ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО 16100-3 – 2010

Системы промышленной автоматизации и интеграция

ПРОФИЛИРОВАНИЕ ВОЗМОЖНОСТИ ИНТЕРОПЕРАБЕЛЬНОСТИ ПРОМЫШЛЕННЫХ ПРОГРАММНЫХ СРЕДСТВ

Часть 3

Службы интерфейса, протоколы и шаблоны возможностей

ISO 16100-3:2005

Industrial automation systems and integration –

Manufacturing software capability profiling for interoperability –

Part 3: Interface services, protocols and capability templates

(IDT)

Издание официальное

Москва
Стандартинформ
2014

Предисловие

Цели и принципы стандартизации в Российской Федерации установлены Федеральным законом от 27 декабря 2002 г. №184-ФЗ «О техническом регулировании», а правила применения национальных стандартов Российской Федерации — ГОСТ Р 1.0—2004 «Стандартизация в Российской Федерации. Основные положения»

Сведения о стандарте

1 ПОДГОТОВЛЕН Научно-техническим центром «ИНТЕК» на основе собственного аутентичного перевода на русский язык стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 100 «Стратегический и инновационный менеджмент»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 21.12.2010 г. № 857-ст

4 Настоящий стандарт идентичен международному стандарту ИСО 16100-3:2005 «Системы промышленной автоматизации и интеграция. Профилирование возможности интероперабельности промышленных программных средств. Часть 3. Службы интерфейса, протоколы и шаблоны возможностей» (ISO 16100-3:2005 «Industrial automation systems and integration – Manufacturing software capability profiling for interoperability – Part 3: Interface services, protocols and capability templates»).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты Российской Федерации, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

Информация об изменениях к настоящему стандарту публикуется в ежегодно издаваемом информационном указателе «Национальные стандарты», а текст изменений и поправок — в ежемесячно издаваемых информационных указателях «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ежемесячно издаваемом информационном указателе «Национальные стандарты». Соответствующая информация, уведомления и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет

Введение

Разработка комплекса стандартов ИСО 16100 обусловлена необходимостью решения следующих проблем, связанных с:

a) постоянно увеличивающейся базой решений, зависящих от поставщиков;

b) трудностями, возникающими у пользователей при применении стандартов;

c) необходимостью перехода к модульным наборам инструментальных средств интеграции системы;

d) признанием того, что прикладное программное обеспечение и практический опыт его применения является интеллектуальным капиталом предприятия.

Комплекс стандартов ИСО 16100 определяет формат профиля возможностей программного обеспечения, интерпретируемого компьютером в электронно-цифровой форме и не вызывающего трудностей при его чтении человеком, а также устанавливает метод, отражающий основные возможности программного обеспечения на производстве в соответствии с ролями, определенными жизненным циклом производственного приложения, независимо от архитектуры определенной системы или платформы реализации.

Некоторые диаграммы, приведенные в комплексе стандартов ИСО 16100, построены на языке UML. Необходимо отметить, что не все концепции, представленные в этих диаграммах, имеют пояснения в тексте, поскольку стандарты рассчитаны на пользователей, имеющих определенное знание языка UML.

В ссылках на классы (объекты) и службы (сервисы) в настоящем стандарте приведены условные обозначения наименований, например:

ServiceAccessPoint — точечный объект доступа к сервису:

registerProfile — сервисный примитив (базисный элемент сервиса) для регистрации профиля.

Настоящий стандарт разработан Техническим комитетом ИСО/ТК 184 «Системы промышленной автоматизации и интеграция», Подкомитетом ПК 5 «Архитектура, коммуникации и структуры интеграции».

Комплекс стандартов ИСО 16100 имеет общее наименование «Системы промышленной автоматизации и интеграция. Профилирование возможности

интероперабельности промышленных программных средств» и включает в себя следующие части:

- часть 1 – Структура;

- часть 2 – Методология профилирования;

- часть 3 – Службы интерфейса, протоколы и шаблоны возможностей.

- часть 4 – Методы аттестационных испытаний, критерии и отчеты;

- часть 5 – Методология согласования конфигураций профилей с помощью многоцелевых структур классов.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы промышленной автоматизации и интеграция

ПРОФИЛИРОВАНИЕ ВОЗМОЖНОСТИ ИНТЕРОПЕРАБЕЛЬНОСТИ ПРОМЫШЛЕННЫХ ПРОГРАММНЫХ СРЕДСТВ

Часть 3

Службы интерфейса, протоколы и шаблоны возможностей

Industrial automation systems and integration.

Manufacturing software capability profiling for interoperability.

Part 3. Interface services, protocols and capability templates

Дата введения – 2011 – 09 – 01

1 Область применения

Настоящий стандарт устанавливает требования к службам интерфейса и протоколам, используемым с целью обеспечения доступа и редактирования профилей возможностей интероперабельности аппаратных и программных средств разных поставщиков, а также доступа к соответствующим шаблонам, используемым для метода профилирования возможностей, определенного в разделе 5 ИСО 16100-2.

Настоящий стандарт устанавливает подробные описания сервисов, используемых для доступа к профилям возможностей и выполнения процесса согласования этих профилей.

Настоящий стандарт применяют с целью определения интероперабельности единиц программного обеспечения разных поставщиков, используемых в производственном домене.

Настоящий стандарт не распространяется на взаимозаменяемость единиц программного обеспечения.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты, которые необходимо учитывать при использовании настоящего стандарта. В случае ссылок на документы, у которых указана дата утверждения, необходимо пользоваться только указанной редакцией. В случае, когда дата

утверждения не приведена, следует пользоваться последней редакцией ссылочных документов, включая любые поправки и изменения к ним:

ИСО 16100-1:2002 Системы промышленной автоматизации и интеграция. Профилирование возможности интероперабельности промышленных программных средств. Часть 1. Структура (ISO 16100-1:2002, Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 1: Framework)

ИСО 16100-2:2003 Системы промышленной автоматизации и интеграция. Профилирование возможности интероперабельности промышленных программных средств. Часть 2. Методология профилирования (ISO 16100-2:2003, Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 2: Profiling methodology)

IEEE 1320.1—1998 Стандарт языка функционального моделирования. Синтаксис и семантика для IDEFO (IEEE 1320.1-1998, Standard for Functional Modeling Language — Syntax and Semantics for IDEF0)

OMG ad/2003-04—01 Унифицированный язык моделирования. Суперструктура, версия v2.0 (OMG ad/2003-04-01, Unified Modeling Language; Superstructure v2.0)

REC-xml-1-19980210 Рекомендация W3C 1.0 по языку XML (REC-xml-19980210, Extensible Markup Language (XML) 1.0 W3C Recommendation)

REC-scap12-20021219 Версия 1.2 SOAP. Часть 1. Структура обмена сообщениями (REC-soap12-20021219, SOAP Version 1.2 — Part 1: Messaging Framework)

REC-xmlschema-1-20010502 Схема языка XML. Часть 1. Структуры (REC-xmlschema-1-20010502 XML Schema Part 1: Structures)

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1 Термины, установленные в настоящем стандарте

3.1.1 **интерфейс профиля возможности** (capability profile interface): Функциональная, независимая от реализации точка доступа к сервису, в которой установлена совокупность сервисов, изложенных в 5.4 для работы с профилями возможностей.

Примечание – В некоторых случаях согласно ИСО 16100-2 интерфейс профиля возможности может быть реализован в качестве сервера баз данных.

3.1.2 **провайдер интерфейса профиля возможности** (capability profile interface provider): Программное обеспечение, реализующее интерфейс профиля возможности.

3.1.3 **кластер** (cluster): Совокупность единиц производственных ресурсов.

3.1.4 **компонент** (component): Часть единицы производственного программного обеспечения, включающая в себя компоненты программного обеспечения производства.

3.1.5 **потребитель** (consumer profile): Пользователь профиля или результат уровня совпадения.

3.1.6 **механизм обнаружения совпадений** (matcher): Метод сравнения предложенного профиля с необходимым профилем возможности интероперабельности.

3.1.7 **уровень совпадения** (matching level): Качественное измерение, определяющее уровень соответствия профиля возможности производственной единицы программного обеспечения функциональным требованиям производственного программного обеспечения.

3.1.8 **возможность интероперабельности единицы программного обеспечения производства** (MSU interoperability): Способность производственной единицы программного обеспечения поддерживать частное применение спецификации интерфейса при обмене наборами прикладной информации с другой производственной единицей программного обеспечения.

3.1.9 **взаимозаменяемость единицы программного обеспечения производства** (MSU interchangeability): Возможность замены одной единицы производственного программного обеспечения на другую единицу при обеспечении выполнения требуемой функции в рамках определенной производственной деятельности.

3.1.10 **производитель** (producer): Создатель профиля или результат уровня согласования для потребления.

3.1.11 **эталонная структура класса возможности** (reference capability class structure): Схема, представляющая иерархию классов возможностей, используемая для профилирования возможности.

3.1.12 **справочный словарь** (reference dictionary): Перечень классов возможностей, содержащий перечень ссылок на классы возможностей.

3.1.13 **схема** (schema): Определение метаданных на языке XML.

3.1.14 **шаблон** (template): Схема профиля возможности производственного программного обеспечения.

3.1.15 **обнаружитель совпадений типа I** (type I matcher): Обнаружитель совпадений, обрабатывающий профили, полученные из одной структуры классов возможностей.

3.1.16 **обнаружитель совпадений типа II** (type II matcher): Обнаружитель совпадений, обрабатывающий профили, полученные из той же самой или другой структуры классов возможностей.

3.2 Термины по ИСО 16100-1

В настоящем стандарте применены следующие термины, определенные в ИСО 16100-1. Ссылка на терминологическую статью приведена в скобках после определения.

3.2.1 **возможность** (capability): Совокупность функций и сервисов программного обеспечения, а также набор критериев для оценки качества функционирования поставщика возможностей.

[статья 3.3]

3.2.2 **профилирование возможности** (capability profiling): Выбор набора предложенных сервисов, определенных особым интерфейсом в рамках структуры возможности интероперабельности программных изделий разных поставщиков.

[статья 3.4]

3.2.3 **производственное программное обеспечение** (manufacturing software): Тип ресурса программного обеспечения в рамках автоматической системы, который имеет значение для производства (например, CAD/PDM) за счет

интеграции данных в работу потока управления и передачи информации между компонентами автоматической системы, вовлеченными в производственный процесс, и другими ресурсами предприятия, а также между предприятиями в цепочке снабжения или спроса.

[статья 3.10]

3.2.4 **возможность производственного программного обеспечения** (manufacturing software capability): Совокупность функций и сервисов по сравнению с критериями оценки функционирования при заданном наборе производственных условий.

[статья 3.14]

3.2.5 **профиль возможности производственного программного обеспечения** (manufacturing software capability profile): Краткое представление возможности производственного программного обеспечения соответствовать требованиям применения на производстве.

[статья 3.15]

3.2.6 **компонент производственного программного обеспечения** (manufacturing software component): Класс ресурса производственного программного обеспечения, предназначенного для поддерживания выполнения частной производственной задачи.

[статья 3.11]

3.2.7 **единица производственного программного обеспечения** (manufacturing software unit): Класс ресурса программного обеспечения, состоящего из одного или более компонентов производственного программного обеспечения, выполняющего определенную функцию в рамках производственной деятельности, одновременным поддерживанием механизма обмена общей информацией с другими единицами.

[статья 3.12]

3.3 Термины по ИСО 16100-2

3.3.1 **класс возможности** (capability class): Элемент метода профилирования возможности, представляющий функциональность и поведение единицы

программного обеспечения в отношении программного обеспечения для производственной деятельности.

[статья 3.3]

3.3.2 **интеграция профиля возможности** (capability profile integration): Процесс, в котором две или более единицы программного обеспечения взаимодействуют с помощью эквивалентных интерфейсов, конфигурируемых в совместимом виде, на что указывают их профили возможностей.

[статья 3.4]

3.3.3 **интерфейс** (interface): Абстракция поведения объекта, состоящего из подмножества взаимодействий этого объекта с учетом накладываемых ограничений при их возможном появлении.

[статья 3.8]

3.3.4 **профиль** (profile): Совокупность одной или более основных спецификаций и/или подпрофилей, и, в приемлемых случаях, идентификация выбранных классов, согласующихся подмножеств, опций и параметров основных спецификаций или подпрофилей, необходимых для выполнения конкретной функции, деятельности или взаимосвязи.

[статья 3.10]

4 Сокращения

CPI – интерфейс профиля возможности (capability profile interface);

DTD – определение типа документа (document type definition);

ML – уровень совпадения (matching level);

MSU – единица производственного программного обеспечения (manufacturing software unit);

UML – унифицированный язык моделирования (unified modeling language);

XML – расширяемый язык гипертекстовой разметки (extensible markup language).

5 Модель производственного программного обеспечения и профиль

5.1 Производственная деятельность и модель информационного обмена

Производственное приложение должно представлять собой ряд производственных процессов, которые разрешаются, управляются и автоматизируются совокупностью производственных ресурсов путем обмена информацией с одновременной передачей материалов и энергии согласно ИСО 16100-1, рисунок 4.

Производственный процесс должен представлять собой последовательность запланированных производственных действий, в которых каждая производственная деятельность ассоциируется с набором производственных функций (см. подраздел 5.3 ИСО 16100-1 и приложение С настоящего стандарта).

Чтобы соответствовать требованиям производственного приложения, совокупность единиц производственного программного обеспечения (далее – ППО) должна быть распределена последовательно по графику, обеспечивающему выполнение всей совокупности производственных функций всех производственных действий, связанных с рядом производственных процессов, составляющих производственное приложение.

С помощью возможности интероперабельности программного обеспечения производства (см. раздел 6 ИСО 16100-1) каждая производственная деятельность должна быть ассоциирована с набором единиц ППО (далее – ЕППО). Согласно приложению А ИСО 16100-1 комплексная производственная деятельность может быть представлена в виде комбинации набора простых производственных действий, каждое их которых должно соответствовать одной функции и должно быть задействовано одной ЕППО.

Каждая производственная функция может быть выполнена с помощью набора ЕППО. Совокупность производственных функций может быть завершена с помощью одной ЕППО (см. 6.2 ИСО 16100-1).

Пример – Простое производственное приложение (например, взять и положить на место) может быть представлено в виде трех производственных процессов (например, загрузить изделие, переместить изделие, выгрузить изделие). Каждый производственный

процесс может быть ассоциирован с единичной деятельностью, формирующей последовательность функций из следующего набора: определить месторасположение, идентифицировать изделие, определить место, двигаться к изделию, взять изделие, определить место, положить изделие на место, уведомить координатора процесса. В одном случае два класса ЕППО (загрузка/выгрузка, перемещение) с двумя шаблонами возможностей могут быть профилированы в три экземпляра ЕППО по (одному экземпляру для каждого действия). В другом случае могут быть представлены четыре действия нижнего уровня или четыре класса ЕППО (определить месторасположение, идентифицировать/уведомить, взять/отпустить, двигаться к цели) и девять экземпляров ЕППО.

Согласно рисунку 1 ЕППО предоставляет несколько интерфейсов возможностей, включая свой профиль возможности. Доступ к профилю возможности может быть обеспечен с помощью интерфейса профиля возможности (CPI). Информацию о других интерфейсах включают в профиль возможности и, следовательно, информация доступна через интерфейс профиля возможности (CPI).

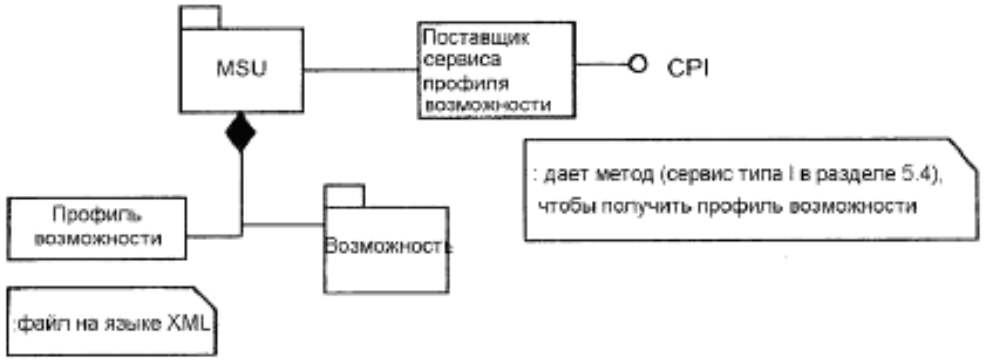

Рисунок 1 – Возможности ЕППО и соответствующие интерфейсы возможности

5.2 Единица производственного программного обеспечения

ЕППО должна быть смоделирована как тип производственного ресурса, соответствующий критериям возможности интероперабельности (программных и аппаратных средств разных поставщиков), установленных с помощью необходимой последовательности и требуемого согласования по времени специфической совокупности производственных функций, реализуемых с помощью ЕППО и информационных обменов, которые ЕППО должна поддерживать.

Согласно рисунку 4 ИСО 16100-1 ЕППО должна быть ассоциирована с производственной деятельностью и соответствующей возможностью. Компонент программного обеспечения не должен быть связан с профилем возможности.

Производственную функцию, ассоциированную с классом возможности производственной деятельности, представляют в виде функции, выполняемой одной или большим числом единиц ЕППО.

Пример – Производственная функция, ассоциированная с производственной деятельностью N (см. рисунок 2), приводится в действие с помощью ЕППО 3. С другой стороны, производственная функция, ассоциированная с производственной деятельностью М, приводится в действие с помощью ЕППО 1 и ЕППО 2.

Классы производственных возможностей, поддерживаемые набором ЕППО, должны быть установлены функцией производственной деятельности и связанными информационными обменами других производственных ресурсов, используемых для активации производственного процесса.

Частный класс ЕППО может применяться в разных деятельностях. Каждая ЕППО должна предоставлять набор интерфейсов. Критерии интероперабельности ЕППО должны быть установлены в соответствии с требованиями функционально совместимых действий. Настоящий стандарт не распространяется на критерии интероперабельности производственных процессов, а также на критерии интероперабельности групп ЕППО, ассоциированные с производственными процессами.

Требования ППО на каждом уровне могут быть представлены в виде совокупности классов возможностей, организованных в структуру, приведенную на рисунке В.1.

Примечание 1 – Производственный процесс (см. ИСО 16100-1, рисунок 4) состоит из множества производственных действий, имеющих вложенную или иерархическую структуру. Интероперабельность ЕППО может применяться только к последним из множества производственных действий.

Если две или более ЕППО обеспечивают необходимую производственную программную функцию в пределах производственный действий, то ЕППО должны соответствовать требованиям к совместной работы. Интерфейс, необходимый для

совместной работы набора ЕППО в пределах определенной деятельности, должен быть указан в профиле возможности программного обеспечения этой деятельности.

Примечание 2 – Интерфейс А (см. рисунок 2), предоставленный с помощью ЕППО 1 от поставщика А, взаимодействует с интерфейсом В, предоставленным с помощью ЕППО 2 от поставщика В. Критерии интероперабельности этих программных средств разных поставщиков обозначают возможностью интероперабельности I на основе требований деятельности М. Профиль возможности интерфейса А должен совпадать с профилем возможности интерфейса В для соответствия интероперабельности I. Данный профиль может отличаться для других производственных действий.

Примечание 3 – При необходимости взаимодействия двух видов деятельности, например М и N (см. рисунок 2), может быть использован набор критериев интероперабельности J, который базируется на общих требованиях деятельности М и N. Набор ЕППО, соответствующий данным видам деятельности, имеет профили возможностей, поддерживающие возможность интероперабельности J.

Совместное использование множества ЕППО должно быть эквивалентно ситуации, в которой требования к ППО конкретной деятельности обеспечивается одной ЕППО. Такое комбинированное поведение одной эквивалентной ЕППО зависит от совместимости использования спецификации интерфейса, общей для множества ЕППО. И наоборот, ЕППО может быть составлена из множества различных ЕППО с целью разделения деятельности на множество действий.

В случае, если ЕППО состоит из набора компонентов ППО, то ППО не должно включать в себя другие ЕППО. Компоненты ЕППО должны принадлежать только этой ЕППО. Настоящий стандарт не распространяется на информационный обмен и связи между интерфейсами определенных компонентов в рамках ЕППО.

Примечание 4 – Согласно рисунку 2 ЕППО 2 от поставщика С можно заменить ЕППО 2 от поставщика В для обеспечения производственной функции, необходимой в производственной деятельности М. Хотя интерфейс А, предоставленный ЕППО 1 от поставщика А, взаимодействует с интерфейсом С, предоставленным ЕППО 2 от поставщика С, полная взаимозаменяемость обоих ЕППО 2 не может быть реализована. Профиль возможности интерфейса В совпадает с профилем возможности интерфейса С для обеспечения поддержки функциональной совместимости, а не возможности интероперабельности программных средств разных поставщиков.

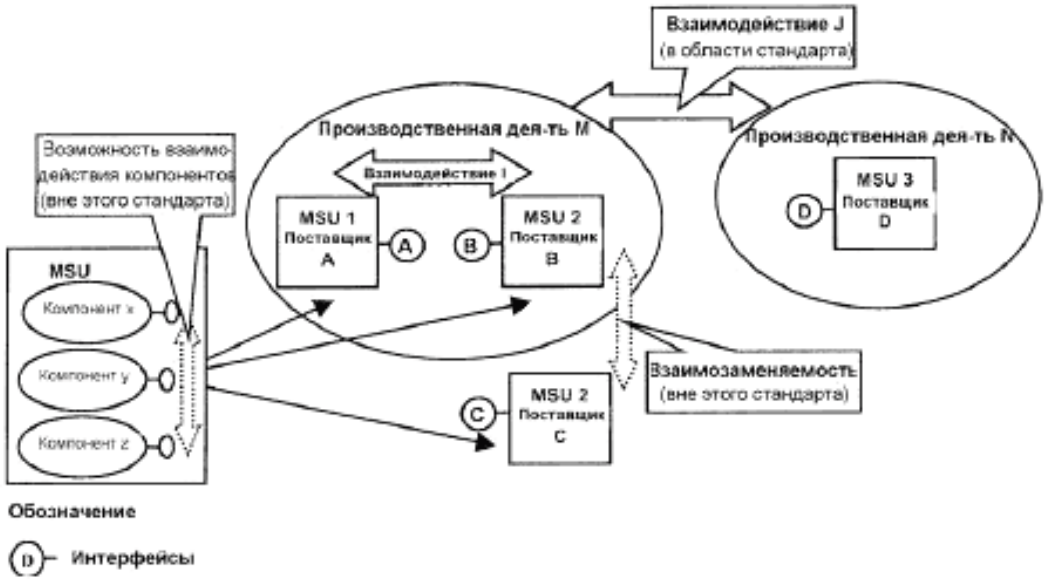

Рисунок 2 – ЕППО в рамках производственной деятельности

5.3 Совпадение профилей возможностей

5.3.1 Общие положения

Структура шаблона возможности ЕППО должна быть определена на основе структуры класса производственной возможности (см. ИСО 16100-2, подраздел 6.3). Профили возможностей в отношении требований возможностей производственной деятельности должны соответствовать ИСО 16100-2, подраздел 6.3. Профили возможностей являются шаблонами возможностей, как минимум, с конкретизированным профилированным именем единицы ПО; другие пункты шаблона возможности должны быть заполнены в соответствии с определенным уровнем согласования (см. ИСО 16100-2, подраздел 6.4).

Атрибут класса производственной возможности определяют по ИСО 16100-2, пункт 6.2.1, а концептуальную структуру – по ИСО 16100-2, пункты 6.2.1 и 6.2.4.

Примечание 1 – Взаимосвязь производственных приложений, производственной деятельности и производственных ресурсов приведена в ИСО 16100-1, рисунок 4; взаимосвязь классов возможностей с производственной деятельностью – в ИСО 16100-2, подраздел 6.2.

Примечание 2 – Эталонную структуру класса возможности представляют в виде схемы согласно приложению А.

Совпадающий профиль возможности интероперабельности ЕППО должен представлять следующие необходимые аспекты производственной деятельности:

a) обработку информации ввода – вывода, связанную с требуемой производственной функцией.

Примечание 3 – Элементы блока шаблона, представляющего собой специфическую часть профиля возможности, могут быть отнесены к категории элементов ввода или вывода производственной функции, связанной с деятельностью (см. ИСО 16100-2, подраздел 6.3). Совпадение профилей ЕППО с необходимой возможностью предполагает, что эти элементы включены в профиль ЕППО;

b) соответствие ЕППО с совместимыми интерфейсами, выраженными их профилями возможностей в случае, когда два вида деятельности являются взаимодействующими.

Примечание 4 – Установочные параметры протокола и сервисов интерфейса ЕППО, согласующихся с соответствующими установками интерфейсов других ЕППО, связанных с другой производственной деятельностью, делают возможным интероперабельность программного обеспечения разных поставщиков, а также запускают в работу их соответствующую производственную деятельность.

Схема согласования профиля возможности ЕППО с необходимым профилем возможности производственной деятельности изображена на рисунке 3.

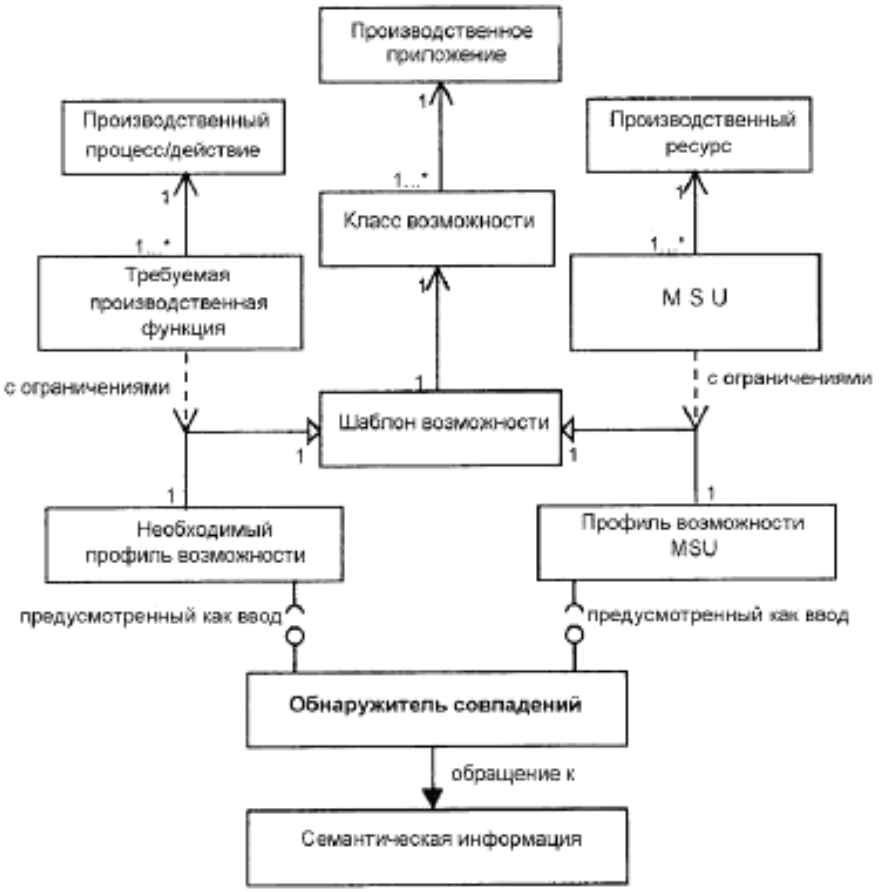

Рисунок 3 – Совпадение профилей возможностей

Все обнаружители совпадений используют семантическую информацию о домене конкретного приложения. Эта семантическая информация может включать в себя информацию об идентичности, создаваемую на основе таксономий, хранящихся в базе данных профиля возможности единицы программного обеспечения.

5.3.2 Обнаружитель совпадений типа 1

Обнаружитель совпадений типа 1 отличается от обнаружителя совпадений типа 2 (см. 5.3.3) тем, что обнаружитель совпадений типа 1 может определять согласование только тех профилей возможностей, которые были созданы с помощью шаблона возможности, полученного из того же класса возможности.

5.3.3 Обнаружитель совпадений типа 2

Обнаружитель совпадений типа 2 может определять согласование тех профилей возможностей, которые были созданы с помощью шаблона возможности,

полученного из того же или разных классов возможностей. В случае разных классов возможностей они относятся к одному либо разным производственным приложениям.

Разные классы возможностей существуют по той причине, что разные предприятия классифицируют производственные функции на основе отличающихся друг от друга доменов деятельности и различных функциональных границ (см. ИСО16100-1, раздел А.1, приложение А). В этом случае одни и те же производственные функции относятся к разным классам возможностей.

Обнаружитель совпадений типа 2 использует семантическую информацию, включающую в себя информацию об идентичности производственных функций разных классов возможностей для осуществления процесса совпадения.

В случае, если обнаружитель совпадений типа 2 приводит в соответствие профили, созданные с помощью шаблонов, соответствующих двум разным классам возможностей одного и того же производственного приложения, то для создания необходимого профиля возможности заказчик приводит описание определенной производственной функции путем заполнения шаблона возможности, соответствующего определенному классу возможности. Для создания профиля возможности ЕППО поставщик приводит описание этой же производственной функции путем заполнения шаблона возможности, соответствующего другому классу возможности. Затем обнаружитель совпадений типа 2 выбирает необходимые профили возможностей для профилей возможностей ЕППО (рисунок 4).

Рисунок 4 – Классы возможностей одного производственного приложения

В других случаях обнаружитель совпадений типа 2 обрабатывает профили, созданные с помощью шаблонов, соответствующих двум разным классам возможностей разных производственных приложений.

5.4 Определение сервиса интерфейса

Для поддержки использования концепции профиля возможности взаимодействия программных средств разных поставщиков согласно ИСО 16100-2 необходимо наличие следующих сервисов:

a) профилирование возможности новой ЕППО;

b) выбор необходимого профиля возможности ЕППО для требуемой возможности;

c) проверка профиля возможности в соответствии с правилами построения профилей;

d) редактирование профилей возможностей и шаблонов;

e) создание новых шаблонов.

Поставщик сервиса профиля возможности (см. рисунок 1) предоставляет следующие типы услуг через интерфейс профиля возможности (CPI), который выполняет сервисы, указанные в перечислениях а) – е):

тип 1 – сервисы, обеспечивающие доступ к профилю возможности ЕППО или ЕППО, необходимой для производственной деятельности, а также возможное согласование двух профилей;

тип 2 – сервисы, обеспечивающие создание и модификацию профиля возможности, а также используемые в процессе профилирования возможности новой ЕППО (см. ИСО 16100-2, рисунки 1 и 2);

тип 3 – сервисы, обеспечивающие создание и модификацию шаблона профилирования возможности и используемые для создания шаблонов (см. ИСО 16100-2, рисунок 3).

Все типы услуг могут быть доступны с помощью одного и того же или разных интерфейсов ЕППО.

В настоящем стандарте приведено только описание определений протокола сервиса типа 1 (см. 6.2).

6 Интерфейс, сервис и протокол профиля возможности

6.1 Использование сервиса профиля возможности

6.1.1 Доступ профиля возможности

Сервисы типа 1 используют для обеспечения доступа к профилям возможностей и их согласования.

Профиль возможности может быть доступен с помощью ЕППО или ресурса, не являющегося частью ЕППО.

6.1.2 Приведение в соответствие двух профилей возможностей

Процесс выбора ЕППО начинают с определения профиля, необходимого для данного действия. Предполагаемое множество ЕППО должно иметь соответствующее множество профилей возможностей, совпадающих с

возможностями, необходимыми для выполнения данного действия. Профиль, необходимый для данного действия, содержит набор обязательных и необязательных возможностей. Множество ЕППО должно, как минимум, предоставлять набор обязательных возможностей для конкретной деятельности.

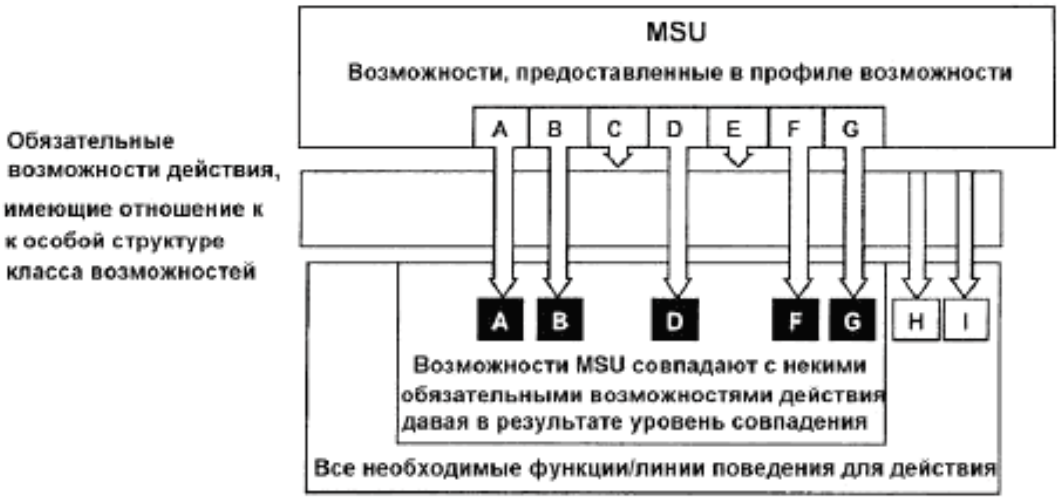

Рисунок 5 – Процесс получения уровня совпадения ЕППО, имеющей отношение к действию

На каждой стадии процесса выбора обязательных и необязательных возможностей, содержащихся в профиле возможностей, они должны совпадать с возможностями профиля, необходимого для определенного действия. Уровень совпадения должен указывать на результат стадии фильтрации (см. рисунок 5).

Отфильтрованные характеристики поведения ЕППО для конкретного действия, сравниваемые с требуемыми характеристиками поведения этого действия, следует использовать как меру уровня совпадения.

Уровень совпадения предполагает одно из следующих значений (см. рисунок 6):

a) полное совпадение – все обязательные и необязательные функции профиля возможностей действия совпадают с профилем ЕППО;

b) полное обязательное совпадение – все обязательные функции профиля возможностей действия совпадают с профилем ЕППО;

c) неполное обязательное совпадение – некоторые обязательные функции профиля возможностей действия совпадают с профилем ЕППО;

d) нет обязательного совпадения – ни одна из обязательных функций профиля возможностей действия не совпадает с профилем возможностей ЕППО.

Примечание 1 – Уровень «некоторое обязательное совпадение» может быть использован для того, чтобы повторно провести процесс выбора во всей структуре класса возможностей.

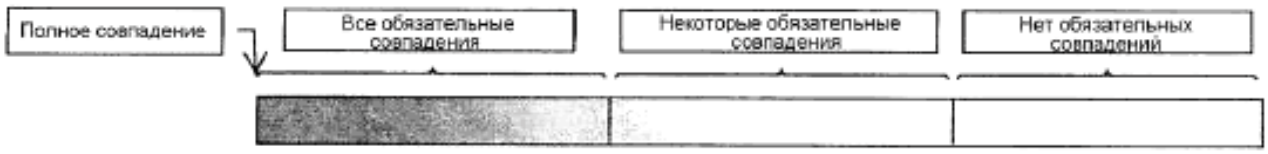

Рисунок 6 – Уровни совпадения

Примечание 2 – Качество обнаружителя совпадений определяет объем и степень полезности доводов, предусмотренных для частичного совпадения, и рекомендаций к способу получения более высокого уровня совпадения (см. 7.4).

Процесс выбора ЕППО считают успешным в случае, если уровень совпадения профилей возможности ЕППО соответствует значению «полное совпадение» или «полное обязательное совпадение».

6.1.3 Базовые элементы набора сервисов типа 1

6.1.3.1 Общие положения

Последовательность сервисов, необходимых для получения доступа и обеспечения выбора профиля возможностей, должна поддерживать следующие случаи, приведенные на рисунке 7:

случай 1 – запрос на профиль требования или профиль ЕППО, если профиль возможностей известен реестру системы; отклик на запрос является профилем;

случай 2 – запрос на профиль ЕППО, если профиль возможностей является резидентной программой ЕППО; отклик на запрос является профилем;

случай 3 – запрос на два профиля, которые необходимо выбрать с помощью сервиса совпадения. Как правило, это один профиль требования и один профиль ЕППО. В параметре запроса приводят либо информацию об обоих профилях или

по желанию заказчика, может быть указан профиль идентификации (ID) одного или обоих вышеуказанных профилей; отклик на запрос является результатом совпадения, который состоит из уровня совпадения и, как минимум, списка совпавших обязательных функций при «неполном обязательном совпадении»;

случай 4 – запрос к ЕППО для обеспечения совпадения ее профиля с профилем ввода. ЕППО может использовать сервис совпадения, аналогичный указанному в случае 3, если профили ввода и ЕППО предоставляются сервису совпадения.

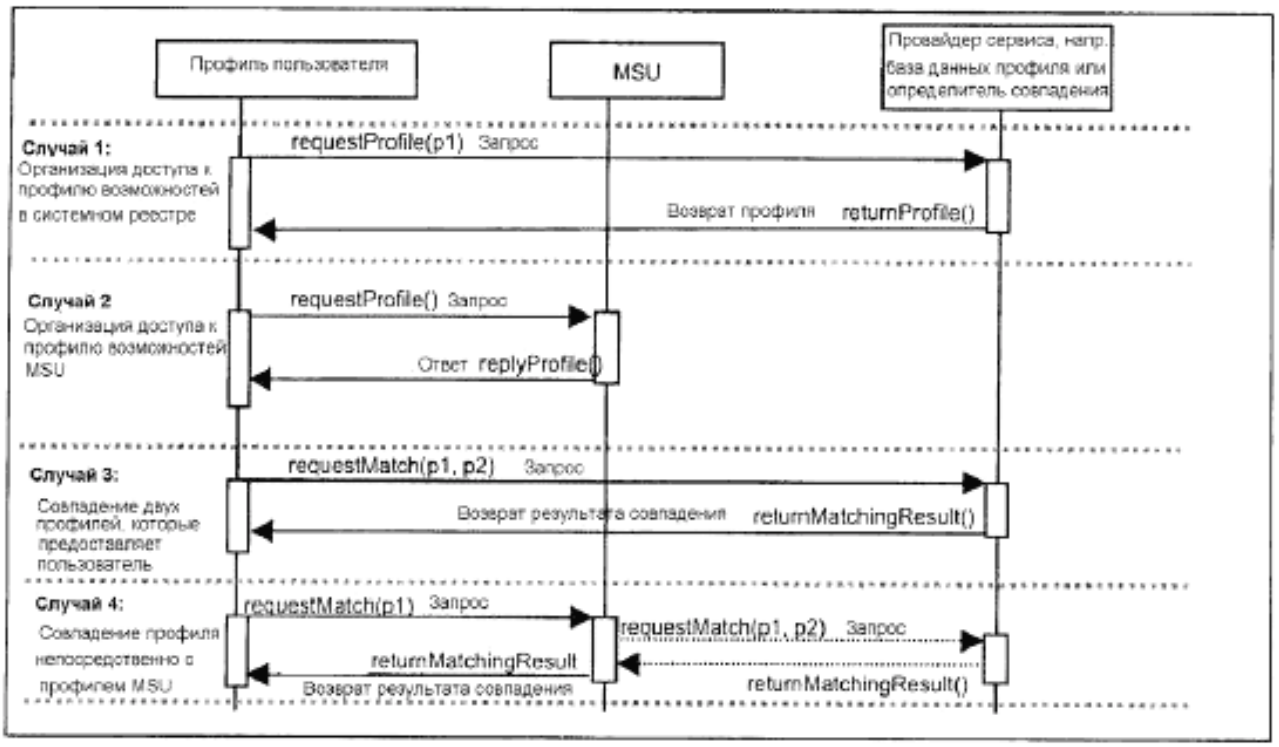

Рисунок 7 – Сервисы типа 1 для доступа к профилю

Точка доступа к сервису ServiceAccessPoint должна быть предоставлена источником профиля возможностей или источником оценки уровня совпадения. Источник, предоставляющий профиль или результат совпадения, выполняет функции поставщика программного средства, а точка назначения профиля или результат совпадения выполняют функции получателя.

Кроме вышеуказанных параметров должен быть обеспечен дополнительный параметр с указанным путем к запрашивающей точке доступа к сервису ServiceAccessPoint.

6.1.3.2 Организация доступа к профилю возможностей через поставщика сервиса (случай 1)

Запрашивающий механизм сервиса запроса профиля requestProfile должен состоять из следующих этапов:

a) пользователь профиля вызывает сервис запроса профиля requestProfile точки доступа к сервису ServiceAccessPoint:

1) параметром обращения к вызову сервиса является:

(i) ID (идентификация) профиля возможностей;

2) работа реагирующего элемента сервиса – асинхронная.

b) поставщик сервиса вызывает сервис возврата профиля returnProfile точки доступа к сервису ServiceAccessPoint:

1) параметрами отклика сервиса являются:

(i) идентификатор (ID) профиля возможностей,

(ii) содержимое профиля,

(iii) статус ошибки доступа;

2) работа инициатора запроса сервиса – асинхронная.

3) запрос другого сервиса не осуществляется до тех пор, пока не будет получен отклик данного сервиса.

Имена профиля возможностей могут ссылаться на профиль возможностей требований или профиль возможностей ЕППО.

6.1.3.3 Организация доступа к профилю возможностей ЕППО (случай 2)

Запрашивающий механизм сервиса запроса профиля requestProfile должен состоять из следующих этапов:

a) пользователь профиля вызывает сервис запроса профиля requestProfile объекта ЕППО:

1) параметры обращения к вызову сервиса отсутствуют,

2) работа реагирующего элемента сервиса – асинхронная;

b) ЕППО вызывает сервис возврата профиля returnProfile точки доступа к сервису ServiceAccessPoint:

1) параметрами отклика сервиса являются:

(i) имя профиля возможностей,

(ii) содержимое профиля,

(iii) статус ошибки доступа;

2) работа инициатора запроса сервиса – асинхронная;

3) запрос другого сервиса не осуществляется до тех пор, пока не будет получен отклик сервиса.

Полученный (возвращенный) профиль является собственным профилем возможностей запрошенной ЕППО.

6.1.3.4 Совпадение двух профилей через определитель совпадений (случай 3)

Запрашивающий механизм сервиса запроса совпадения requestMatch должен состоять из следующих этапов:

a) пользователь профиля вызывает сервис запроса совпадения requestMatch от поставщика сервиса ServiceProvider:

1) параметрами обращения к вызову сервиса являются:

(i) ID первого профиля возможностей,

(ii) ID второго профиля возможностей,

2) работа инициатора запроса сервиса – асинхронная;

b) поставщик программного средства вызывает сервис возврата результата совпадения returnMatchingResult точки доступа к сервису ServiceAccessPoint:

1) параметрами отклика сервиса являются:

(i) ID первого профиля возможностей,

(ii) ID второго профиля возможностей,

(iii) результат совпадения,

(iv) статус ошибки совпадения;

2) работа инициатора запроса сервиса – асинхронная;

3) запрос другого сервиса не осуществляется до тех пор, пока не будет получен отклик сервиса.

Имена профиля возможностей могут ссылаться на профиль возможностей требований или профиль возможностей ЕППО.

6.1.3.5 Совпадение профиля непосредственно с профилем MSU (случай 4)

Запрашивающий механизм сервиса запроса совпадения requestMatch должен состоять из следующих этапов:

a) пользователь профиля вызывает сервис запроса совпадения requestMatch точки доступа к сервису ServiceAccessPoint:

1) параметром обращения к вызову сервиса является:

(i) имя возможности,

2) работа инициатора запроса сервиса – асинхронная;

b) автор программного средства вызывает сервис результата совпадения returnMatchingResult точки доступа к сервису ServiceAccessPoint:

1) параметрами отклика сервиса являются:

(i) имя первого профиля возможностей,

(ii) имя профиля возможностей ЕППО,

(iii) результат совпадения,

(iv) статус ошибки совпадения;

2) работа реагирующего элемента сервиса – асинхронная;

3) запрос другого сервиса не происходит до тех пор, пока не получен отклик сервиса.

Имена профиля возможностей могут ссылаться на профиль возможностей требований или профиль возможностей ЕППО.

6.1.4 Общие сервисы менеджмента для профилирования возможностей и процесса анализа

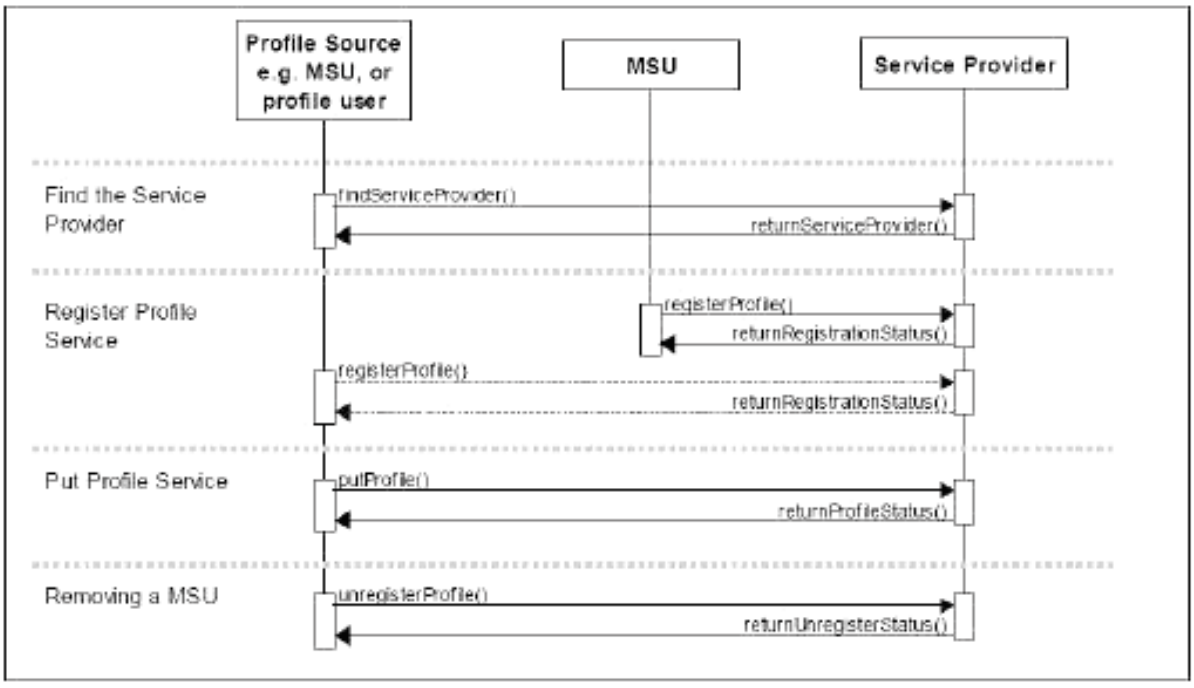

Рисунок 8 – Общие сервисы менеджмента

6.1.4.1 Общие положения

Описание общих сервисов менеджмента, предоставляющих интерфейс профиля возможностей (CPI), приведено в 6.1.4. Эти сервисы поддерживают набор сервисов типа 1 для получения профилей и приведения их в соответствие.

ЕППО предлагает собственную возможность взаимодействия, а также находит другие возможности поддерживания интероперабельности общей производственной деятельности. Все ЕППО, входящие в процесс профилирования, должны использовать сервисы общего менеджмента, предоставляющие интерфейс профиля возможностей (CPI).

ЕППО или другие источники профиля взаимодействуют с поставщиком сервисов, используя общие сервисы менеджмента.

Общие сервисы менеджмента допускают возможность интероперабельности программных изделий разных поставщиков для обработки:

a) структуры профилей;

b) содержания профилей.

6.1.4.2 Нахождение поставщика сервиса

Запрашивающий механизм сервиса findServiceProvider, обеспечивающий поиск поставщика сервиса, должен состоять из следующих этапов:

a) пользователь профиля вызывает сервис, обеспечивающий поиск поставщика сервиса findServiceProvider точки доступа к сервису ServiceAccessPoint:

1) параметры обращения к вызову сервиса отсутствуют;

2) поведение инициатора запроса сервиса – асинхронное;

b) автор программного средства вызывает сервис поставщика сервиса returnServiceProvider точки доступа к сервису ServiceAccessPoint:

1) параметрами отклика сервиса являются:

(i) ссылка на поставщика сервиса через универсальный идентификатор ресурса (URI),

(ii) статус ошибки доступа;

2) поведение реагирующего элемента сервиса – асинхронное;

3) запрос другого сервиса не осуществляется до тех пор, пока не будет получен отклик сервиса.

6.1.4.3 Регистрация профиля у поставщика сервиса

Сервис регистрации профиля registerProfile должен предоставлять источнику профиля, например ЕППО, возможность отправить однозначный и различимый ID профиля. Сервис регистрации профиля registerProfile должен выполнять следующие задачи с указанными параметрами:

a) пользователь профиля вызывает сервис статуса регистрации returnRegistrationStatus точки доступа к сервису ServiceAccessPoint:

1) параметром обращения к вызову сервиса является:

(i) ID профиля возможностей,

2) поведение инициатора запроса сервиса – асинхронное;

b) автор программного средства вызывает сервис возврата статуса регистрации returnRegistrationStatus точки доступа к сервису ServiceAccessPoint:

1) параметрами отклика регистрации являются:

(i) статус регистрации,

(ii) статус ошибки доступа;

2) поведение реагирующего элемента сервиса – асинхронное;

3) запрос другого сервиса не осуществляется до тех пор, пока не будет получен отклик сервиса.

Даже если регистрацию профиля выполняет ЕППО, любой пользователь профиля также может зарегистрировать профиль.

6.1.4.4 Поставка профиля поставщику сервиса

Сервис поставки профиля putProfile должен предоставлять пользователю сервиса возможность отправить профиль поставщику сервиса. Сервис поставки профиля putProfile должен выполнять следующие задачи с указанными параметрами:

a) пользователь профиля вызывает сервис поставки профиля putProfile точки доступа к сервису ServiceAccessPoint:

1) параметрами обращения к вызову сервиса являются:

(i) ID профиля возможностей,

(ii) профиль возможностей;

2) поведение инициатора запроса сервиса – асинхронное;

b) автор программного средства вызывает сервис статуса поставки профиля returnPutProfileStatus точки доступа к сервису ServiceAccessPoint:

1) параметрами отклика поставки профиля putProfile являются:

(i) статус передачи профиля,

(ii) статус ошибки доступа;

2) поведение реагирующего элемента сервиса – асинхронное;

3) запрос другого сервиса не осуществляется до тех пор, пока не будет получен отклик сервиса.

Сервис поставки профиля putProfile является действием «активной доставки» «push» для соответствующего действия «получения информации» «pull» профиля requestProfile.

6.1.4.5 Отсутствие регистрации профиля у поставщика сервиса

Сервис нерегистрации профиля unregisterProfile должен быть использован источником профиля, например ЕППО, с целью предотвращения регистрации до удаления соответствующей ЕППО. Сервис нерегистрации профиля unregisterProfile должен выполнять следующие задачи с указанными параметрами:

a) пользователь профиля вызывает сервис нерегистрации профиля unregisterProfile точки доступа к сервису ServiceAccessPoint:

1) параметром обращения к вызову сервиса является:

(i) ID профиля возможностей;

2) поведение инициатора запроса сервиса – асинхронное;

b) автор программного средства вызывает сервис статуса отсутствия регистрации returnUnregistrationStatus точки доступа к сервису ServiceAccessPoint:

1) параметрами отклика регистрации являются:

(i) статус отсутствия регистрации,

(ii) статус ошибки доступа;

2) поведение реагирующего элемента сервиса – асинхронное;

3) запрос другого сервиса не осуществляется до тех пор, пока не будет получен отклик сервиса.

6.1.5 Проверка достоверности профилей возможностей

Проверка достоверности переданных профилей возможностей, например путем поставки профиля putProfile или запроса профиля requestProfile, является обязанностью поставщика сервиса типа 2 или типа 3, который обеспечивает общий сервис проверки достоверности CommonValidationService.

Профиль возможностей должен состоять из общей и специальной частей. Общий сервис проверки достоверности CommonValidationService должен обеспечивать оценку профиля возможностей путем его сравнения с общей частью схемы, приведенной в 7.2, и специальной частью профиля возможностей.

Все строки, использованные в профилях возможностей, должны быть закодированы с помощью UTF-8.

6.2 Спецификации протоколов

6.2.1 Синтаксис URL сервиса

Протоколы обеспечивают прямой доступ запросов к сервисам. Запрос обрабатывается сервисом, который реагирует путем отправки ответа сервису.

Унифицированный указатель информационного ресурса (URL сервис) начинается со строки «service:». URL сервис включает в себя тип сервиса, затем следует соответствующая точка сервисного доступа до символа «:», которая не входит в эту строку и с которой начинается спецификация адреса. Информация атрибута, характерная для сервиса, следует за спецификацией адреса, закодированного с использованием грамматики URL.

URL сервис должен быть следующим:

service:<service-type>:<service-access-point>://<address>;<attribute-list>

Перечень атрибутов состоит из перечня назначений атрибутов, разделенных точкой с запятой « ; » . Назначения атрибутов должны иметь следующую форму:

<attribute-id>=<attribute-value>

Для атрибутов ключевого слова используют следующую форму:

<attribute-id>

6.2.2 Протокол сервиса типа 1

6.2.2.1 Профиль возможностей запросов

Сервис запроса профиля requestProfile возвращает профиль возможностей интероперабельности требований или профиль возможностей интероперабельности ЕППО и должен быть следующим:

<service-type> = «requestCapabilityProfile»

Соответствующий атрибут должен быть следующим:

capability_profile_ID =«the_capability_profile_id»

Сервис возврата профиля returnProfile возвращает профиль возможностей интероперабельности требований или профиль возможностей интероперабельности ЕППО и должен быть следующим:

<service-type> = «returnCapabilityProfile»

Соответствующие атрибуты должны быть следующим:

capability_profile_ID = «the_capability_profile_id»

capability_profile_cjntent =«the_capability_profile_content»

access_status = «the_access_status»

6.2.2.2 Организация доступа профиля возможностей ЕППО

Сервис запроса профиля requestProfile запрашивает профиль возможностей ЕППО и должен быть следующим:

<service-type> = «requestCapabilityProfile»

Атрибуты отсутствуют.

Сервис возврата профиля returnProfile возвращает профиль возможностей интероперабельности ЕППО и должен быть следующим:

<service-type> = «returnCapabilityProfile»

Соответствующие атрибуты должны быть следующими:

capability_profile_ID = «the_capability_profile_id»

capability_profile_cjntent = «the_capability_profile_content»

access_status = «the_access_status»

6.2.2.3 Совпадение двух профилей

Сервис совпадения профиля requestMatch возвращает результат совпадения и должен быть следующим:

<service-type> = «requestCapabilityProfileMatch»

Соответствующие атрибуты должны быть следующими:

capability_profile_1_ID =« the_first_capability_profile_id»

capability_profile_2_ID =« the_second_capability_profile_id»

Сервис возврата результата совпадения returnMatchingResult возвращает результат совпадения и должен быть следующим:

<service-type> = «returnCapabilityProfileMatchResult»

Соответствующие атрибуты должны быть следующими:

capability_profile_1_ID =« the_first_capability_profile_id»
capability_profile_2_ID =« the_second_capability_profile_id»

matching_result =« the_matching_level;the_matching_comment»

access_status =« the_access_status»

6.2.2.4 Приведение в соответствие профиля с профилем ЕППО

Сервис запроса совпадения requestMatch возвращает результат приведения в соответствие и должен быть следующим:

<service-type> = «requestCapabilityProfileMatch»

Соответствующий атрибут должен быть следующим:

capability_profile_ID = «the_capability_profile_id»

Сервис возврата результата приведения в соответствие returnMatchingResult возвращает результат приведения в соответствие и должен быть следующим:

<service-type> = «returnCapabilityProfileMatchResult»

Соответствующие атрибуты должны быть следующими:

capability_profile_1_ID = «the_first_capability_profile_id»

capability_profile_2_ID = «the_MSU_capability_profile_id»

matching_result = «the_matching_level;the_matching_comment»

access_status = «the_access_status»

6.2.3 Протокол общего сервиса менеджмента

6.2.3.1 Нахождение поставщика сервиса

Сервис нахождения поставщика сервиса findServiceProvider осуществляет возврат поставщика сервиса и должен быть следующим:

<service-type> = «requestServiceProvider»

Атрибуты отсутствуют.

Сервис возврата поставщика сервиса returnServiceProvider возвращает поставщика требуемого сервиса и должен быть следующим:

<service-type> = «returnServiceProvider»

Соответствующие атрибуты должны быть следующими:

service_provider_URI= «the_service_provider_URI»

access_status= «the_access_status»

6.2.3.2 Регистрация профиля возможностей

Сервис регистрации профиля registerProfile осуществляет регистрацию профиля возможностей с ID и должен быть следующим:

<service-type> = «registerCapabilityProfile»

Соответствующий атрибут должен быть следующим:

capability_profile_ID= «the_capability_profile_id»

Сервис возврата статуса регистрации returnRegistrationStatus осуществляет возврат статуса регистрации профиля возможностей и должен быть следующим:

<service-type> = «returnRegistrationStatus»

Соответствующие атрибуты должны быть следующими:

registration_status = «the_registration_status»

access_status = «the_access_status»

6.2.3.3 Поставка профиля поставщику сервиса

Сервис поставки профиля putProfile принимает профиль возможностей с ID и должен быть следующим:

<service-type> = «putCapabilityProfile»

Соответствующие атрибуты должны быть следующим:

capability_profile_ID = «the_capability_profile_id»

capability_profile_content = «the_capability_profile_content»

Сервис возврата, определяющий статус поставки профиля returnPutProfileStatus, возвращает статус передачи профиля возможностей и должен быть следующим:

<service-type> = «returnPutProfileStatus»

Соответствующие атрибуты должны быть следующими:

transmission_status = «returnPutProfileStatus»

access_status =« the_access_status»

6.2.3.4 Незарегистрированный профиль возможностей

Сервис профиля unregisterProfile не регистрирует профиль возможностей с ID и должен быть следующим:

<service-type> = «unRegisterCapabilityProfile»

Соответствующий атрибут должен быть следующим:

capability_profile_ID = «the_capability_profile_id»

Профиль сервиса возврата статуса нерегистрации returnUnregistrationStatus возвращает статус нерегистрации профиля возможностей и должен быть следующим:

<service-type> = «returnUnRegistrationStatus»

Соответствующие атрибуты должны быть следующими:

unregistration_status = «the_unregistration_status»

access_status = «the_access_status»

6.2.4 Протоколы сервисов типа 2 и типа 3

Спецификации протоколов сервисов типа 2 и типа 3 будут приведены в следующем издании комплекса стандартов ИСО 16100.

7 Шаблоны

7.1 Структура

7.1.1 Общие положения

Структура любого шаблона возможностей интероперабельности должна быть описана с использованием XML схемы, с помощью которого можно определить элементы и их атрибуты.

Структура шаблона состоит из двух частей:

a) общая часть типа commonPartType;

b) специальная часть типа specificPartType.

7.1.2 Формальная структура

Формальной структурой шаблона является следующая:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="ISO16100CapabilityProfileSchema"
xmlns:cpsc="ISO16100CommonSchema" xmlns="ISO16100CapabilityProfileSchema"
xmlns:xs="http://www.w3.org/2001/XMLSchema" id="CapabilityProfiling" xml:lang="EN">
 <xs:element name="CapabilityProfiling">
  <xs:complexType>
   <xs:sequence maxOccurs="unbounded">
    <xs:element name="type">
     <xs:complexType>
      <xs:attribute name="id" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="CapabilityProfile">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="pkgtype">
```

```
            <xs:complexType>
              <xs:attribute name="version" type="xs:string" form="unqualified"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="Common" type="CommonPartType"/>
          <xs:element name="Specific" type="SpecificPartType"/>
        </xs:sequence>
        <xs:attribute name="date" type="xs:string" form="unqualified"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

7.2 Общая часть

7.2.1 Общие положения

Общая часть шаблона должна состоять из следующих основных элементов схемы (см. ИСО 16100-2, рисунок 6):

Выбор требования профиля ЕППО выполняют с помощью идентификатора: Requirement или MSU_Capability.

В обоих случаях идентификатор ID добавляют как атрибут.

ID	–	строка, в которой указано, что ЕППО отличается от других ЕППО по уровню совпадения

Затем должна следовать неограниченная последовательность пар эталонной структуры класса возможностей и идентификатор шаблона:

Reference Capability Class Structure	–	указывает на классы деятельности;
TemplateID	–	идентификатор, с помощью которого различают начальный класс в рамках эталонной структуры класса возможностей. Как правило, это значение равно нулю в том случае, если совпадение требуется для описания полной структуры класса возможностей.

В общей части шаблона, предназначенного для разных областей приложения, допускается приводить ссылки на более чем одну эталонную структура класса

возможностей. В специальной части для каждого имени опорной структуры класса возможностей должен быть определен подходящий профиль.

Формальная структура общей части шаблона приведена в 7.2.2 настоящего стандарта с использованием характеристики содержания шаблона, приведенной в 6.1.2 ИСО 16100-2, и концептуальной структуры шаблона, приведенной в 6.3 ИСО 16100-2.

7.2.2 Формальная структура

Формальной структурой общей части типа шаблона CommonPartType является следующая:

```
<xs:complexType name="CommonPartType">
 <xs:sequence>
  <xs:choice>
   <xs:element name="Requirement">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="ID" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" form="unqualified"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="MSU_Capability">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="ID" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" form="unqualified"/>
   </xs:complexType>
  </xs:element>
 </xs:choice>
 <xs:sequence maxOccurs="unbounded">
  <xs:element name="ReferenceCapabilityClassStructure">
   <xs:complexType>
    <xs:attribute name="id" type="xs:string" form="unqualified"/>
    <xs:attribute name="name" type="xs:string" form="unqualified"/>
    <xs:attribute name="version" type="xs:string" form="unqualified"/>
    <xs:attribute name="url" type="xs:string" form="unqualified"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="TemplateID">
   <xs:complexType>
    <xs:attribute name="ID" type="xs:string" form="unqualified"/>
```

```
  </xs:complexType>
 </xs:element>
</xs:sequence>
<xs:element name="Version">
 <xs:complexType>
  <xs:attribute name="major" type="xs:string" form="unqualified"/>
  <xs:attribute name="minor" type="xs:string" form="unqualified"/>
 </xs:complexType>
</xs:element>
<xs:element name="Owner">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="name" type="xs:string" minOccurs="0"/>
   <xs:element name="street" type="xs:string" minOccurs="0"/>
   <xs:element name="city" type="xs:string" minOccurs="0"/>
   <xs:element name="zip" type="xs:string" minOccurs="0"/>
   <xs:element name="state" type="xs:string" minOccurs="0"/>
   <xs:element name="country" type="xs:string" minOccurs="0"/>
   <xs:element name="comment" type="xs:string" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="ComputingFacilities" minOccurs="0" maxOccurs="unbounded">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Processor0" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:attribute name="type" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="OperatingSystem0" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:attribute name="type" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="Language" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:attribute name="name" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="Memory" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:attribute name="size" type="xs:string" form="unqualified"/>
     <xs:attribute name="unit" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="DiskSpace" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
```

```
      <xs:attribute name="size" type="xs:string" form="unqualified"/>
      <xs:attribute name="unit" type="xs:string" form="unqualified"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="type" type="xs:string" form="unqualified"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="Performance" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
   <xs:attribute name="ElapsedTime" type="xs:string" form="unqualified"/>
   <xs:attribute name="TransactionsPerUnitTime" type="xs:string" form="unqualified"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="ReliabilityData" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="UsageHistory" type="xs:string" minOccurs="0"/>
    <xs:element name="Shipments" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="number" type="xs:string" form="unqualified"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="IntendedSafetyIntegrity" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="level" type="xs:string" form="unqualified"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Certification" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="no1" type="xs:string" form="unqualified"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="SupportPolicy" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
   <xs:attribute name="index" type="xs:string" form="unqualified"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="PriceData" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
   <xs:attribute name="invest" type="xs:string" form="unqualified"/>
   <xs:attribute name="annualSupport" type="xs:string" form="unqualified"/>
   <xs:attribute name="unit" type="xs:string" form="unqualified"/>
  </xs:complexType>
   </xs:element>
```

```
</xs:sequence>
</xs:complexType>
```

Примечание – Пример шаблона приведен в разделе А.1 приложения А.

7.3 Специальная часть

Специальная часть шаблона должна быть представлена в виде XML схемы. Специальная часть шаблона может состоять из одного или нескольких специальных комплексных элементов. Число этих элементов устанавливают в общей части шаблона. Каждый элемент специальных комплексных элементов должен относиться к одной эталонной структуре класса возможностей.

Это позволяет определить и выразить структуру класса, следующую из модели деятельности.

Каждый шаблон должен быть получен из своей эталонной структуры класса возможностей. Шаблон должен характеризовать всю или часть опорной структуры класса возможностей. Эталонная структура класса возможностей может иметь несколько ассоциированных шаблонов, охватывающих части полной структуры.

Требуемый профиль возможностей и профиль возможностей ЕППО должны характеризоваться структурой шаблона, представленной в виде XML схемы. Эти шаблоны должны быть производными одной и той же эталонной структуры класса возможностей.

Примечание – Примеры приведены в разделах А.2 и А.3 приложения А.

7.4 Использование шаблонов

Шаблоны используют для создания профилей возможностей интероперабельности требований, а также профилей возможностей интероперабельности ЕППО разных поставщиков.

В процессе приведения в соответствие с целью совпадения профилей осуществляют сравнение двух профилей. Для этого необходима общая базовая структура, которая определяется структурой шаблона, представленного в виде XML схемы.

Правила приведения в соответствие зависят от типа функции. Определитель совпадений использует семантическую информацию о домене специфического приложения.

Определитель совпадений должен вернуть обратно уровень приведения в соответствие согласно 6.1.2 и комментарий к процессу приведения в соответствие.

Примечание – Пример процесса приведения в соответствие приведен в разделе А.2 приложения А.

8 Соответствие

Концепции и правила оценки соответствия профилей возможностей интероперабельности аппаратных и программных средств разных поставщиков подробно изложены в ИСО 16100-4.

Приложение А

(справочное)

Шаблон профиля возможностей

А.1 Общий шаблон профиля возможностей

А.1.1 Заполненный шаблон

Заполненный шаблон является профилем возможностей. Примером заполненного шаблона является следующий:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ... -->
<CapabilityProfiling xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\...\ISO16100-General.xsd">
  <Type id="ReqProf_0012"/>
  <CapabilityProfile date="2003-12-11">
    <pkgtype version="1,0,0,2"/>
    <Common>
      <!-- Fill in the common part of the capability profile -->
      <!-- sample in A.1.2 -->
    </Common>
    <Specific>
      <!-- Fill in the specific part of the capability profile -->
      <!-- two samples, one in A.2.2 and one in A.3.2 -->
    </Specific>
  </CapabilityProfile>
</CapabilityProfiling>
```

А.1.2 Образец общей части шаблона

Образцом общей части шаблона является следующий, два образца специальной части приведены в А.2.2 и А.3.2.

```
        <Common>
          <Requirement ID="abc-Req-672726"/>
          <ReferenceCapabilityClassStructure id="CRCS_001" name="CEA_ReferenceCapabilitySchema"
version="1.0.1" url="www.asam.net/..."/>
          <TemplateID ID=" "/>
          <Version major="1" minor="1"/>
          <Owner>
            <name>ABC Inc.</name>
            <street>Waterstreet 56</street>
            <city>Softcity</city>
            <zip>734562</zip>
```

```
    <state>Alabama</state>
    <country>USA</country>
    <comment>This is a comment</comment>
   </Owner>
   <ComputingFacilities type="PC">
    <Processor type="Intel"/>
    <OperatingSystem type="WindowsXP"/>
    <Language name="en"/>
    <Memory size="138" unit="kByte"/>
    <DiskSpace size="23" unit="MByte"/>
   </ComputingFacilities>
   <Performance ElapsedTime="14ms" TransactionsPerUnitTime="743"/>
   <ReliabilityData>
    <UsageHistory>history discription</UsageHistory>
    <Shipments number="65"/>
    <IntendedSafetyIntegrity level="high"/>
    <Certification no="ISO9001"/>
   </ReliabilityData>
   <SupportPolicy index="String"/>
   <PriceData invest="10000" annualSupport="5000" unit="$"/>
  </Common>
```

А.2 Производственная структура класса возможностей

А.2.1 Образец эталонной структуры класса с использованием синтаксиса языка XML

Эталонная структура класса должна соответствовать требованиям 6.2.1 и 6.3 ИСО 16100-2. Следующая структура не является полным описанием схемы, так как охарактеризованы не все узлы, но она иллюстрирует главный принцип. Описание заполненного шаблона приведено в А.2.2 и А.2.3.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Reference Capability Class Structure -->
<ReferenceCapabilityClassStructure id="CRCS_001" name="DiscreteManufacturingActivity"
version="1.0.1" url="www.xxx.net/..."/>
<TemplateID ID="861"/>
<Version major="1" minor="1"/>
 <ManufacturingActivity level="0" ID="A">
  <DevelopProducts level="1" ID="AA">
   <EngineeringProcess level="2" ID="AA2">
    <DevelopDetailedProcessPlan level="3" ID="AA22">
     <GenerateProcessSequence level="4" ID="AA221">
      <ScheduleSetUp level="5" ID="AA2211">
      <!-- see Fig.B10 of part 1 -->
      </ScheduleSetUp>
      <ScheduleHandling level="5" ID="AA2212"></ScheduleHandling>
      <ScheduleProcessing level="5" ID="AA2213">
       <ScheduleInspection level="6" ID="AA22131"></ScheduleInspection>
       <SchedulePartMaking level="6" ID="AA22132">
        <ScheduleShaping level="7" ID="AA221321">
         <ScheduleMaterialRemoving level="8" ID="AA2213211">
          <!-- see Annex B of part 1 -->
          <ScheduleMechanicalRemoving level="9" ID="AA22132111">
           <ScheduleMachining level="10" ID="AA221321111">
            <ScheduleSinglePointCutting level="11" ID="AA2213211111">
```

```
          <Boring level="12" ID="AA22132111111">
          </Boring>
          <Threading_Sp level="12" ID="AA22132111112">
          </Threading_Sp>
          <Turning level="12" ID="AA22132111113">
          </Turning>
         </ScheduleSinglePointCutting>
         <ScheduleMultiplePointCutting level="11" ID="AA2213211112">
          <Drilling level="12" ID="AA22132111121">
          </Drilling>
          <Milling level="12" ID="AA22132111122">
          </Milling>
          <Threading_Mp level="12" ID="AA22132111123">
          </Threading_Mp>
         </ScheduleMultiplePointCutting>
        </ScheduleMachining>
       </ScheduleMechanicalRemoving>
       <ScheduleChemicalRemoving>
        <!-- continue -->
       </ScheduleChemicalRemoving>
      </ScheduleMaterialRemoving>
     </ScheduleShaping>
    </SchedulePartMaking>
   </ScheduleProcessing>
   <ScheduleLoadUnload></ScheduleLoadUnload>
   <ScheduleIdling></ScheduleIdling>
  </GenerateProcessSequence>
  <GenerateOperations level="4" ID="AA222"></GenerateOperations>
 </DevelopDetailedProcessPlan>
</EngineeringProcess>
</DevelopProducts>
</ManufacturingActivity>
<ManufacturingInformationExchange level="0" ID="C">
 <ComputingFacilitiesRequired level="1" ID="CC">
  <Processor level="2" ID="CC1">
   <RISC level="3" ID="CC11"/>
   <CISC level="3" ID="CC12"/>
   <DSP level="3" ID="CC13"/>
   <ASIC level="3" ID="CC14"/>
  </Processor>
  <OperatingSystem level="2" ID="CC2">
   <Windows level="3" ID="CC21">
    <XP level="4" ID="CC211">
    </XP>
   </Windows>
   <JavaVM level="3" ID="CC22">
   </JavaVM>
   <Linux level="3" ID="CC23">
   </Linux>
   <Unix level="3" ID="CC24">
   </Unix>
   <GenericPLC_OS level="3" ID="CC25">
   </GenericPLC_OS>
  </OperatingSystem>
  <Language level="2" ID="CC3">
   <!-- refer to proper standard in ISO -->
  </Language>
  <Memory level="2" ID="CC4">
   <Volatile level="3" ID="CC41">
   </Volatile>
   <NonVolatile level="3" ID="CC42">
   </NonVolatile>
  </Memory>
  <DiskSpace level="2" ID="CC5">
```

```
    <capacity>40GB</capacity>
    <partitioning>fixed</partitioning>
   </DiskSpace>
  </ComputingFacilitiesRequired>
 </ManufacturingInformationExchange>
</ReferenceCapabilityClassStructure>
```

А.2.2 Пример профиля возможностей требований

В следующем примере приведен профиль возможностей требований разных поставщиков с использованием шаблона согласно приложению В в соответствии с моделью деятельности В-2 (разработка изделия) по ИСО 16100-1. Подробности приведены на рисунке В.11 настоящего стандарта.

```
<?xml version="1.0" encoding="UTF-8"?>
<CapabilityProfiling xmlns="http://tempuri.org/CapabilityProfileTemplate.xsd">
  <Type id="ReqProf_786z7" />
  <CapabilityProfile date="2003-09-11">
  <pkgtype version="1,0,0,2"></pkgtype>
    <Common>
      <Requirement ID="81-0001"/>
      <ReferenceCapabilityClassStructure id="rcs_1001" name="DiscreteManufacturingActivity"
version="001" url="" />
      <TemplateID ID="manuAct32" />
      <Version major="7" minor="3" />
      <Owner>
        <name>MM Production Inc.</name>
        <street>Summer Ave.7</street>
        <city>Softcity</city>
        <zip>4711</zip>
        <state>CA</state>
        <country>USA</country>
        <comment>Only best experiences!</comment>
      </Owner>
      <ComputingFacilities type="required">
        <Processor type="INTEL" />
        <OperatingSystem type="LINUX" />
        <Language name="EN" />
        <Memory size="28" unit="MB" />
        <DiskSpace size="30" unit="GB" />
      </ComputingFacilities>
      <Performance ElapsedTime="61ms" TransactionsPerUnitTime="621" />
      <ReliabilityData>
        <UsageHistory>
          abc1
          abc2
         </UsageHistory>
        <Shipments number="55" />
        <IntendedSafetyIntegrity level="3" />
        <Certification no1="ISO9001" />
      </ReliabilityData>
      <SupportPolicy index="23" />
      <PriceData invest="12000" annualSupport="2400" unit="USD" />
      <TemplateID ID_Number="Ex_A22_DevelopProdcuct_ISO-DIS16100-01" />
    <!-- gives the uppest level, relative root; e.g. to level 4-->
    <CapabilityClassName id="ccn_1001"> GenerateProcessSequence </CapabilityClassName>
    <CapabilityClassName id="ccn_1002"> GenerateControlPrograms </CapabilityClassName>
  </Common>
  <Specific>
    <ReferenceCapabilityClassStructure id="rcs_1001">
      <ManufacturingActivity>
        <!-- for standard matching rules -->
        <Activity id="act_2001" level="12" mandatoryLevel="10">
          <Boring>
```

```
          </Boring>
        </Activity>
        <Activity id="act_2002" level="12" mandatoryLevel="10">
          <Threading_sp>
          </Threading_sp>
        </Activity>
        <Activity id="act_2003" level="12" mandatoryLevel="10">
          <Threading_Mp>
          </Threading_Mp>
        </Activity>
        <Activity id="act_2004" level="12" mandatoryLevel="10">
          <Drilling attrl="abc">
            <Material type="Copper">
              <Depth> 100mm </Depth>
              <Speed> 5mm/s </Speed>
            </Material>
          </Drilling>
        </Activity>
        <!-- end of standard matching rules -->
      </ManufacturingActivity>
      <ManufacturingInformationExchange>
        <Processor id="111" level="3" mandatoryLevel="3">
          <RISC />
        </Processor>
        <OperatingSystem id="112" level="3" mandatoryLevel="2">
          <XP />
        </OperatingSystem>
      </ManufacturingInformationExchange>
    </ReferenceCapabilityClassStructure>
  </Specific>
 </CapabilityProfile>
</CapabilityProfiling>
```

A.2.3 Пример профиля возможностей ЕППО

Примером описания профиля возможностей ЕППО в соответствии с моделью B-5, приведенной в ИСО 16100-1, является следующее:

```
<?xml version="1.0" encoding="UTF-8"?>
<CapabilityProfiling xmlns="http://tempuri.org/CapabilityProfileTemplate.xsd">
 <Type id="MSU_Profile" />
 <CapabilityProfile date="2003-09-11">
 <pkgtype version="1,0,0,2"></pkgtype>
  <Common>
   <MSU_Capability ID="xyz-MSU-101010"/>
   <ReferenceCapabilityClassStructure id="rcs_1001" name="DiscreteManufacturingActivity"
version="001" url="" />
   <ReferenceCapabilityClassStructure id="rcs_1002" name="DiscreteEngineeringActivity"
version="001" url="" />
   <Requirement ID="81-0001"></Requirement>
   <Version major="2" minor="3" />
   <Owner>
    <name>MSU Developer Inc.</name>
    <street>Winter Ave.7</street>
    <city>Softcity</city>
    <zip>4712</zip>
    <state>CA</state>
    <country>USA</country>
    <comment>Only best experiences!</comment>
   </Owner>
   <ComputingFacilities type="required">
    <!-- see Fig 6 part 2 -->
    <Processor type="INTEL" />
    <OperatingSystem type="LINUX" />
    <Language name="EN" />
```

```
 <Memory size="28" unit="MB" />
 <DiskSpace size="30" unit="GB" />
</ComputingFacilities>
<Performance ElapsedTime="61ms" TransactionsPerUnitTime="621" />
<TemplateID ID_Number="Ex_A22_DevelopProdcuct_ISO-DIS16100-01" />
<!-- gives the uppest level, relative root; e.g. to level 4-->
<CapabilityClassName id="ccn_1001"> GenerateProcessSequence </CapabilityClassName>
<CapabilityClassName id="ccn_1002"> GenerateControlPrograms </CapabilityClassName>
</Common>
<Specific>
 <ReferenceCapabilityClassStructure id="rcs_2001">
  <ManufacturingActivity>
   <!-- for standard matching rules -->
   <Activity id="act_2001" level="12" mandatoryLevel="10">
    <Boring>
    </Boring>
   </Activity>
   <Activity id="act_2002" level="12" mandatoryLevel="10">
    <Threading_sp>
    </Threading_sp>
   </Activity>
   <Activity id="act_2003" level="12" mandatoryLevel="10">
    <Threading_Mp>
    </Threading_Mp>
   </Activity>
   <Activity id="act_2004" level="12" mandatoryLevel="10">
    <Drilling attr1="abc">
     <Material type="Copper">
      <Depth> 200mm </Depth>
      <Speed> 8mm/s </Speed>
     </Material>
     <Material type="Steel">
      <Depth> 120mm </Depth>
      <Speed> 4mm/s </Speed>
     </Material>
     <Material type="Aluminium">
      <Depth> 250mm </Depth>
      <Speed> 7mm/s </Speed>
     </Material>
    </Drilling>
   </Activity>
   <!-- end of standard matching rules -->
  </ManufacturingActivity>
  <ManufacturingInformationExchange>
   <Processor id="111" level="3" mandatoryLevel="3">
    <RISC />
   </Processor>
   <OperatingSystem id="112" level="3" mandatoryLevel="2">
    <XP />
   </OperatingSystem>
  </ManufacturingInformationExchange>
 </ReferenceCapabilityClassStructure>
 <ReferenceCapabilityClassStructure id="rcs_3002">
  <ManufacturingActivity>
   <!-- for standard matching rules -->
   <Activity id="act_3001" level="7" mandatoryLevel="6">
    <ControlFunctionBlock>
    </ControlFunctionBlock>
   </Activity>
   <Activity id="act_3002" level="7" mandatoryLevel="6">
    <IOFunctionBlock>
    </IOFunctionBlock>
   </Activity>
   <!-- end of standard matching rules -->
  </ManufacturingActivity>
  <ManufacturingInformationExchange>
   <Processor id="115" level="3" mandatoryLevel="3">
    <ARM />
   </Processor>
   <OperatingSystem2 id="1" level="3" mandatoryLevel="2">
    <GenericPLC_OS />
```

```
    </OperatingSystem2>
   </ManufacturingInformationExchange>
  </ReferenceCapabilityClassStructure>
 </Specific>
</CapabilityProfile>
</CapabilityProfiling>
```

А.2.4 Приведение в соответствие требуемого профиля возможностей с одной ЕППО

При приведении в соответствие следует учитывать все элементы базового шаблона деятельности «сверление», указанной в А.2.3.

ЕППО предлагает сверление трех разных материалов с тремя разными характеристиками скорости и глубины. В соответствии с этим используют тип сверления «Copper» (медь). Оба атрибута скорости и глубины сверления меди соответствуют профилю ЕППО. Таким образом, по этому элементу существует полное совпадение.

Для достижения полного совпадения процесс приведения в соответствие должен быть продолжен по другим элементам.

А.3 Структура класса возможностей для тестового модуля

А.3.1 Образец эталонной структуры класса, использующий синтаксис языка XML

Примером образца справочного словаря для тестового модуля является следующий:

```
<?xml version="1.0"?>
<!-- edited with . . . -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="qualified" id="CapabilityProfiling">
 <xs:element name="CapabilityProfiling">
  <xs:complexType>
   <xs:sequence maxOccurs="unbounded">
    <xs:element name="type">
     <xs:complexType>
      <xs:attribute name="id" type="xs:string" form="unqualified"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="ReferenceCapabilityClassStructure">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="TestActivities">
        <xs:complexType>
         <xs:choice>
          <xs:element name="Importer">
           <xs:complexType>
            <xs:choice>
             <xs:element name="File">
```

```
<xs:complexType>
 <xs:sequence>
  <xs:element name="SingleFile" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="ASCII" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="FeatureS" minOccurs="0" maxOccurs="unbounded">
         <xs:complexType>
          <xs:sequence>
           <xs:element name="Feature" nillable="true" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
             <xs:simpleContent>
              <xs:extension base="xs:string"/>
             </xs:simpleContent>
            </xs:complexType>
           </xs:element>
          </xs:sequence>
         </xs:complexType>
        </xs:element>
       </xs:sequence>
       <xs:attribute name="level" type="xs:int" form="unqualified"/>
       <xs:attribute name="ID" type="xs:string" form="unqualified"/>
      </xs:complexType>
     </xs:element>
     <xs:element name="EDAS" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:attribute name="level" type="xs:int" form="unqualified"/>
       <xs:attribute name="ID" type="xs:string" form="unqualified"/>
      </xs:complexType>
     </xs:element>
     <xs:element name="Excel" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="Excel-Binary" minOccurs="0" maxOccurs="unbounded">
         <xs:complexType>
          <xs:attribute name="level" type="xs:int" form="unqualified"/>
          <xs:attribute name="ID" type="xs:string" form="unqualified"/>
         </xs:complexType>
        </xs:element>
        <xs:element name="Excel-Text" minOccurs="0" maxOccurs="unbounded">
         <xs:complexType>
          <xs:attribute name="level" type="xs:int" form="unqualified"/>
          <xs:attribute name="ID" type="xs:string" form="unqualified"/>
         </xs:complexType>
        </xs:element>
       </xs:sequence>
       <xs:attribute name="level" type="xs:int" form="unqualified"/>
       <xs:attribute name="ID" type="xs:string" form="unqualified"/>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
    <xs:attribute name="level" type="xs:int" form="unqualified"/>
    <xs:attribute name="ID" type="xs:string" form="unqualified"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="MultipleFile" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="Diadem" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:attribute name="level" type="xs:int" form="unqualified"/>
       <xs:attribute name="ID" type="xs:string" form="unqualified"/>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
    <xs:attribute name="level" type="xs:int" form="unqualified"/>
    <xs:attribute name="ID" type="xs:string" form="unqualified"/>
   </xs:complexType>
  </xs:element>
```

```
      <xs:element name="Folder" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="ISO13499" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
           <xs:attribute name="level" type="xs:int" form="unqualified"/>
           <xs:attribute name="ID" type="xs:string" form="unqualified"/>
          </xs:complexType>
         </xs:element>
        </xs:sequence>
        <xs:attribute name="level" type="xs:int" form="unqualified"/>
        <xs:attribute name="ID" type="xs:string" form="unqualified"/>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="Database">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="SQL" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
        <xs:attribute name="level" type="xs:int" form="unqualified"/>
        <xs:attribute name="ID" type="xs:string" form="unqualified"/>
       </xs:complexType>
      </xs:element>
      <xs:element name="ODS" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
        <xs:attribute name="level" type="xs:int" form="unqualified"/>
        <xs:attribute name="ID" type="xs:string" form="unqualified"/>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
  </xs:choice>
  <xs:attribute name="level" type="xs:int" form="unqualified"/>
  <xs:attribute name="ID" type="xs:string" form="unqualified"/>
 </xs:complexType>
</xs:element>
<xs:element name="Worker">
 <xs:complexType>
  <xs:choice>
   <xs:element name="Calculation">
    <xs:complexType>
     <xs:choice>
      <xs:element name="Formula">
       <xs:complexType>
        <xs:choice>
         <xs:element name="FormulaEditor">
          <xs:complexType>
           <xs:sequence>
            <xs:element name="ArithmetikOperators" minOccurs="0" maxOccurs="unbounded">
             <xs:complexType>
              <xs:attribute name="level" type="xs:int" form="unqualified"/>
              <xs:attribute name="ID" type="xs:string" form="unqualified"/>
             </xs:complexType>
            </xs:element>
            <xs:element name="ArithmeticFunctions" minOccurs="0" maxOccurs="unbounded">
             <xs:complexType>
              <xs:attribute name="level" type="xs:int" form="unqualified"/>
              <xs:attribute name="ID" type="xs:string" form="unqualified"/>
             </xs:complexType>
            </xs:element>
            <xs:element name="StatisticalFunctions" minOccurs="0" maxOccurs="unbounded">
             <xs:complexType>
              <xs:attribute name="level" type="xs:int" form="unqualified"/>
```

```
            <xs:attribute name="ID" type="xs:string" form="unqualified"/>
           </xs:complexType>
          </xs:element>
          <xs:element name="TrigonometricFunctions" minOccurs="0" maxOccurs="unbounded">
           <xs:complexType>
            <xs:attribute name="level" type="xs:int" form="unqualified"/>
            <xs:attribute name="ID" type="xs:string" form="unqualified"/>
           </xs:complexType>
          </xs:element>
          <xs:element name="HyperbolicFunctions" minOccurs="0" maxOccurs="unbounded">
           <xs:complexType>
            <xs:attribute name="level" type="xs:int" form="unqualified"/>
            <xs:attribute name="ID" type="xs:string" form="unqualified"/>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute name="level" type="xs:int" form="unqualified"/>
         <xs:attribute name="ID" type="xs:string" form="unqualified"/>
        </xs:complexType>
       </xs:element>
       <xs:element name="FixedFormula">
        <xs:complexType>
         <xs:attribute name="level" type="xs:int" form="unqualified"/>
         <xs:attribute name="ID" type="xs:string" form="unqualified"/>
        </xs:complexType>
       </xs:element>
      </xs:choice>
      <xs:attribute name="level" type="xs:int" form="unqualified"/>
      <xs:attribute name="ID" type="xs:string" form="unqualified"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="FrequenyAnalysis">
     <xs:complexType>
      <xs:choice>
       <xs:element name="FFT">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="TwoToTwoUnLimited" type="xs:string" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="level" type="xs:int" form="unqualified"/>
         <xs:attribute name="ID" type="xs:string" form="unqualified"/>
        </xs:complexType>
       </xs:element>
       <xs:element name="DFT">
        <xs:complexType>
         <xs:attribute name="level" type="xs:int" form="unqualified"/>
         <xs:attribute name="ID" type="xs:string" form="unqualified"/>
        </xs:complexType>
       </xs:element>
      </xs:choice>
      <xs:attribute name="level" type="xs:int" form="unqualified"/>
      <xs:attribute name="ID" type="xs:string" form="unqualified"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Filtering">
     <xs:complexType>
      <xs:choice>
       <xs:element name="FIR-Filter">
        <xs:complexType>
         <xs:attribute name="level" type="xs:int" form="unqualified"/>
         <xs:attribute name="ID" type="xs:string" form="unqualified"/>
        </xs:complexType>
       </xs:element>
       <xs:element name="FFT-Filter">
        <xs:complexType>
         <xs:choice>
          <xs:element name="CFC">
           <xs:complexType>
            <xs:attribute name="level" type="xs:int" form="unqualified"/>
            <xs:attribute name="ID" type="xs:string" form="unqualified"/>
```

```
                </xs:complexType>
               </xs:element>
               <xs:element name="xyz">
                <xs:complexType>
                 <xs:attribute name="level" type="xs:int" form="unqualified"/>
                 <xs:attribute name="ID" type="xs:string" form="unqualified"/>
                </xs:complexType>
               </xs:element>
              </xs:choice>
              <xs:attribute name="level" type="xs:int" form="unqualified"/>
              <xs:attribute name="ID" type="xs:string" form="unqualified"/>
             </xs:complexType>
            </xs:element>
           </xs:choice>
           <xs:attribute name="level" type="xs:int" form="unqualified"/>
           <xs:attribute name="ID" type="xs:string" form="unqualified"/>
          </xs:complexType>
         </xs:element>
         <xs:element name="Statistic">
          <xs:complexType>
           <xs:choice>
            <xs:element name="HIC">
             <xs:complexType>
              <xs:attribute name="level" type="xs:int" form="unqualified"/>
              <xs:attribute name="ID" type="xs:string" form="unqualified"/>
             </xs:complexType>
            </xs:element>
           </xs:choice>
           <xs:attribute name="level" type="xs:int" form="unqualified"/>
           <xs:attribute name="ID" type="xs:string" form="unqualified"/>
          </xs:complexType>
         </xs:element>
        </xs:choice>
        <xs:attribute name="level" type="xs:int" form="unqualified"/>
        <xs:attribute name="ID" type="xs:string" form="unqualified"/>
       </xs:complexType>
      </xs:element>
     </xs:choice>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="Exporter">
    <xs:complexType>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="Viewer">
    <xs:complexType>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
  </xs:choice>
  <xs:attribute name="level" type="xs:int" form="unqualified"/>
  <xs:attribute name="ID" type="xs:string" form="unqualified"/>
 </xs:complexType>
</xs:element>
<xs:element name="InformationExchange">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="InputDataTypes" type="DataTypes"/>
   <xs:element name="OutputDataTypes" type="DataTypes"/>
  </xs:sequence>
  <xs:attribute name="level" type="xs:int" form="unqualified"/>
  <xs:attribute name="ID" type="xs:string" form="unqualified"/>
 </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="name" type="xs:string" form="unqualified"/>
```

```
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:complexType name="DataTypes">
 <xs:sequence>
  <xs:element name="Numerical" minOccurs="0">
   <xs:complexType>
    <xs:sequence>
      <xs:element name="Zero-Dimensional" type="BasicNumeric"/>
      <xs:element name="One-Dimensional" type="BasicNumeric"/>
      <xs:element name="Two-Dimensional" type="BasicNumeric"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="MultiMedia" type="BasicMultiMedia" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="BasicNumeric">
  <xs:sequence>
   <xs:element name="double" minOccurs="0">
    <xs:complexType>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="int" minOccurs="0">
    <xs:complexType>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="string" minOccurs="0">
    <xs:complexType>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="time" minOccurs="0">
    <xs:complexType>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="BasicMultiMedia">
  <xs:sequence>
   <xs:element name="Video" minOccurs="0">
    <xs:complexType>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="Audio" minOccurs="0">
    <xs:complexType>
     <xs:attribute name="level" type="xs:int" form="unqualified"/>
     <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

А.3.2 Пример профиля возможностей требований

Примером профиля возможностей требований, использующего шаблон, приведенный в А.3.1, является следующий:

```
<?xml version="1.0" encoding="UTF-8"?>
<CapabilityProfiling>
  <Type id="Requirement Profile"/>
  <CapabilityProfile date="2003-09-11">
   <!-- ISO 8601 defined date yyyy-mm-dd -->
   <pkgtype version="V01.01.03a"/>
    <!--DTD-NAME= V<Application Profile Nr>.<Version Nr>.<Revision Nr>[<Patch Level>]-->
    <Common>
     <Requirement id="AMS-r701-0001" version="V01.01.01a"/>
     <Owner>
      <name>MSU User Ltd.</name>
      <street>Spring Blv.2</street>
      <city>Softcity</city>
      <zip>0915</zip>
      <state/>
      <country>France</country>
      <comment>Never seen before</comment>
     </Owner>
     <ComputingFacilities type="offered">
      <!-- see Fig 6 part 2 -->
      <Processor type="INTEL"/>
      <OperatingSystem>LINUX</OperatingSystem>
      <Language name="EN"/>
      <Memory size="28" unit="MB"/>
      <DiskSpace size="30" unit="GB"/>
    </ComputingFacilities>
    <Performance>
      <ElapsedTime value="200ms"/>
      <TransactionsPerSecond value="200"/>
    </Performance>
    <TemplateID ID_Number="Cea_A1_Evaluation_ISO-DIS16100-01" />
    <!-- gives the uppest level, relative root; e.g. to level 4 -->
    <CapabilityClassName id="ccn_1001"> Calculate_FFT </CapabilityClassName>
   </Common>
   <Specific>
    <ReferenceCapabilityClassStructure id="cea_1001" name="CEA_TestActivity"
version="001" url="">
      <TestActivities>
        <FormulaEditor level="4" ID="ABAAA" status="mandatory"/>
      </TestActivities>
      <InformationExchange>
        <InputDataTypes level="1" ID="BA">
          <DoubleValue level="4" ID="BAAAA" status="optional"/>
          <IntegerValue level="4" ID="BAAAB" status="optional"/>
          <DoubleVector level="4" ID="BAABA" status="mandatory"/>
          <IntegerVector level="4" ID="BAABB" status="optional"/>
        </InputDataTypes>
        <OutputDataTypes level="1" ID="BB">
          <Vector level="3" ID="BBAB" status="mandatory"/>
        </OutputDataTypes>
      </InformationExchange>
    </ReferenceCapabilityClassStructure>
   </Specific>
 </CapabilityProfile>
</CapabilityProfiling>
```

А.3.3 Пример профиля возможностей ЕППО

Примером описания профиля возможностей контрольного действия CEA_TestActivity, с помощью которого проводят анализ быстрого преобразования Фурье (FFT) в случае, когда на входе принимаются парные, а также целые значения является следующий. Результатом вычисления будут парные значения.

```
<?xml version="1.0" encoding="UTF-8"?>
<CapabilityProfiling>
  <Type id="Capability Profile"/>
  <CapabilityProfile date="2003-11-13">
    <!-- ISO 8601 defined date yyyy-mm-dd -->
    <pkgtype version="V01.01.01a"/>
    <!--DTD-NAME= V<Application Profile Nr>.<Version Nr>.<Revision Nr[<Patch Level>]-->
    <Common>
      <MSU_Capability id="AMS-101-0001" name="SampleCeaWorker" version="V01.02.01a" uri=""/>
      <Owner>
        <name>MSU Developer Inc.</name>
        <street>Winter Ave.7</street>
        <city>Softcity</city>
        <zip>4712</zip>
        <state>CA</state>
        <country>USA</country>
        <comment>Only best experiences!</comment>
      </Owner>
        <ComputingFacilities type="required"> <!-- see Fig 6 part 2 -->
           <Processor type="INTEL"/>
           <OperatingSystemS>
             <OperatingSystem>LINUX</OperatingSystem>
             <OperatingSystem>JAVA</OperatingSystem>
           </OperatingSystemS>
           <Language name="EN"/>
           <Memory size="28" unit="MB"/>
           <DiskSpace size="30" unit="GB"/>
        </ComputingFacilities>
        <Performance>
           <ElapsedTime value="61ms"/>
           <TransactionsPerSecond value="621"/>
        </Performance>
        <TemplateID ID_Number="Cea_A1_Evaluation_ISO-DIS16100-01" />
        <!-- gives the uppest level, relative root; e.g. to level 4-->
        <CapabilityClassName id="ccn_1001"> Calculate_FFT </CapabilityClassName>
    </Common>
    <Specific>
        <ReferenceCapabilityClassStructure id="cea_1001" name="CEA_TestActivity" version="001" url="">
           <TestActivities>
             <Worker>
               <Calculation>
                 <FrequencyAnalysis>
                  <FFT/>
                 </FrequencyAnalysis>
               </Calculation>
             </Worker>
           </TestActivities>
           <InformationExchange>
             <InputDataTypes level="1" ID="BA">
               <Numerical level="2" ID="BAA">
                 <One-Dimensional level="3" ID="BAAA">
                  <double level="3" ID="BAAAA"/>
                  <int level="3" ID="BAAAB"/>
                 </One-Dimensional>
               </Numerical>
             </InputDataTypes>
             <OutputDataTypeslevel="1" ID="BB">
               <Numerical level="2" ID="BAA">
                 <One-Dimensional level="3" ID="BAAA">
```

```
                <double level="3" ID="BAAAA"/>
              </One-Dimensional>
            </Numerical>
          </OutputDataTypes>
        </InformationExchange>
      </ReferenceCapabilityClassStructure>
    </Specific>
  </CapabilityProfile>
</CapabilityProfiling>
```

Приложение В
(справочное)

Эталонные модели возможностей

В.1 Диаграмма класса возможностей и модель объекта

Модель использования и обмена информацией производственного процесса должна включать в себя объектную модель классов, используемых в технологическом процессе. Объектная модель должна характеризовать данные и функции, используемые в процессе проектирования и производства. Главной целью разработки модели является облегчение разработки спецификаций на информационное содержание. Эти условия необходимы для профилирования возможностей интероперабельности программных средств разных поставщиков.

Например, для реального совместного проектирования и производства, информационные представления проектирования изделия и производственных процессов должны поддерживать многочисленные уровни абстракции для обеспечения двунаправленного (или многонаправленного) обмена информацией между прикладными процессами. В процессе концептуальной фазы проектирования изделия важно установить все возможные компромиссы и последствия принятых проектных решений на высшем уровне. Описания проектов изделия, для которых не определены геометрические принципы, могут предоставить достаточное количество входных данных для определения характеристик производственного процесса и оценки базовой стоимости. Формальное представление таких ранних проектных описаний может обеспечить ввод данных в концептуальное планирование процесса и производственные приложения. Формальные представления могут быть использованы для разработки технических условий на интерфейсы для интеграции проектирования изделия и производственной технологической деятельности.

Класс представляет собой информационную конструкцию, необходимую для представления концепции или физического объекта. Он состоит из атрибутов и функций. Класс может дополнять другой класс атрибутами класса. Информационная

модель включает в себя следующие ключевые слова: класс, расширение, строка, целое число и пара.

Информационный обмен между проектированием изделия и прикладными программами планирования производственного процесса, а также другими приложениями существует более чем на одной стадии производства. На рисунке В.1 представлены стадии коммуникации, которые могут существовать при установлении возможности интероперабельности между концептуальным проектированием и концептуальным планированием технологического процесса.

Концептуальное проектирование представляет собой действие на ранней проектной стадии, на которой формулируется концепция изделия. Концепция изделия включает в себя требования к изделию, его возможные линии поведения, форму/структуру (схему расположения) и ассоциативные свойства.

Рисунок В.1 – Иерархия или формирование гнездовой структуры действий производственного процесса

Свойства включают в себя материал, допустимые отклонения на этапе сборки, критическую шероховатость поверхности и параметры твердости, а также критические размеры. Функциональное проектирование изделия определяет главные функции изделия и разбивает эти функции на более простые в соответствии с установленными требованиями. Проектирование поведения изделия преобразовывает функции в линии поведения. Проектирование конструктивного исполнения определяет требования к форме и структуре изделия на основе функций и линий поведения. Подробную информацию об изделии

определяют в процессе концептуального проектирования. Концептуальную проектную информацию использует в процессе разработки рабочего проекта, например конфигурации, топологии, допустимых отклонений и размерной спецификации.

Концептуальное проектирование разбивают на пять составных действий А11 – А15 согласно рисунку В.2. Действие А11 предназначено для определения функций и ограничений, которые определяют на основе входных данных и технических требований. Это действие называется функциональным проектированием. Действие А12 предназначено для создания линии поведения изделия на основе его функций и ограничений, указанных как выходные данные дейаствия А11. Это действие предназначено для создания линий поведения. Для тех изделий, которые не имеют линий поведения, например таких, как структурные статические объекты, это действие следует пропустить. Действие А13 предназначено для разбиения функций и ограничений на составные части так, что каждая часть, сборочный узел и изделие в сборе имеют свои собственные функции, ограничения и, в приемлемых случаях, линии поведения. При разбиении функций, ограничений и линий поведения составные части могут быть спроектированы и изделие может быть сконфигурировано по этим частям в А14. Действие А14 может быть разбито на два составных действия: А141, заключающееся в установлении структуры изделия на основе его функции, и А142, заключающееся в указании подробной информации об артефакте (искусственном объекте), который будут производить.

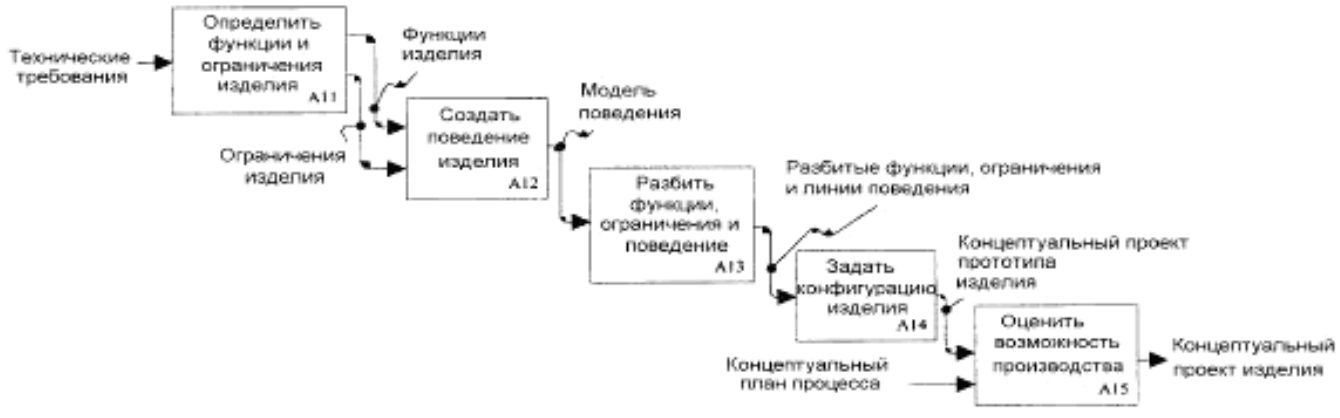

Рисунок В.2 – Функциональное разбиение на составные части концептуального проектирования

Выходные данные действия A14 составляют концепцию изделия, которая является концептуальным проектом изделия. Действие A15 предназначено для оценки концептуального проекта на возможность производства изделия. Для анализа возможности производства используют входные данные, полученные в результате концептуального планирования процесса и концептуального проектирования. При этом определяют форму и структуру изделия. Полученная подробная информация является необходимой для планирования производства, например для выбора производственного процесса, ресурса и оценки стоимости.

Модель деятельности характеризует функции и их входные и выходные данные в концептуальном проекте на основе комплексного планирования проекта и процесса. Модель деятельности определяет контекст, в рамках которого разрабатывается модель объекта.

Модель объекта содержит многочисленные аспекты концептуального проекта, описанного в модели действия. Характеристики изделия в целом, его функции и взаимосвязи при сборке приведены далее. Классы в отношении требований к изделию не описывают, так как концептуальное проектирование начинают с создания функционального проекта. Модель представляют с использованием диаграмм классов унифицированного языка моделирования UML. Концепции следующих классов заимствуют из ядра: артефакт (Artefact), функция (Function) и ограничение (Constraint). Следующие классы выводят из ядра: поведение (Behaviour), материал (Material), требование (Requirement) и ограничение (Constraint). Остальные классы указывают в концептуальном проекте, объединенном с концептуальным планированием технологического процесса. Данную модель используют в качестве:

- эталонной архитектуры концептуального проектирования следующего поколения изделий;

- основы для разработки стандартных интерфейсов;

- схемы базы данных.

Артефакт является общим термином для ссылки на индивидуальный компонент в иерархической структуре изделия, например часть, сборочный узел

или изделие. Специальная информация об артефакте включает в себя параметры материала, допустимые отклонения, размеры, а также параметры шероховатости поверхности, твердости поверхности и текстуры.

Диаграмма класса искусственного объекта (артефакта) приведена на рисунке В.3. Класс артефакта имеет рекурсивное определение. В иерархической структуре присутствуют артефакты, которые определяют структуру изделия в иерархической структуре. Артефакт имеет два производных класса: ArtefactToBeMade (артефакт, который следует изготовить) и ArtefactToBeBought (артефакт, который следует купить). Некоторые компоненты (artifact`s) могут быть приобретены у поставщика, другие изготавливают на предприятии, на котором разрабатывается конкретное изделие. ArtefactToBeMade имеет последовательность производственных технологических процессов. ArtefactToBeBought может быть получен от поставщика. Класс поставщика Supplier определяет информационное описание поставщика.

Артефакт имеет следующие ассоциативные классы: EngineeringRequirement (техническое требование), ArtefactConstraint (ограничение артефакта), Fuction (функция), Behaviour (поведение), Form (форма), Material (материал), SurfaceCondition (состояние поверхности) и Tolerance (допустимое отклонение).

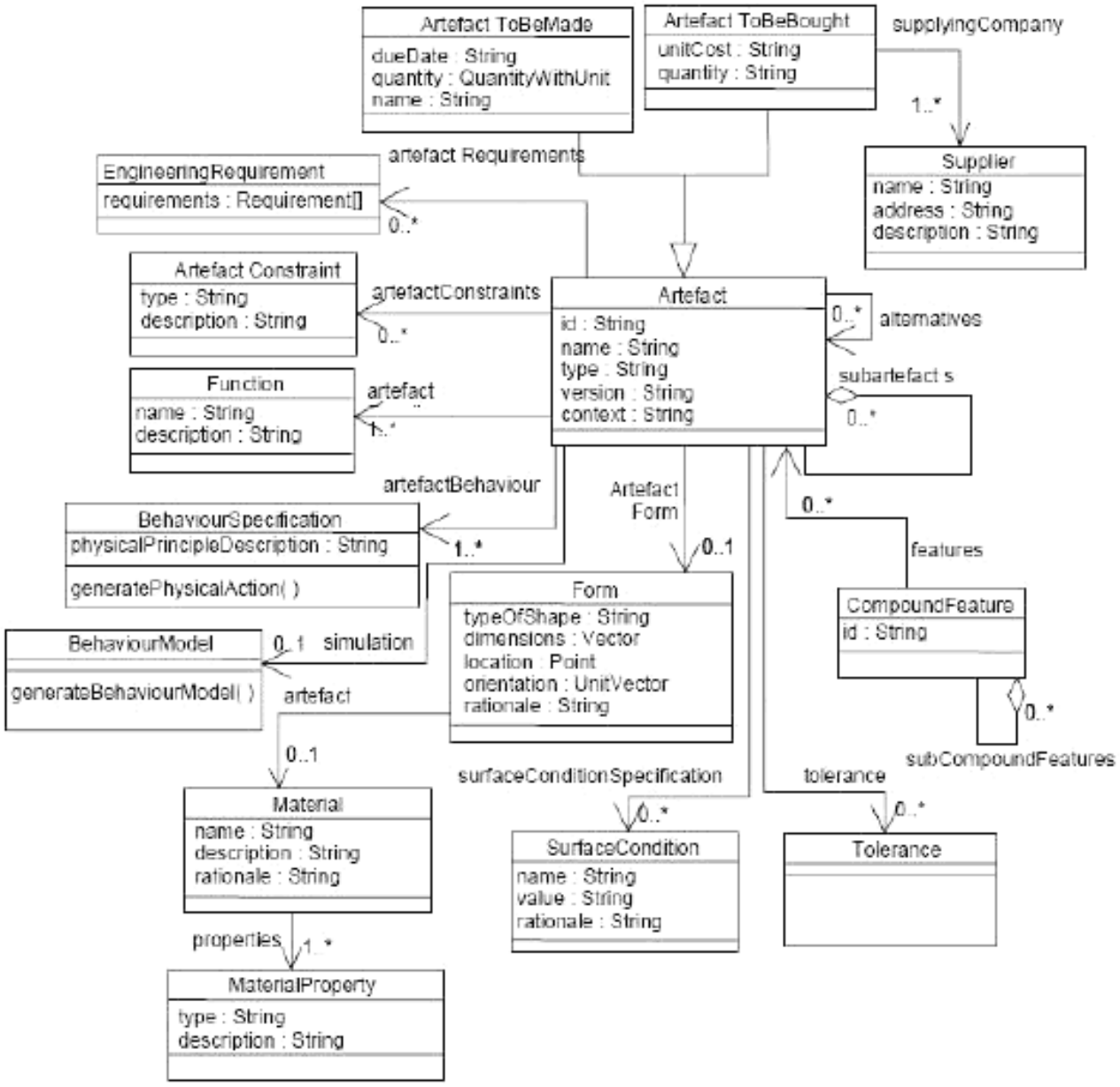

Рисунок В.3 – Диаграмма класса Artefact

Класс EngineeringRequirement (техническое требование) включает в себя требования к технологии, например, требования к перемещению, потребности в электроэнергии, а также требования к допустимым отклонениям.

Класс ArtefactConstraint (ограничение артефакта) имеет отношение к тем факторам, которые устанавливают ограничения к функционированию изделия, например, физические ограничения, охрана окружающей среды и правила техники безопасности.

Класс Tolerance (допустимое отклонение) определяет предельные значения, при которых свойство может изменяться. Tolerance включает в себя допустимые отклонения формы, местоположения, профиля и т. д. Диаграмма класса допустимого отклонения приведена на рисунке В.4. Логическое обоснование включает в себя причины, по которым устанавливают допуск, выбирают материал, разрабатывают форму и свойства. Класс Tolerance состоит из трех подклассов: ShapeTolerance (допуск на форму), PositionTolerance (допуск на местоположение), RunoutTolerance (допуск на биение).

Класс ShapeTolerance состоит из двух подклассов: ProfileTolerance (допустимое отклонение профиля) и FormTolerance (допустимое отклонение формы).

Класс PositionTolerance состоит из двух подклассов: LocationTolerance (допуск на местоположение) и OrientationTolerance (допуск на ориентацию).

Класс Function объединяет функции изделия, полученные на основе технических требований. Функция связана с назначением или целью изделия и является преобразованием между входом и выходом, т. е. энергией, материалом и информационным потоком. Функция передачи согласно рисунку В.5 является производным классом от Function. Она включает в себя передаточные функции, имеющие ввод и вывод информации, материала и/или энергии. Ввод имеет тип входных данных, количество и источник. Вывод имеет тип, количество и пункт назначения. Функции изготовленных частей изменяются и должны быть ограничены AllowableVariation (допустимым изменением), которое устанавливает допустимые предельные отклонения функции. Функциональное допустимое отклонение может быть преобразовано в допустимые отклонения определенных параметров свойства с помощью, например, методов Taguchi.

Класс Form (форма) имеет концептуальную или изображенную в виде схемы конфигурацию изделия и его компонентов. Форма характеризует тип конфигурации, размер, местоположение, ориентацию и материал артефакта. Свойство – это часть формы в рекурсивном определении. Свойство может состоять из составных свойств, которые есть в иерархической структуре. Например, винтовое отверстие состоит из свойства зенковки и свойства отверстия, а свойство отверстия – из

цилиндрического свойства и свойства резьбы. Этот класс может также представлять сложные свойства, составные свойства разных артефактов.

Класс Material (материал) представляет информацию свойств материалов, имеющих отношение к реализации конфигурации изделия.

Класс SurfaceCondition (состояние поверхности) устанавливает требования к твердости, шероховатости и текстуре изделия и эталонным данным.

Класс Behaviour (рисунок В.6) охватывает перемещение изделия. Перемещение определяют или имитируют в этом классе с использованием метода generateBehaviourModel (создать модель поведения). Поведение изделия – это физические действия объекта в определенных условиях, являющихся входными усилиями и потоками артефакта, например, воздействующими внешними силами. Физические действия означают выходные усилия и потоки артефакта, например, перемещение или деформацию артефакта при воздействии внешних сил. Поведение может характеризоваться его атрибутами, переменными ввода и вывода линии поведения.

Поведенческая спецификация (проект) – это процесс отображения данных функций изделия в линии поведения изделия. Поведение может быть отображено в форме на основе физических принципов и предварительных знаний. Класс BehaviourSpecification (спецификация поведения) имеет следующие атрибуты: physicalPrincipleDescription (описание физических принципов), inputflowVariable (переменная входного потока), inputEffortVariable (переменная входного усилия), outputflowVariable (переменная выходного потока), OutputEffortVariable (переменная выходного усилия). GeneratePhysicalAction (создать физическое действие) – это метод класса поведения для создания физического действия. Класс InputFlow (входной поток) имеет четыре атрибута: inputFlowName (имя входного потока), inputFlowType (тип входного потока), inputFlowDirectiom (направление входного потока), inputFlowPosition (позиция входного потока). Класс InputEffort (входное усилие) имеет четыре аттрибута: inputEffortName (имя входного усилия), inputEffortType (тип входного усилия), inputEffortDirection (направление входного усилия), inputEffortPosition (позиция входного усилия). Класс OutputFlow (выходной

поток) имеет четыре атрибута: outputFlowName (имя выходного потока), outputFlowType (тип выходного потока), outputFlowDirection (направление выходного потока), outputFlowPosition (позиция выходного потока). Класс OutputEffort (выходное усилие) имеет четыре аттрибута: OutputEffortName (имя выходного усилия), OutputEffortType (тип выходного усилия), OutputEffortDirection (направление выходного усилия), OutputEffortPosition (позиция выходного усилия).

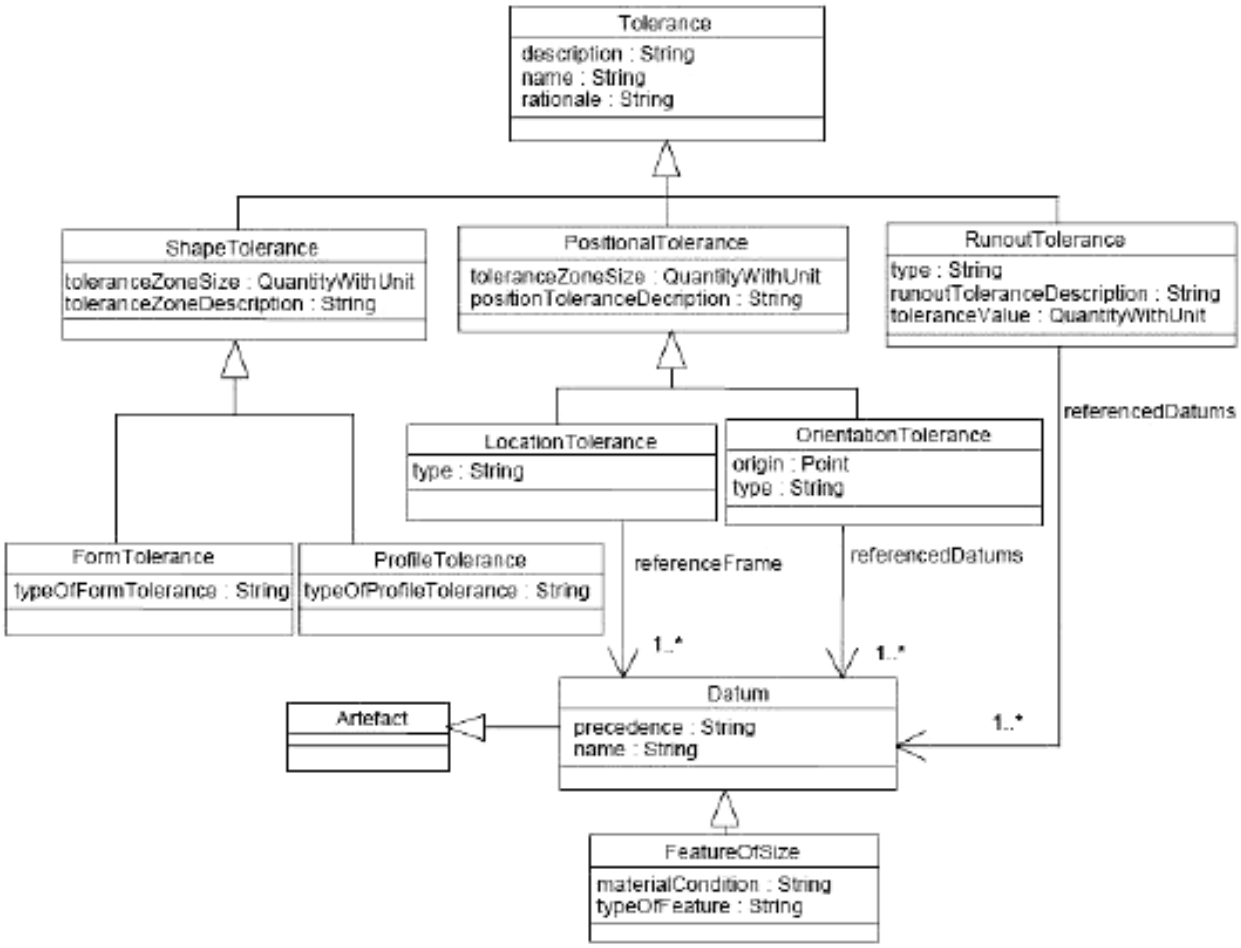

Рисунок В.4 – Диаграмма класса Tolerance

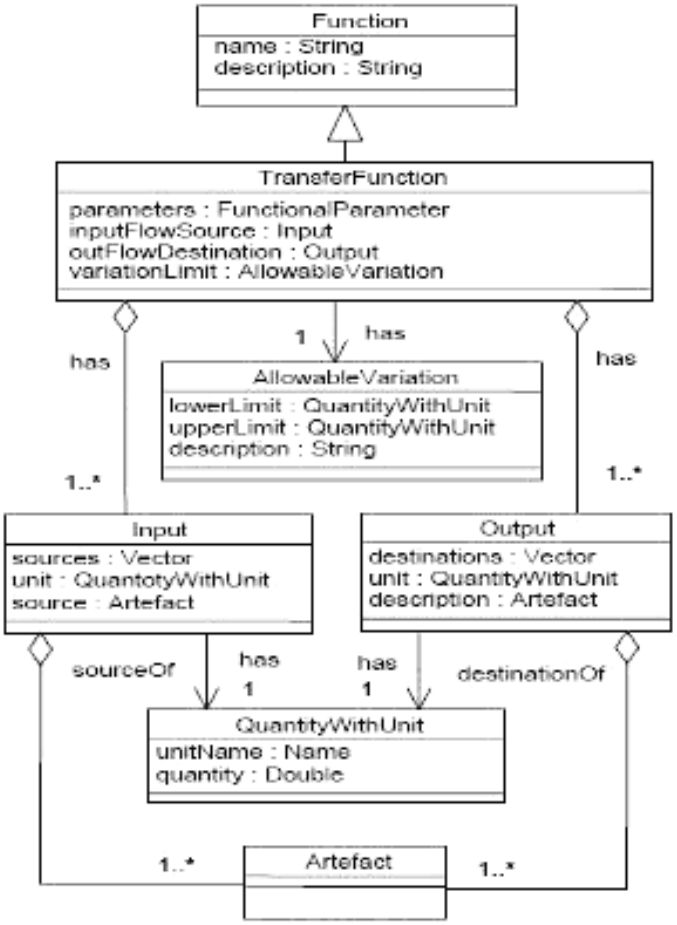

Рисунок В.5 – Диаграмма класса Function

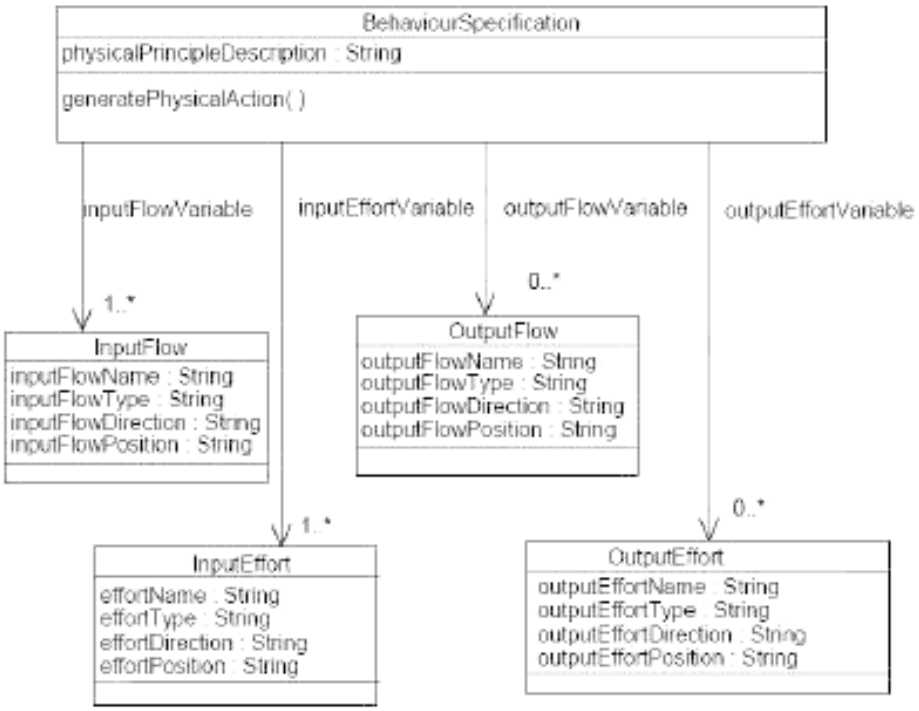

Рисунок В.6 – Диаграмма класса Behaviour

Модель соединения артефакта характеризует процесс сборки двух искусственных изделий согласно рисунку В.7. ArtefactJoint (сборка артефактов) описывает взаимосвязь двух артефактов. Этот класс имеет рекурсивное определение и формирует иерархию сборки. Чем ниже пара в иерархии, тем раньше следует проводить ее сборку. Таким образом, иерархия указывает последовательность сборки. Каждую пару собранных артефактов соединяют с другой парой с теми же самыми соотношениями и свойствами артефактов. FeatureJoint (соединение свойств) определяет при сборке взаимоотношения двух свойств и тип соединения (жесткий или кинематический). Этот класс также имеет рекурсивную структуру, обеспечивающую представление характеристик взаимоотношения свойств. Подобно классу ArtefactJoint последовательность особенности соединения/сборки представляют в виде иерархической структуры.

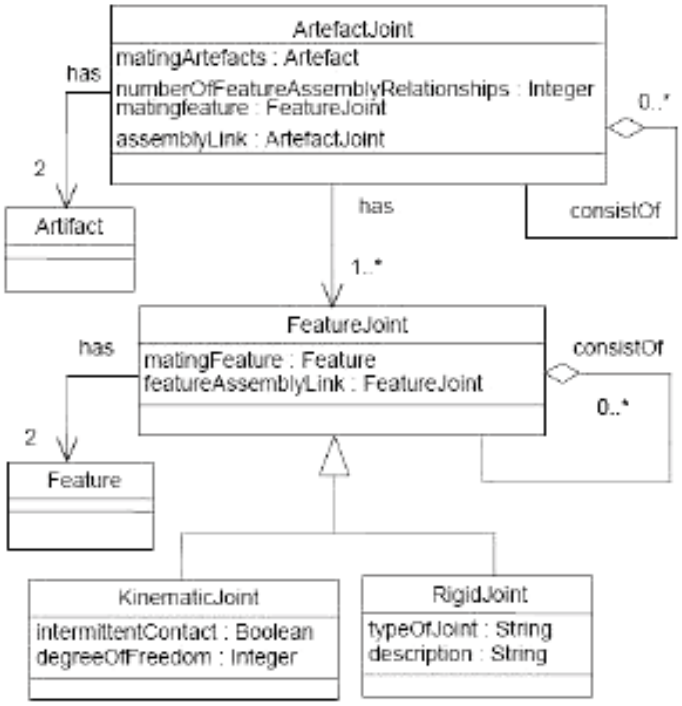

Рисунок В.7 – Диаграмма класса Joining

В.2 Диаграмма сотрудничества по возможностям программных средств

Концептуальное планирование производственного процесса представляет собой действие по выбору предварительных производственных процессов,

исходной финансовой оценки возможности производства и приблизительного расчета стоимости концептуального проекта. Данные, собранные в процессе концептуального планирования технологического процесса, используют при планировании подробного технологического процесса, например последовательности операций, параметров процесса и спецификации набора заданных значений.

В данном случае информационную модель концептуального проектирования создают с целью облегчения концептуального планирования технологического процесса. Чтобы установить контекст, в котором используется определенная информация, разрабатывают модель действий концептуального планирования технологического процесса. Эта модель разбивает главную идею на более подробные уровни.

Концептуальное планирование процесса (CPP) – это действие по выбору предварительных производственных процессов, оценке возможности производства и расчету стоимости концептуального проекта на ранней стадии проектирования изделия. Это действие направлено на установление производственных процессов, выбор ресурсов и оборудования, оценку производственных расходов и времени. Цель концептуального планирования процесса заключается в поддержке проекта изделия в отношении оптимальной формы изделия, конфигурации и выбора материалов с целью сведения к минимуму производственных затрат.

Одним из действий концептуального планирования процесса является планирование подробного технологического процесса. В отличие от CPP, планирование подробного технологического процесса является действием, основанным на рабочем проекте и результатах концептуального планирования процесса с целью установления последовательности операций, выбора станков и оснастки, которые следует использовать, определения настроечных установок и параметров процесса и продолжительности процесса и производственных расходов. На основе CPP разрабатывают модель действия с использованием методологии IDEFO с целью дать подробное описание функций и потока данных в действии CPP.

Действие по концептуальному планированию процесса А2 (действие А1 является концептуальным проектированием) состоит из трех составных действий А21 – А23 (см. рисунок В.8). Действие А21 предназначено для определения производственных процессов. В зависимости от концептуальной информации об изделии, например материале, форме, структуре и допустимых отклонениях, выбирают начальные производственные процессы, например такие, как литье, ковка, формовка и механическая обработка на станках. Это действие также включает в себя составную последовательность процессов завершения производства изделия.

Действие А22 предназначено для выбора производственных ресурсов. На основе отобранных производственных ресурсов выбирают подходящие производственные ресурсы, включая физические и трудовые ресурсы. В состав ресурсов входят станки, инструментальная оснастка и квалификация рабочих. Действие А23 предназначено для оценки стоимости производства. В стоимость входят материалы, закупаемые части, труд, использование инструментов, капитал, использование станков и накладные расходы.

Действие А22 охватывает ряд функций выбора ресурсов и может быть разбито на три составных действия, указанных на рисунке В.9. Действие А221 предназначено для выбора станков, отбираемых из станочного парка предприятия в соответствии с требованиями производственного процесса для производства изделия. Машины включают в себя оснастку, ковочные машины, оборудование литейного производства, обращение с материалами, сборочные машины и измерительные устройства. Действие А222 предназначено для выбора инструментальной оснастки и зажимов и осуществляется на основе отобранных станков. Выбирают оснастку и зажимы, необходимые для поддержки выбранных производственных процессов. Действие А223 предназначено для выбора квалифицированных рабочих и осуществляется на основе выбора станков и инструментальной оснастки. Квалификация рабочих должна соответствовать требованиям, предъявляемым к работе на станках и использованию инструментов в процессе производства.

Рисунок В.8 – Функциональное разбиение концептуального планирования процесса

Рисунок В.9 – Выбор производственных ресурсов

Модель объекта содержит информацию о потоке данных в процессе вышеуказанного концептуального планирования процесса. Характеристики модели процессов, ресурсов и структуры затрат на концептуальный проект приведены далее. Эту модель представляют с использованием диаграмм классов на языке моделирования UML.

Классы производственных процессов указаны на рисунке В.10. ArtefactToBeMade имеет последовательность объектов производственного процесса ManufacturingProcess. Производственный процесс ManufacturingProcess является процессом, характерным для определенного класса. Класс

производственной деятельности имеет рекурсивное определение, которое образует иерархическую структуру. Уровнями этой иерархии могут быть маршрутизация, рабочая станция, операция, шаг и свойство. Тип производственной деятельности указывает на определенный уровень иерархии. Производственная деятельность имеет следующие атрибуты: параметры производства, ресурсы производства, расчетные затраты/время, альтернативы, ветвление и соединение. Ветвление и соединение используют вместе для создания структуры совпадающих и параллельных действий. Специализация первого уровня производственной деятельности включает в себя настройку (Setup), манипулирование (Handling), обработку (Processing), загрузку/выгрузку (LoadUnload) и холостой ход (Idling). Настройка (Setup) представляет собой действия по регулировке станка, подготовке инструментальной оснастки или установки рабочей станции (АРМ). Манипулирование (Handling) включает в себя информацию о манипуляциях с материалами, включая передачу материалов или инструментов из одного места в другое. Загрузка/выгрузка (Load/Unload) представляют собой действия по загрузке и выгрузке обрабатываемой детали или инструментов в машине. Холостой ход (Idling) представляет собой продолжительность простоя между двумя действиями. Обработка (Processing) представляет собой производственную деятельность, которая включает в себя инспекцию (Inspection), изготовление частей (PartMaking) и сборку (Assembly).

Изготовление частей (PartMaking) представляет собой действие по изготовлению детали и может быть специализировано далее. Производными классами PartMaking являются SurfaceTreatment (обработка поверхности) и Shaping (придание формы). Класс SurfaceTreatment является действием по обработке поверхностей обрабатываемой детали в пределах допустимых отклонений, удовлетворению технических требований к шероховатости и твердости поверхности. Тремя производными классами SurfaceTreatment являются Finishing (чистовая отделка), Coating (покрытие) и Hardening (закалка). Класс Shaping представляет собой действие по трансформации обрабатываемой детали в определенную форму. Этот класс имеет два производных класса: MaterialRemoving (удаление материала) и Forming (формование). Класс Forming включает в себя формовку, литье или ковку материала в определенную форму. Класс

MateriakRemoving является действием по созданию формы, в результате чего материал удаляют с обрабатываемой детали для получения требуемой формы. Это может быть химическое удаление (ChemicalRemoving), термическое удаление (ThermalRemoving) или механическое удаление (MechanicalRemoving). Химическое удаление может быть фотохимическим дроблением, электрохимической обработкой (ECM) или химическим дроблением. Термическое удаление может представлять собой обработку лучом высокой энергии, электрическим разрядом (EDM) или с помощью газового резака.

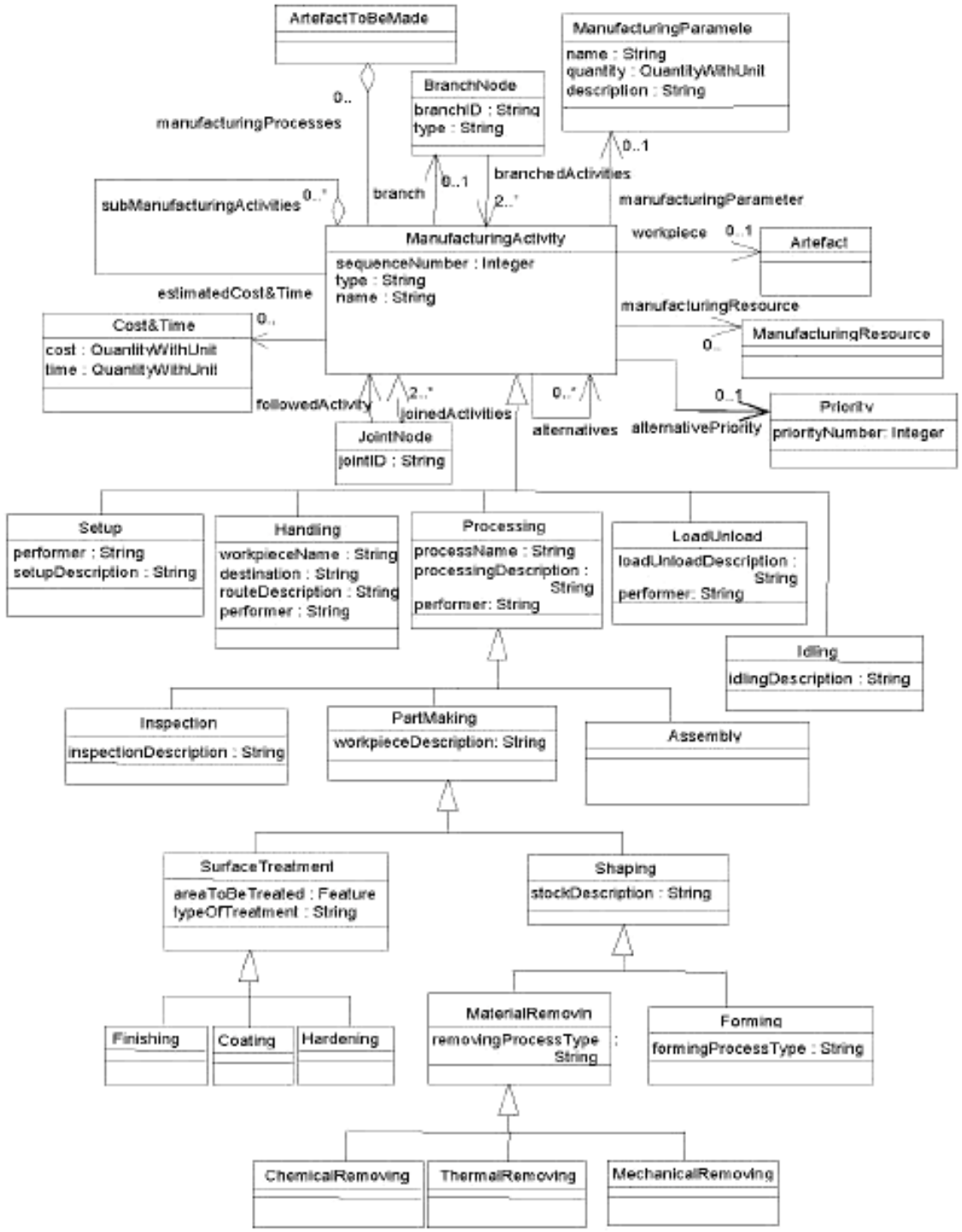

Рисунок В.10 – Диаграмма класса ManufacturingProcess

Классы механического удаления указаны на рисунке В.11. Класс MechanicalRemoving представляет собой действие по удалению лишнего материала с обрабатываемой детали с использованием механических методов, например обработку на станке или резку. Существуют два производных класса:

разделка листа (SheetSeparating) и обработка на станке (Mashining). При разделке листа (SheetSeparating) используют методы резки металлического листа. Специальными методами являются прокол, перфорирование и строгание. Обработка на станке (Mashining) является действием разрезания металла с помощью резца. Этот класс включает в себя три производных класса: одноточечная резка (SinglePointCutting), многоточечная резка (MultiplePointCutting) и абразивная резка (Abrasive). Одноточечная резка (SinglePointCutting) включает в себя расточку, токарную обработку, проточку канавок и нарезку резьбы. Многоточечная резка (MultiplePointCutting) включает в себя сверление, развертывание, фрезеровку, многоточечную нарезку резьбы, прошивку, зубонарезку и распилку. Абразивная резка (Abrasive) включает в себя шлифование, хонингование, обработку ультразвуком, чистовое шлифование и притирку.

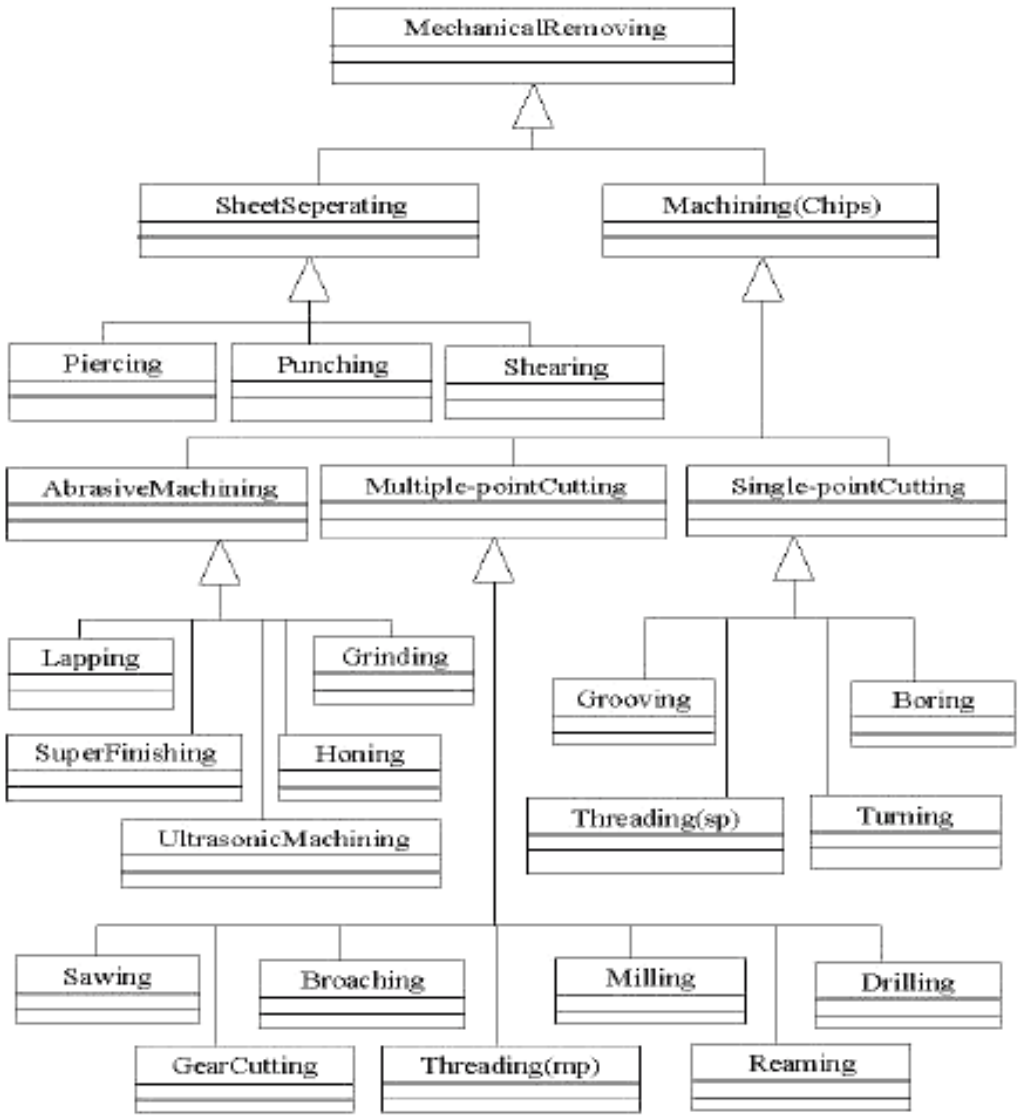

Рисунок В.11 – Объектная модель процесса MechanicalRemoving

Классы формовки приведены на рисунке В.12. Класс придание формы (Forming) включает в себя два производных класса: деформацию (Deformation) и застывание (Consolidation). Деформация (Deformation) является классом, который представляет собой действие по изменению формы обрабатываемой детали. Специальными процессами являются ковка, выдавливание, волочение, прокатка, накатка и зубофрезерование. Застывание (Consolidation) является классом, который представляет собой действие по приданию детали формы в пресс-форме или с помощью штампа. Специальными процессами являются отливка, полимерное прессование, керамическое прессование, уплотнение и изготовление слоистых материалов.

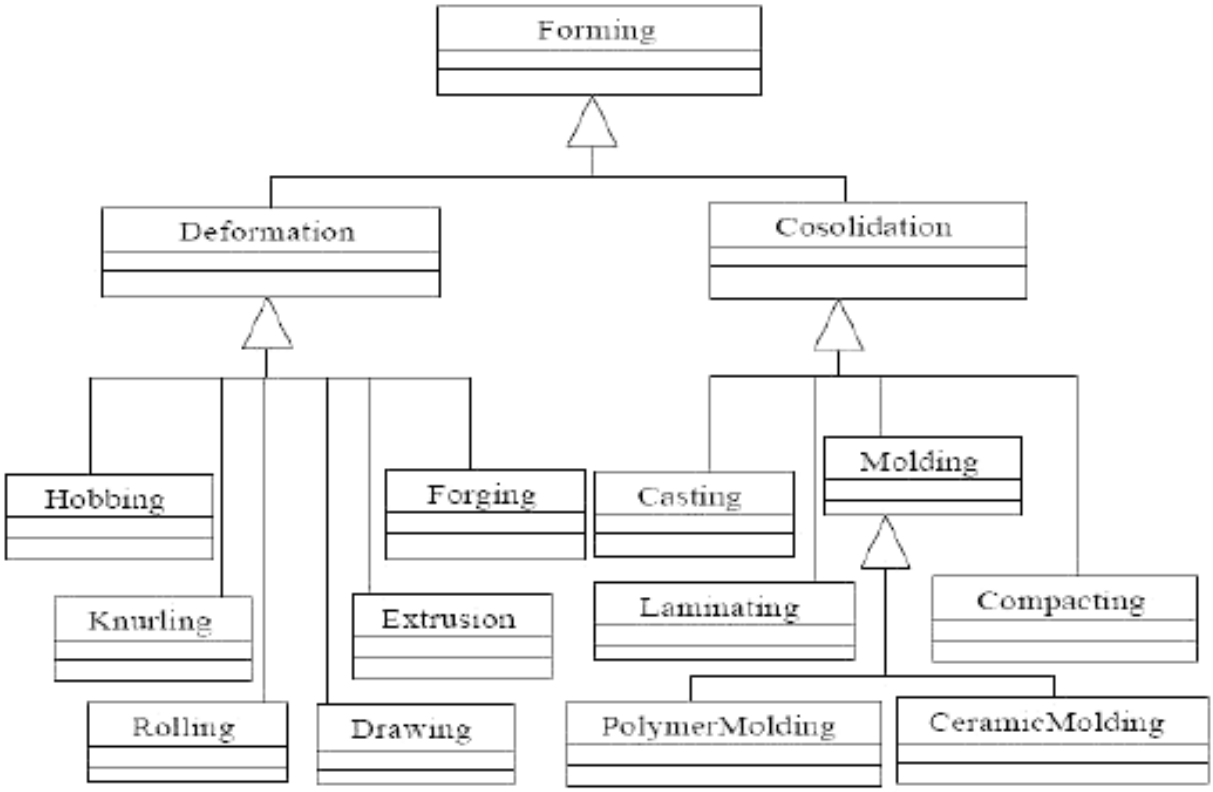

Рисунок В.12 – Модель процесса придания формы

AssemblyProcess (процесс сборки) (см. рисунок В.13) является классом, который представляет собой действие по сборке компонентов в изделие. Производными классами являются MechanicalJoining (механическое соединение), ThermalJoining (термическое соединение) и ChemicalJoining (химическое соединение). ChemicalJoining является классом, который представляет собой действие по соединению компонентов с использованием химических методов, например адгезионного склеивания. ThermalJoining является классом, который представляет собой действие по соединению компонентов с использованием методов термической связи, например пайку твердым припоем, термическую сварку и пайку мягким припоем. MechanicalJoining является классом, который представляет собой действие по соединению компонентов с использованием механических методов, например посадку, крепление и механическую сварку. Посадка может быть свободной подгонкой двух компонентов изделия или путем приложения усилия. Крепление может быть заклепочным или с использованием

замков, шпилек или винтов. Механическая сварка включает в себя сварку компонентов трением, взрывом, ультразвуком и давлением.

ManufacturingResource (производственные ресурсы) (см. рисунок В.14), являются классом, который представляет собой физический объект или квалифицированный труд, используемый в производственном процессе. ManufacturingEquipment (производственное оборудование) является классом, который представляет собой компонент оборудования, используемый в процессе производственной обработки. Примерами являются машины, инструменты, зажимы и измерительные устройства. Компонент оборудования имеет набор параметров, которые его характеризуют. EquipmentParameter (параметр оборудования) является классом, который представляет собой параметры. Производными классами ManufacturingEquipment являются Machine (машина), ToolForMachining (инструмент для машинной обработки), Mold (пресс-форма) и Die (штамп). Машина может быть обрабатывающим центром, ковочным механизмом, аппаратом EDM и т.д. Инструменты для машинной обработки (ToolForMachining) являются инструментальной оснасткой, используемой в процессе машинной обработки, например резец, удлинитель, держатель или измерительное устройство.

Концептуальное планирование процесса (Conceptional Process Planning) представляет собой определение возможности производства, которое включает в себя выбор производственных процессов на основе концептуального проекта и производственных ресурсов, а также расчет производственных затрат и времени. Включение в проект производственного анализа может гарантировать, что этот проект пригоден для производства изделия в рамках расчетных затрат. Анализ возможности производства и расчет затрат являются важными для минимизации себестоимости. Однако в настоящее время существует еще недостаточно программных средств для концептуального планирования технологического процесса. Производители нуждаются в создании новых программных средств, которые позволят эффективно поддерживать перевод концептуального проекта в производственный процесс и выбор ресурсов для оценки производственного времени и затрат. Для разработки новых программных средств необходимы

информационные модели, предназначенные для поддержки таких разработок и интеграции программных средств.

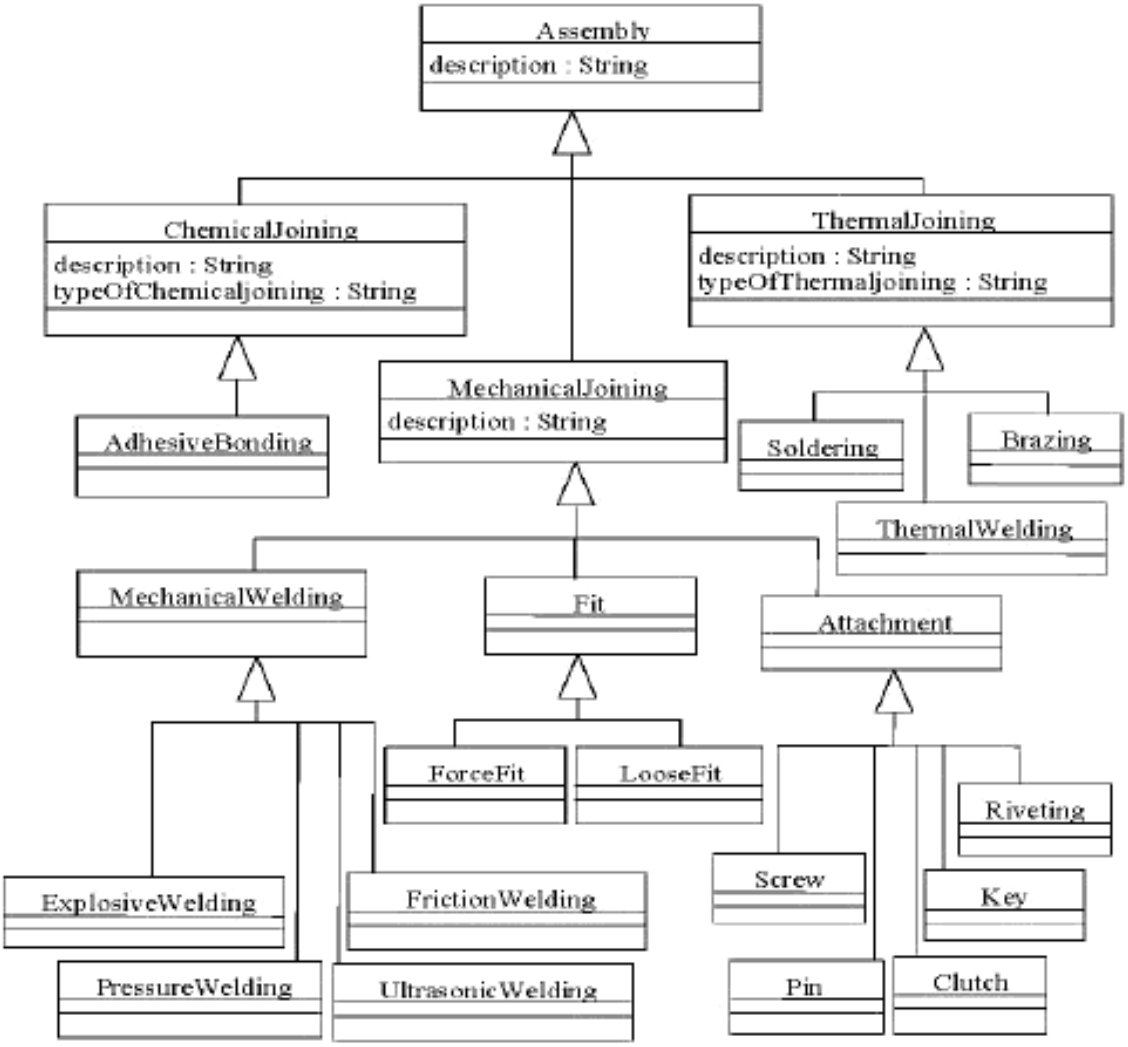

Рисунок В.13 – Объектная модель AssemblyProcess

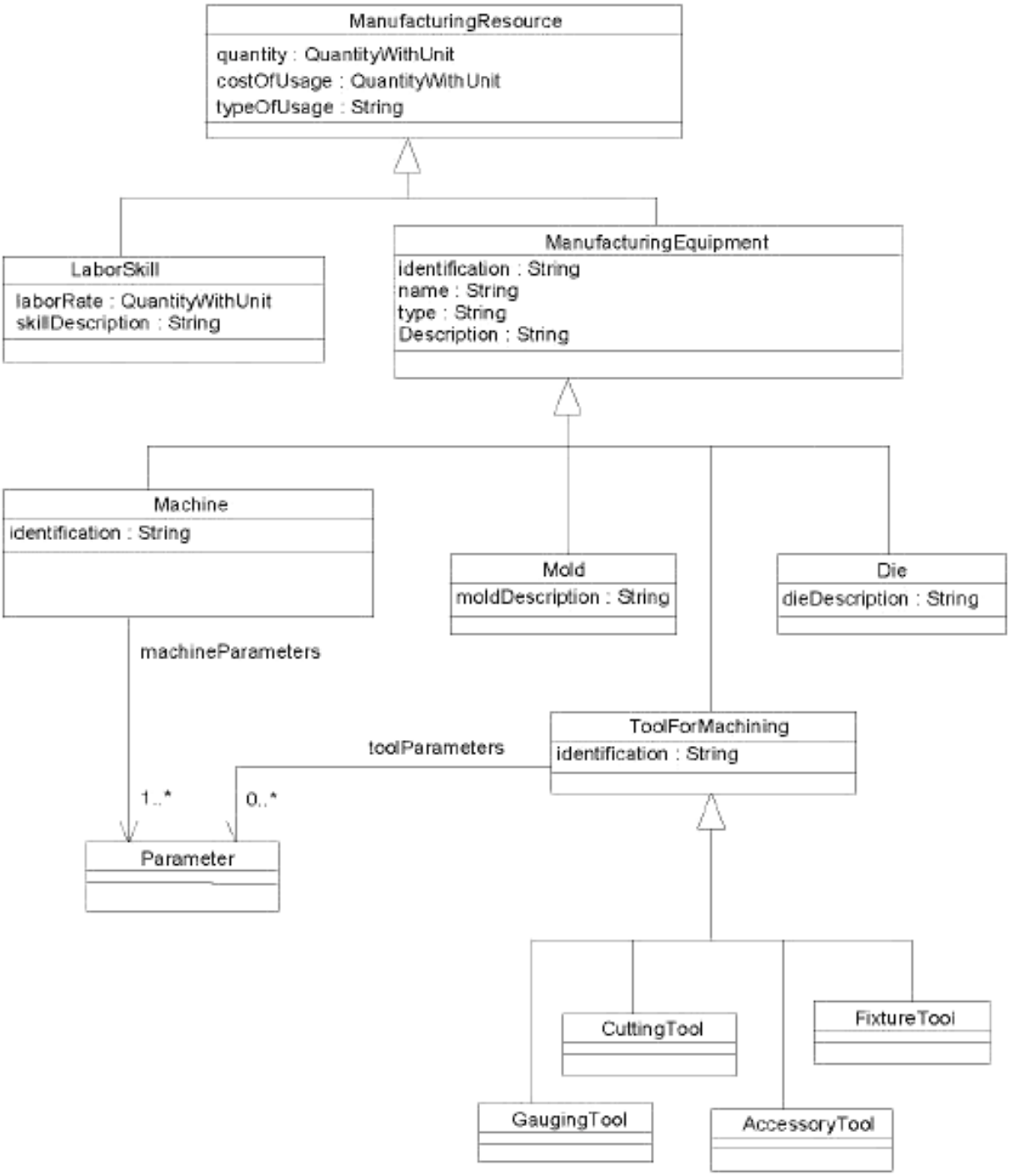

Рисунок В.14 – Объектная модель ManufacturingResource

Приложение С
(справочное)

Примеры информационной модели единицы программного обеспечения

С.1 Единица программного обеспечения для анализа данных и визуализации

Единица программного обеспечения для анализа данных и визуализации (DAV) определяет архитектуру, а также основные функции, с помощью которых осуществляют разблокировку гибкой сборки оценочных приложений разных функциональных компонентов. Основным требованием является комбинированное использование компонентов программного обеспечения нескольких производителей (поставщиков) для определения единицы программного обеспечения без информации о том, как определенный компонент реализуется или изменяет реализацию другого компонента.

Архитектура DAV должна обладать возможностью подгонки единицы программного обеспечения в согласованном направлении, например для обеспечения контроля и, может быть, просмотра – для всех участвующих компонентов. Повторно используемые модули создают для специфических, заказанных для конкретного проекта производственных решений. Они выполняют конкретную задачу, не используя дополнительную функциональность. В отношении возможности повторного использования компонентов каждый раз необходимо начинать с нуля. Кроме того, производителю не нужно рассматривать все аспекты оценивания. Главные свойства модели, связанные с аспектами архитектуры DAV, указывают в общих чертах для обеспечения общего понимания того, что входит в единицу программного обеспечения DAV и как ее используют. Общая архитектура DAV представлена на рисунке С.1.

Рисунок С.1 – Структура единицы программного обеспечения DAV на основе компонента

Описание единицы программного обеспечения сфокусировано на описании прикладных программ для просмотра данных (например, статистических данных процесса), анализа и визуализации, обычно выполняемых на одном компьютере (с доступом к другим базам данных, файлам, информационным системам и т. д., которые могут находиться в другом месте). Подробное описание применения единицы программного обеспечения в данном случае не приводят. Вместо этого для поддерживания таких прикладных программ указывают полную инфраструктуру связи и подходящие интерфейсы компонентов для обеспечения разработки стандартных компонентов.

Главным преимуществом для пользователя является независимость от отдельных поставщиков программного обеспечения, а также отсутствие разрыва между конкурирующими стандартами и индивидуальным программированием за счет предложения наилучшего решения. Для реализации этого приводят определение коммуникационной шины, которая позволяет создать основу единицы индивидуального программного обеспечения, позволяющей уменьшить связность функциональных единичных компонентов программного обеспечения. Стандартизация необходима для того, чтобы обеспечить возможность индивидуально и независимо разработанным компонентам работать вместе с помощью такой шины. Сама шина изначально является локальной концепцией, ориентированной на персональный компьютер, так как обычное приложение

является совокупностью локальных компонентов, связанных вместе с целью построения информационного потока, необходимого для оценки данных. Только единичные компоненты могут включать в себя некоторую дистанционную связь с ресурсами, не входящими в единицы программного обеспечения, главным образом, для доступа к распределенному информационному окружению производственной информационной системы.

С.2 Сервисы. Предложение общепринятых функций

Помимо шины сервис системы также является основой единицы программного обеспечения. Он определяет фундаментальную для компонентов системы функциональность. В общем случае концепцию данных для сервисов компонентов единицы программного обеспечения используют для того, чтобы обеспечить специальную функциональность единицы программного обеспечения, а также ее стандартных блоков. Некоторые из таких сервисов приведены на рисунке С.1. Дополнительные сервисы или расширения функциональности существующих сервисов будут определены в последующих изданиях настоящего стандарта.

В настоящее время служба помощи (Help Service) предоставляет стандартизованный доступ к полезной информации в отношении компонентов. Сервисная программа персонального компьютера (Desktop Service) (см. рисунок С.1) обеспечивает доступ к окнам и общему контексту графических средств в качестве абстракции среды окружения GUI (графического интерфейса пользователя) конкретного персонального компьютера, например, Microsoft Windows, UNIX X-Windows или Java Swing. Сервис регистрации и отслеживания (Logging and Tracing Service) обеспечивает выполнение простых операций, позволяющих компоненту записывать разные виды зарегистрированных сообщений, контролируя их выполнение. Это помогает при отладке программ и обеспечении повторяемых результатов вычисления. Сервис подготовки сценариев (скриптов) (Scripting Service) предоставляет простой интерфейс для выполнения контрольного считывания единичной конфигурации и этапов выполнения программы построения законченного сценария, который обеспечивает возможность точного повторения или более позднего повторного вычисления одного и того же приложения, то есть той же совокупности компонентов, соединенных друг с другом

и имеющих одинаковые параметры. Сервис свойства (Property Service) предоставляет родовую абстракцию свойствам компонента, называемым атрибутами или особенностями. Эта сервисная программа позволяет скомпоновать набор свойств общих для нескольких компонентов. Заводской сервис (Factory Service), приведенный на рисунке С.1 обеспечивает для компонента возможность создания новых объектов данных, которые могут быть использованы для связи между компонентами.

С.3 Элементарные группы данных. Сообщенные объекты

Информация (данные) передается между компонентами с помощью унифицированной гомогенной концепции, называемой items (элементарные группы данных). Фактически шина является менеджером элементарной группы данных, т. е. она управляет объявлением новых данных, доступных в форме элементарной группы, или их разрушением, если группа не является больше действительной или нужной. Элементарные группы могут быть разных видов (типов), представляющих разные базовые данные. Каждая группа может иметь специальное значение для сценария прикладной программы. Компонент, поставляющий данные (например, фасадный метод к базе данных или доступ к файлу для извлечения данных измерения) в единицу программного обеспечения, осуществляет это путем создания элементарной группы подходящего типа, заполняет ее действительными значениями данных и информирует об этом шину. В свою очередь шина информирует все компоненты, зарегистрированные в ней, о доступности новой элементарной группы. Компоненты могут извлекать информацию о новой элементарной группе (ее имени, разновидности и т. д.), чтобы решать вопрос, насколько она им нужна, например для того, чтобы провести некоторые вычисления данных этой группы.

Следовательно, во время конфигурации компонента пользователь ассоциирует необходимые группы (обычно по имени) с вводом компонента. При этом создается единица программного обеспечения, так как ассоциация элементарных групп с компонентами соединяет функциональные стандартные блоки. Для передачи элементарных групп между компонентами без искажения идеи свободного соединения компонентов используют концепцию событий (events).

Событие, которое представляет программный объект с точки зрения средства реализации, определяет элементарную группу и сигнализирует о ее доступности, изменении или удалении. В итоге основой соединения компонентов друг с другом и с шиной является регистрация источников и получателей информации. Для обеспечения гибкости и высокой степени сопряжения большинство событий обмениваются данными через шину и не обмениваются данными непосредственно между компонентами. Разные виды событий, например изменение или отмена данных, различают с помощью соответствующих меток, сопровождающих каждое событие. Помимо шины, сервиса, элементарной группы данных и концепций событий основной базой, на которую опирается архитектура DAV, является нотация «компонент».

С.4 Компоненты программного обеспечения. Функциональные модули единицы программного обеспечения

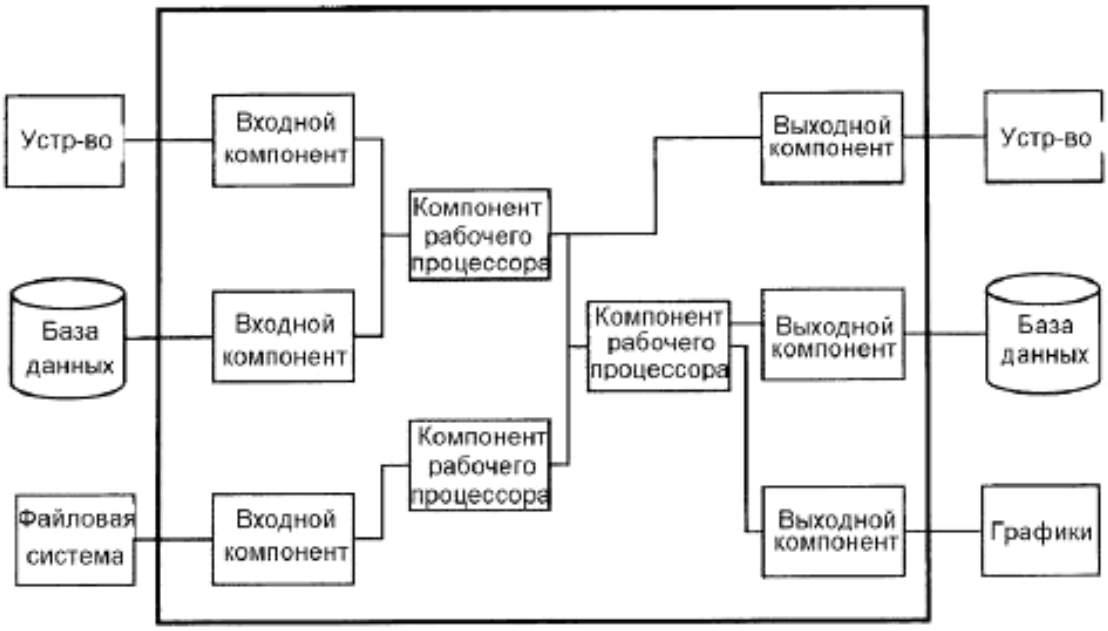

Рисунок С.2 – Логический вид структуры и потока информации единицы программного обеспечения DAV

Термин единица программного обеспечения обозначает агрегатный состав функциональных модулей (компонентов программы). Каждый модуль определяет специальную полную замкнутую функциональность, достаточно обширную для инкапсуляции в простом индивидуальном стандартном блоке и достаточно небольшую для обеспечения возможности компоновки с дополнительной

функциональностью (в форме других таких же стандартных блоков) для создания завершенных цепей вычислений. Независимый поставщик программного средства может предложить единицу вычисления, достаточно функциональную, как и его собственная разработка. В этом случае каждый стандартный блок называют компонентом. Если программные интерфейсы соответствуют требованиям настоящего стандарта, то это гарантирует возможность их работы с другими компонентами (возможно, распространяемыми другими поставщиками), также соответствующими настоящему стандарту.

Предпочтительность использования архитектуры, созданной на основе компонентов, заключается в следующем:

a) компоненты могут быть созданы разными поставщиками:

1) не существует зависимости от одного поставщика компонентов;

2) может быть выпущено множество разных компонентов для решения общих или специальных проблем;

3) пользователи могут выбирать компоненты из большего числа наборов;

4) проекты могут интегрировать компоненты третьей стороны, сокращая время разработки (и, возможно, затраты);

5) производство компонентов является менее сложным, чем целых прикладных программ, поэтому продавать компоненты могут небольшие специализированные предприятия;

6) разработчику компонентов не обязательно знать конечное приложение. Он может уделять больше внимания эффективности, функционированию и результату частной проблемы, в которой он является экспертом;

b) компоненты предназначены для многократного использования:

1) компоненты могут быть созданы или куплены отдельно и затем быть использованы вместе или в разных комбинациях с другими компонентами в разных приложениях или контекстах приложений;

2) единичные концепции могут быть реализованы независимо от существующих концепций;

3) для повторного использования компонентов в как можно большем числе ситуаций, необходимо иметь собственных клиентов и диалоги конфигураций, которые позволяют инженеру-испытателю устанавливать характеристики и атрибуты компонентов в пределах, предпочтительных для разработчика компонента. Приложения с такими компонентами обычно имеют больше ошибок, чем произвольно модифицируемое ПО. Диалоги конфигурации могут изменяться от управляющих элементов, позволяющих выбирать вариант из простого множества, до полного мастера настройки (Wizard) и помогать инженеру в комплексной конфигурации;

c) композиция разных компонентов обеспечивает быструю разработку новых, очень гибких приложений, которые снижают расходы на техническое обслуживание и администрирование:

1) опыт, полученный в результате создания аппаратных средств, например, применение интегральных схем, внедряют в разработки;

2) компонент может быть заменен другим компонентом (если необходимо, от другого поставщика), а также новый компонент может быть добавлен, не изменяя другие части приложения;

3) существующие системы и функциональность могут быть определены в качестве модели компонента и более легко интегрированы в новые разработки.

Для облегчения сборки разных компонентов приложения их необходимо передавать в виде автономных пакетов, включающих в себя информацию о технической функциональности, руководства пользователей (помощь), версию, национальный язык и информацию об уровне сертификации.

Чтобы существование таких компонентов стало реальностью, необходима стандартизация основы и возможно, модели компонента, имеющего отношение к конкретной предметной области приложения. Приведенная модель в основном касается интерфейсов, связей, а также поддерживающих сервисов.

С.5 Установка единицы программного обеспечения

В настоящем разделе рассмотрен технический уровень операции, используемый во время установки и выполнения единицы ПО. Диаграммы последовательности используют для того, чтобы дать описание необходимых

шагов. Каждая диаграмма показывает соответствующие обращения к операциям. Так как некоторые детали, в частности, загрузку компонента, не указывают, курсивный шрифт или аннотацию используют для описания псевдоопераций аналогично созданию экземпляра объекта.

Имеется некоторый уровень вариации взаимодействия, например тестовая сервисная программа может обрабатывать данные параллельно, например давать описания загрузки компонента и выполнять сервисы создания экземпляров, или компонент может ожидать выполнения процедуры конфигурирования пользователем до проведения операции просмотра элементарной группы. Типовой сценарий начальной установки единицы программного обеспечения, а также загрузка первого компонента программного обеспечения приведены на рисунке С.3.

Образец жизненного цикла компонента программного обеспечения включает в себя следующие этапы:

- загрузку;
- инициализацию;
- конфигурирование;
- выполнение;
- выключение.

Допускается также временное введение циклов в программу между операциями выполнения и конфигурации.

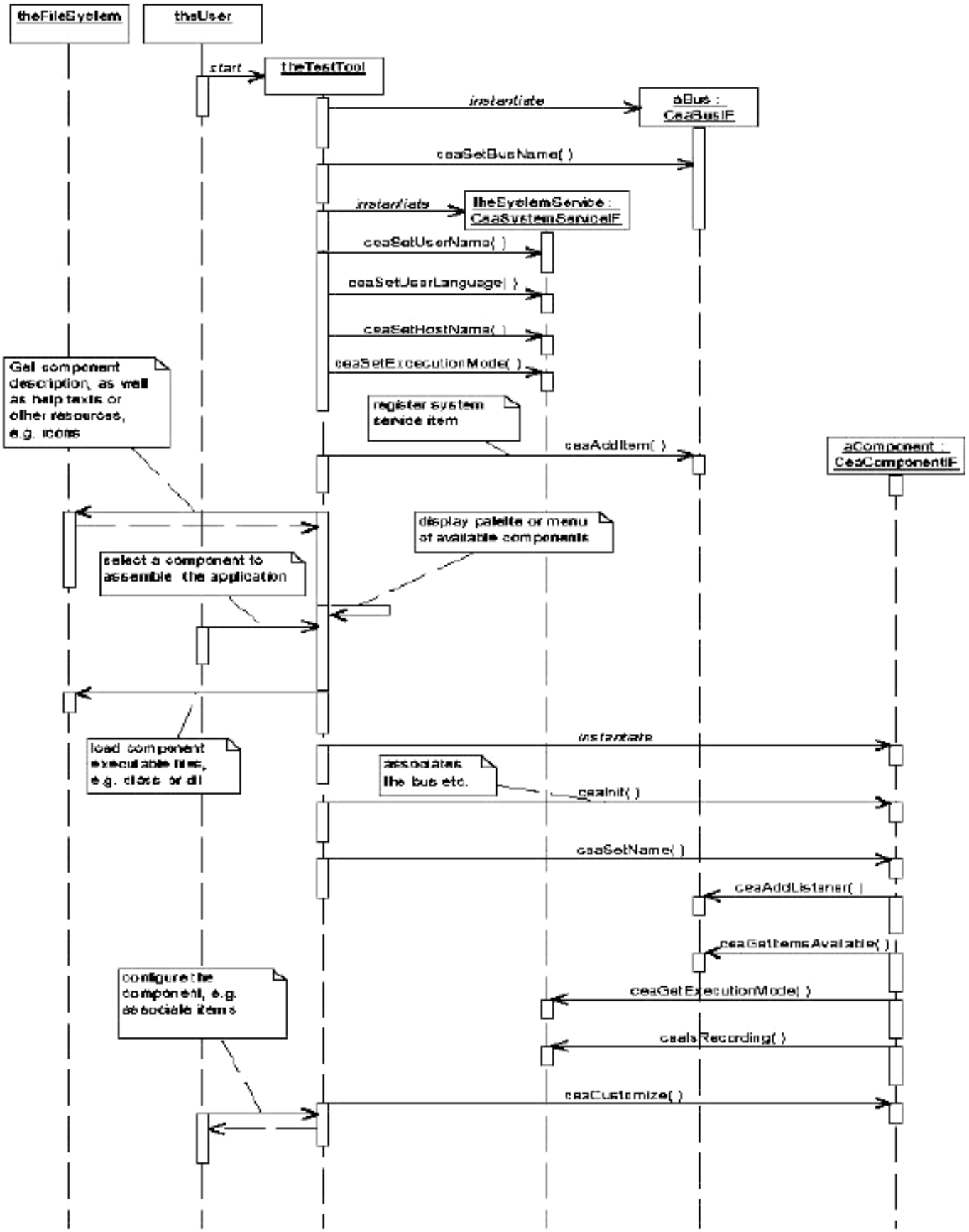

Рисунок С.3 – Жизненный цикл единицы программного обеспечения DAV: суммирование первого компонента

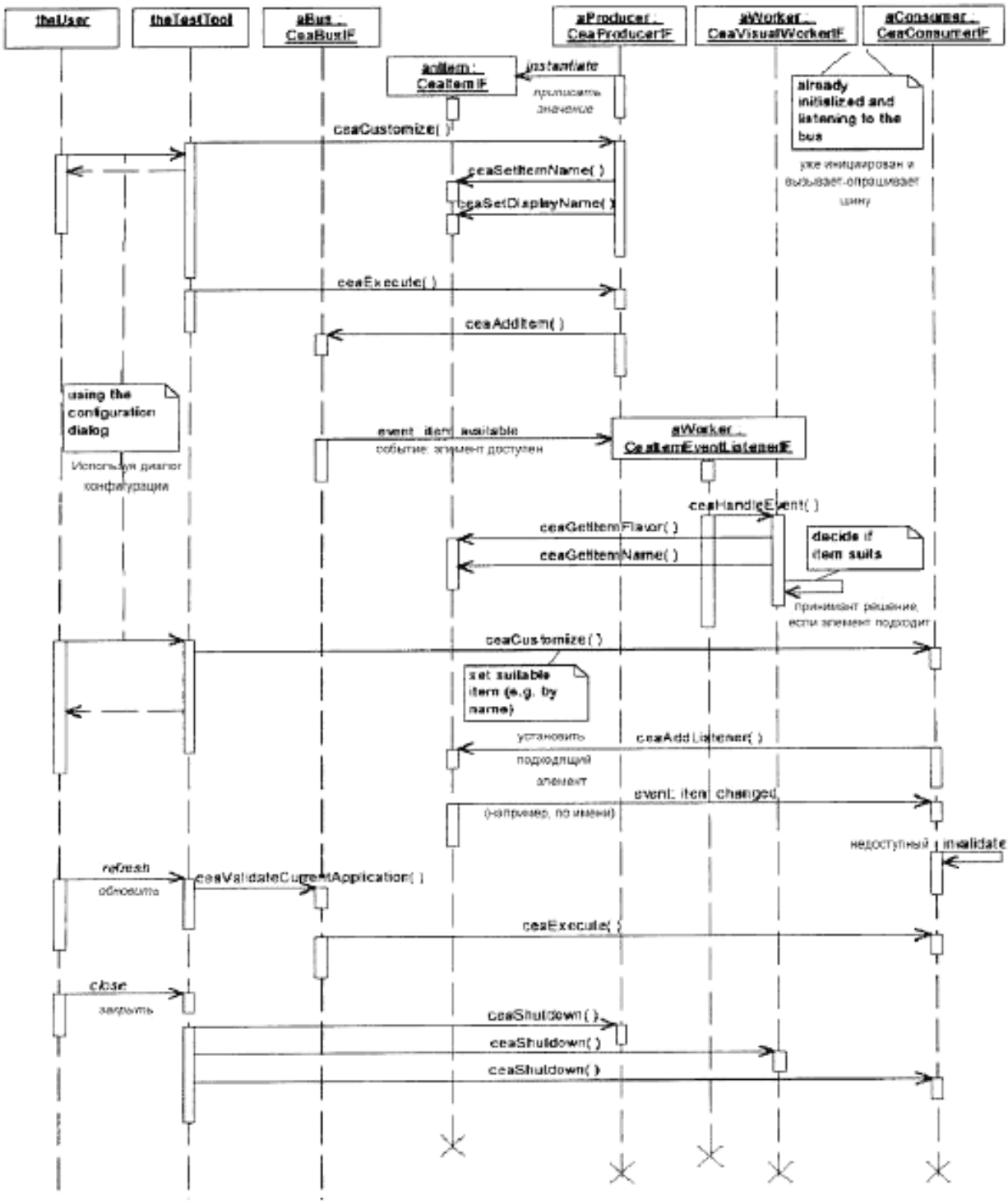

Рисунок С.4 – Жизненный цикл единицы программного обеспечения DAV: типовое взаимодействие в процессе прогона программы

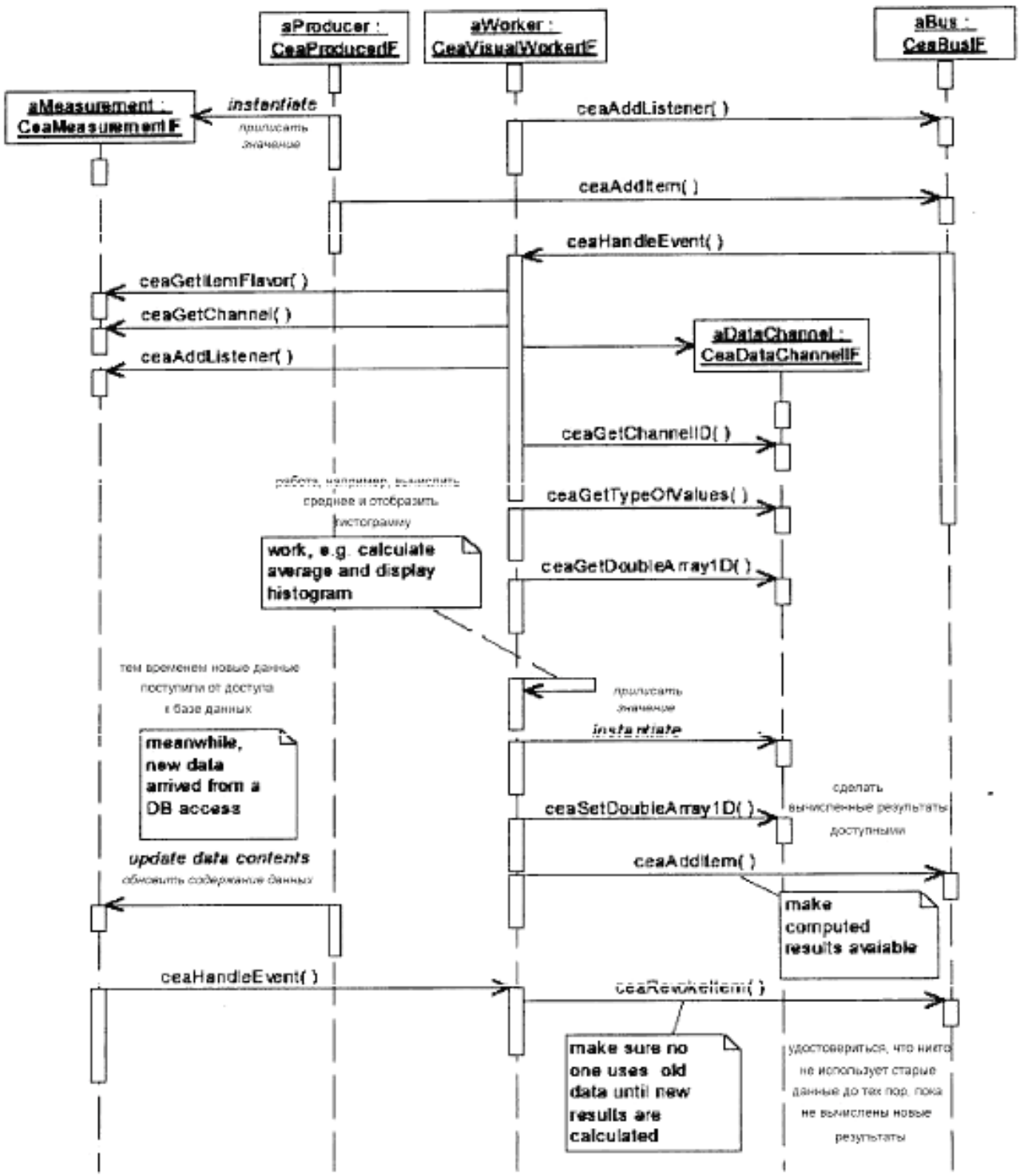

Рисунок С.5 – Жизненный цикл единицы программного обеспечения DAV: типовое взаимодействие, имеющее отношение к элементарным группам данных

С.6 Пример объектов, обменивающихся информацией

На рисунке С.6 приведен пример обменивающихся между собой информацией объектов, характеризующих диаграмму класса исполнительной производственной системы (Manufacturing Execution System)

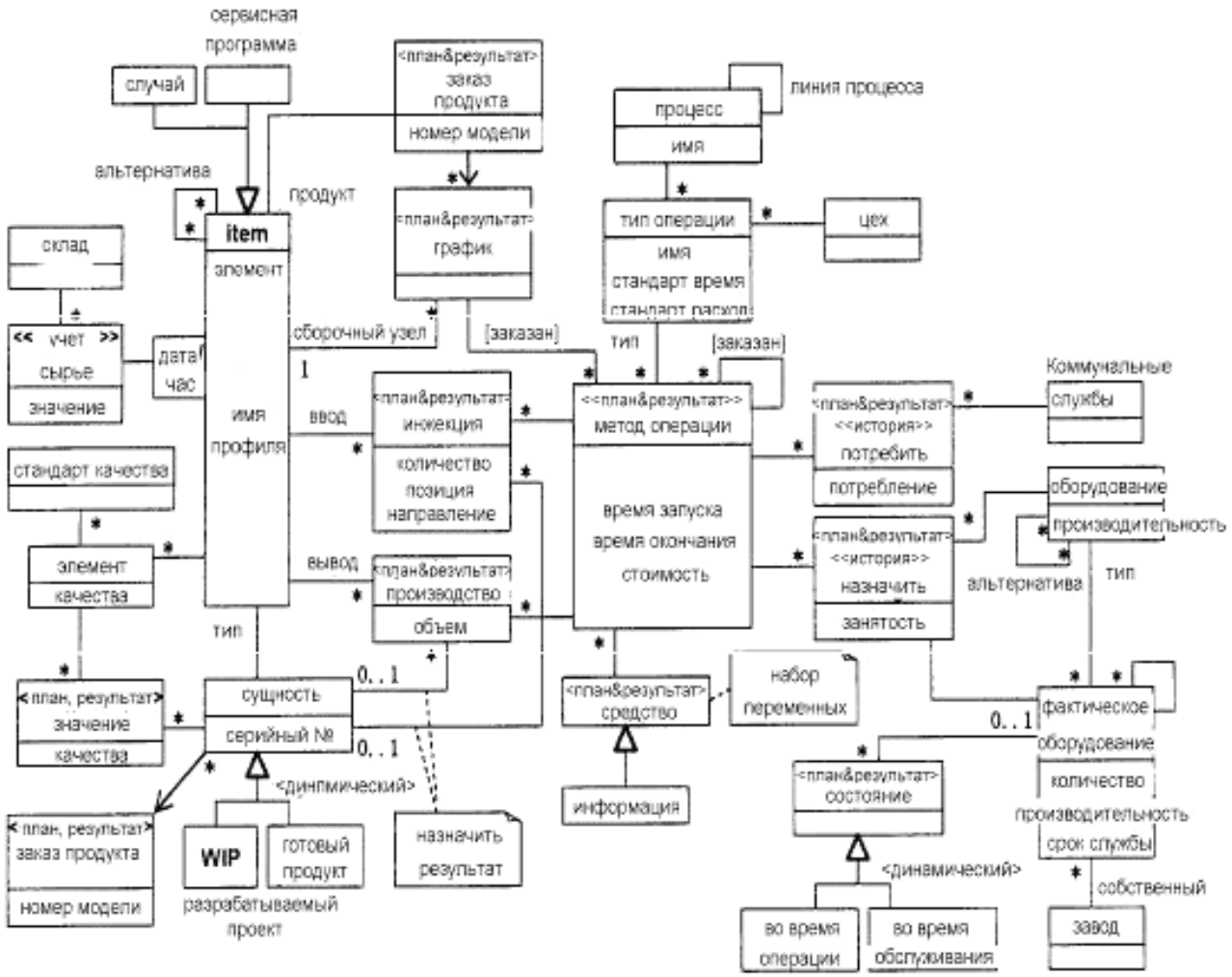

Рисунок С.6 – Пример диаграммы класса Manufacturing Execution System

На этом примере единица программного обеспечения, оценивающая качество сущности по измеренному значению качества, получает следующие объекты, обменивающиеся информацией:

a) атрибут серийного номера сущности;

b) значение качества сущности, измеренное с помощью оборудования;

c) элементы качества из базы данных стандартного качества.

Единица программного обеспечения сравнивает полученные значения и выдает сведения о качестве сущности как взаимосвязанных информационных

объектах в единицу программного обеспечения, например, в исполнительную производственную систему (MES) на верхних уровнях иерархии, показанную на рисунке В.1.

Приложение ДА

(справочное)

Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ИСО 16100-1:2002	-	*
ИСО 16100-2:2003	-	*

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

УДК 658.52.011.56 ОКС 25.040.40 Т 58

Ключевые слова: автоматизированные промышленные системы, интеграция, жизненный цикл систем, управление производством

Подписано в печать 30.04.2014. Формат 60x84 1/8.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

ФГУП «СТАНДАРТИНФОРМ»

123995 Москва, Гранатный пер., 4.
www.gostinfo.ru info@gostinfo.ru