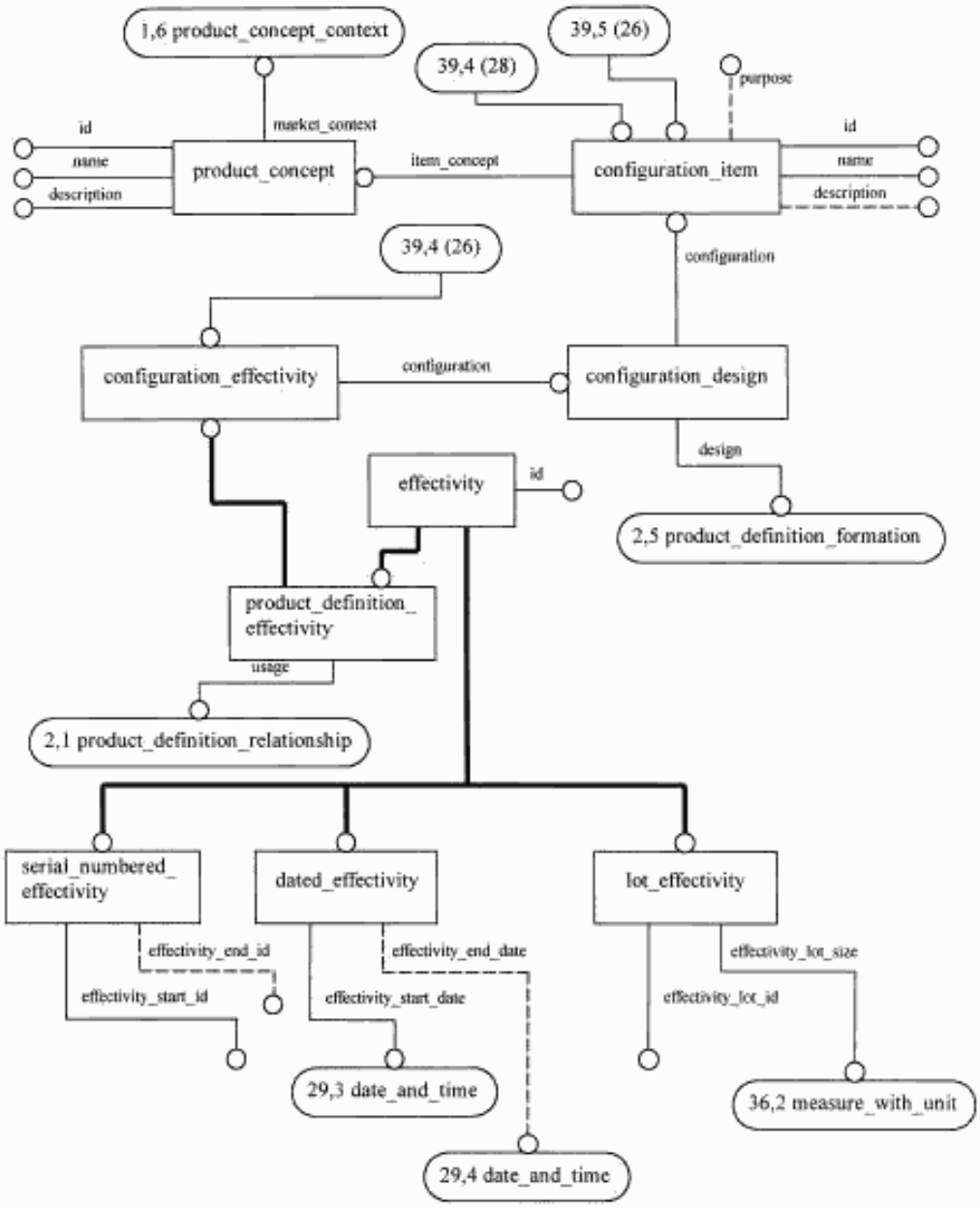


Рисунок H.39 — **configuration** — ПИМ EXPRESS-G диаграмма 39 из 39

ПРИЛОЖЕНИЕ J
(справочное)

Машинно-интерпретируемые листинги

В настоящем приложении приведен полный листинг EXPRESS-схемы, описанной в приложении А, без комментариев и пояснений. В приложении также приведен листинг полных и сокращенных имен объектов, указанных в приложении В. Содержание данного приложения в машинно-интерпретируемом виде может быть получено на следующих сайтах Интернета:

EXPRESS-схема: http://www.mel.nist.gov/step/parts/part203/is/tc2/

Сокращенные имена: http://www.mel.nist.gov/div826/subject/apde/snr/

При невозможности доступа к указанным сайтам можно обратиться в центральный секретариат ИСО или непосредственно в секретариат ИСО ТК 184/ПК4 по электронной почте (E-mail): sc4sec@cme.nist.gov.

Примечание — Информация, представленная в машинно-интерпретированном виде, носит справочный характер. Обязательным является текст настоящего стандарта.

ПРИЛОЖЕНИЕ К
(справочное)

Руководство по применению прикладного протокола

К.1 Цели прикладного тестирования

В настоящем разделе описаны цели прикладного (функционального) тестирования. Основной целью тестирования является проверка функциональных возможностей реализации в части выдачи конкретных результатов по комплексным запросам, включающим ряд объектов и атрибутов, обеспечивающим выдачу соответствующих ответов на реальные вопросы.

К.1.1 Функциональная единица (ФЕ) Effectivity

Следующий список определяет цели прикладного тестирования для ФЕ Effectivity:

- задавая идентификатор объекта **product,** определить наименование и адрес лица (лиц) с ролью **configuration_manager**;
- задавая объект **product_concept**, определить элементы конфигурации, для которых объект **dated_effectivity** имеет некоторое значение, заданное атрибутом **end_date**;
- задавая идентификатор объекта **product_definition** для детали (компонента), определить количество данных деталей (компонентов), входящих в каждый объект **configuration_design**, заданный посредством **serial_numbered_effectivity**.

К.1.2 Функциональная единица (ФЕ) End_item_identification

Следующий список определяет цели прикладного тестирования для ФЕ End_item_identification:

- для заданного объекта **product_concept** определить, к каким связанным с ним объектам **configuration_item** относятся предложенные **change_request**;
- для заданного объекта **product_concept** определить, какие детали для него предпочтительнее закупить, а не изготавливать;
- для заданного объекта **configuration_item** определить, какие входящие в него компоненты, определенные объектами **product_definition,** имеют взаимозаменяемые (альтернативные) изделия.

К.1.3 Функциональная единица (ФЕ) Bill_of_material

Следующий список определяет цели прикладного тестирования для ФЕ Bill_of_material:

- задавая сборочную единицу посредством объекта **product_definition,** определить, какие входящие в нее компоненты являются стандартными изделиями;
- задавая идентификатор объекта **product**, определить наивысший уровень сборочной единицы, содержащий список деталей (компонентов), устанавливающий число конкретных деталей, входящих в сборочную единицу следующего уровня;
- задавая сборочную единицу посредством объекта **product_definition,** определить степень готовности (статус выпуска) каждой комплектующей детали;
- задавая сборочную единицу посредством объекта **product_definition**, определить уровень конфиденциальности, заданный для каждой комплектующей детали;
- задавая изделие посредством объекта **product_definition_formation**, определить относящиеся к нему контракты и соответствующие подрядные организации;
- задавая объект **product_definition,** определить версию объекта **product_definition_formation** для каждого заменяющего изделия и атрибут **reference_designator** для исходного и заменяющего объекта **product_definition.**

К.1.4 Функциональная единица (ФЕ) Part_identification

Следующий список определяет цели прикладного тестирования для ФЕ Part_identification:

- задавая объект **product_definition_formation,** определить его классификацию и допустимые **product_categories**;
- задавая объект **product_definition,** определить была ли данная деталь спроектирована на основе другой, и если да, то на основе какой;
- задавая объект **product_definition** для детали, определить уровень конфиденциальности каждого его применения в любой последующей сборочной единице.

К.1.5 Функциональная единица (ФЕ) Design_information

Следующий список определяет цели прикладного тестирования для ФЕ Design_information:

- задавая объект **product_definition** для детали, определить требования к финишной обработке ее поверхности и соответствующие ссылки на этот объект из некоторой заданной сборочной единицы;
- задавая объект **product_definition** для детали, определить код, класс и твердость соответствующего материала.

К.1.6 Функциональная единица (ФЕ) Source_control

Следующий список определяет цели прикладного тестирования для ФЕ Source_control:

- задавая деталь, определить может ли какая-либо ее версия, заданная объектом **product_definition_formation**, быть поставленной извне, и в положительном случае определить адрес поставщика каждой такой версии.

К.1.7 Функциональная единица (ФЕ) Design_activity_control

Следующий список определяет цели прикладного тестирования для ФЕ Design_activity_control:

- задавая для детали объект **change_request,** определить контракты, связанные с теми объектами **product_definition_formation**, к которым относится данное изменение;
- задавая для детали утвержденный объект **change**, определить объекты **configuration_item**, содержащие изменяемые версии детали, заданные объектами **product_definition_formation**;
- задавая объект **configuration_item,** определить, какие его комплектующие детали должны быть изменены в соответствии с объектом **change_request**;
- задавая объект **change_request**, определить лицо(а) с ролью **request_recipient** и статус запроса (**request_status**), а также лицо(а) с ролью **approver** и статус утверждения;
- задавая для детали утвержденный объект **change**, определить объекты **effectivity**, влияющие на применение измененной детали.

К.1.8 Функциональная единица (ФЕ) Shape

Следующий список определяет цели прикладного тестирования для ФЕ Shape:

- задавая для конкретной детали объект **product_definition,** определить геометрию и/или топологию, установленную в представлении его формы посредством объекта **shape_representation**;
- задавая для сборочной единицы объект **product_definition,** определить геометрию и/или топологию, связанную только с представлением ее формы посредством соответствующего объекта **shape_representation**;
- задавая для сборочной единицы объект **product_definition** и некоторое координатное пространство, описанное в **shape_representation,** определить все комплектующие детали данной сборочной единицы, находящиеся в этом пространстве;
- задавая объект **shape_representation,** определить все объекты **product_definition_formation**, в описаниях которых присутствует данный **shape_representation**;
- задавая для детали объект **product_definition,** определить технические требования, относящиеся к конкретным объектам **shape_aspect** или областям применения данной детали;
- определить все объекты **geometric_representation_item**, не используемые в представлении формы любого изделия (детали), заданного **product_definition** (отрицательный результат).

К.2 Пример детали

Ниже приведен пример представления детали в формате, определенном в ГОСТ Р ИСО 10303-21. Для полноты понимания данного примера в нем отсутствуют сокращенные наименования имен объектов на языке EXPRESS из приложения В, а атрибуты, заданные одной длинной строкой, разбиты на несколько коротких строк. Данный пример детали соответствует схеме ПИМ, приведенной в приложении А.

```
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=CARTESIAM_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( 'cpi7' , (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;
```

ПРИЛОЖЕНИЕ L
(справочное)

Библиография

[1] "Guidelines for 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 Avalloneand 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] Р 50.1.028-2001 Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования.

[5] "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.

Тематический указатель

Примечание — Данный указатель сформирован в соответствии с наименованием тематических объектов настоящего стандарта на английском языке.

УДК 656.072:681.3:006.354 ОКС 25.040.40 П87 ОКСТУ 4002

Ключевые слова: автоматизация, средства автоматизации, прикладные автоматизированные системы, промышленные изделия, данные, представление данных, обмен данными, протокол, проектирование

Редактор *В.П. Огурцов*
Технический редактор *О.Н. Власова*
Корректоры *М.В. Бучная, В.С. Черная*
Компьютерная верстка *Е.Н. Мартемьяновой*

Изд. лиц. № 02354 от 14.07.2000. Сдано в набор 12.11.2003. Подписано в печать 09.01.2004. Усл. печ. л. 35,34.
Уч.-изд. л. 36,20. Тираж 300 экз. С 112. Зак. 2375.

ИПК Издательство стандартов, 107076 Москва, Колодезный пер., 14.
http://www.standards.ru e-mail: info@standards.ru
Набрано в Издательстве на ПЭВМ.
Отпечатано в Калужской типографии стандартов, 248021 Калуга, ул. Московская, 256.
ПЛР № 040138