МЕЖДУНАРОДНЫЙ СОЮЗ ЭЛЕКТРОСВЯЗИ

МСЭ-Т X.744.1

(03/2003)

СЕКТОР СТАНДАРТИЗАЦИИ
ЭЛЕКТРОСВЯЗИ МСЭ

СЕРИЯ X: СЕТИ ПЕРЕДАЧИ ДАННЫХ
И ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ

Управление ВОС – Функции управления
и функции ODMA

Служба управления программными средствами TMN на основе CORBA

Рекомендация МСЭ-Т X.744.1

РЕКОМЕНДАЦИИ МСЭ-Т СЕРИИ X

СЕТИ ПЕРЕДАЧИ ДАННЫХ И ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ

СЕТИ ПЕРЕДАЧИ ДАННЫХ ОБЩЕГО ПОЛЬЗОВАНИЯ	
Службы и услуги	X.1–X.19
Интерфейсы	X.20–X.49
Передача, сигнализация и коммутация	X.50–X.89
Сетевые аспекты	X.90–X.149
Техническое обслуживание	X.150–X.179
Административные предписания	X.180–X.199
ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ	
Модель и обозначение	X.200–X.209
Определения служб	X.210–X.219
Спецификации протоколов в режиме с установлением соединений	X.220–X.229
Спецификации протоколов в режиме без установления соединений	X.230–X.239
Проформы PICS	X.240–X.259
Идентификация протоколов	X.260–X.269
Протоколы обеспечения безопасности	X.270–X.279
Управляемые объекты уровня	X.280–X.289
Испытание на соответствие	X.290–X.299
ВЗАИМОДЕЙСТВИЕ МЕЖДУ СЕТЯМИ	
Общие положения	X.300–X.349
Спутниковые системы передачи данных	X.350–X.369
IP-сети	X.370–X.399
СИСТЕМЫ ОБРАБОТКИ СООБЩЕНИЙ	X.400–X.499
СПРАВОЧНИК	X.500–X.599
ОРГАНИЗАЦИЯ СЕТИ ВОС И СИСТЕМНЫЕ АСПЕКТЫ	
Организация сети	X.600–X.629
Эффективность	X.630–X.639
Качество обслуживания	X.640–X.649
Наименование, адресация и регистрация	X.650–X.679
Абстрактно-синтаксическая нотация 1 (ASN.1)	X.680–X.699
УПРАВЛЕНИЕ ВОС	
Структура и архитектура управления системами	X.700–X.709
Служба и протокол связи для управления	X.710–X.719
Структура управляющей информации	X.720–X.729
Функции управления и функции ODMA	**X.730–X.799**
БЕЗОПАСНОСТЬ	X.800–X.849
ПРИЛОЖЕНИЯ ВОС	
Фиксация, параллельность и восстановление	X.850–X.859
Обработка транзакций	X.860–X.879
Удаленные операции	X.880–X.899
ОТКРЫТАЯ РАСПРЕДЕЛЕННАЯ ОБРАБОТКА	X.900–X.999

Для получения более подробной информации просьба обращаться к перечню Рекомендаций МСЭ-Т.

Рекомендация МСЭ-Т X.744.1

Служба управления программными средствами TMN на основе CORBA

Резюме

Настоящая Рекомендация является одной из серии Рекомендаций, определяющих требования к интерфейсу CORBA для обеспечения взаимосвязи между операционными системами (ОС) и сетевыми элементами (СЭ), между ОС и связующим устройством (СУ), между ОС и адаптером Q (QA), а также между различными ОС в сети управления электросвязью (TMN). В настоящей Рекомендации предлагается реализация управления программными средствами на основе CORBA, основываясь на Рекомендации МСЭ-Т X.744. Цель настоящей Рекомендации состоит в том, чтобы определить модель CORBA/IDL, аналогичную модели, определенной в Рекомендации МСЭ-Т X.744, с использованием CMISE. В настоящей Рекомендации определены оба вида интерфейсов – утонченной структуры и грубой (т. е. фасадной) структуры для функций управления программными средствами. Настоящая Рекомендация совместима со стандартами по моделированию CORBA согласно Рекомендациям МСЭ-Т X.780, X.780.1, Q.816, Q.816.1 и M.3120.

Источник

Рекомендация МСЭ-Т X.744.1 была подготовлена 4-й Исследовательской комиссией МСЭ-Т (2001–2004 гг.) и утверждена 29 марта 2003 года в соответствии с Резолюцией 1 ВАСЭ.

ПРЕДИСЛОВИЕ

Международный союз электросвязи (МСЭ) является специализированным учреждением Организации Объединенных Наций в области электросвязи. Сектор стандартизации электросвязи МСЭ (МСЭ-Т) – постоянный орган МСЭ. МСЭ-Т отвечает за изучение технических, эксплуатационных и тарифных вопросов и за выпуск Рекомендаций по ним с целью стандартизации электросвязи на всемирной основе.

Всемирная ассамблея по стандартизации электросвязи (ВАСЭ), которая проводится каждые четыре года, определяет темы для изучения Исследовательскими комиссиями МСЭ-Т, которые, в свою очередь, вырабатывают Рекомендации по этим темам.

Утверждение Рекомендаций МСЭ-Т осуществляется в соответствии с процедурой, изложенной в Резолюции 1 ВАСЭ.

В некоторых областях информационных технологий, которые входят в компетенцию МСЭ-Т, необходимые стандарты разрабатываются на основе сотрудничества с ИСО и МЭК.

ПРИМЕЧАНИЕ

В настоящей Рекомендации термин "администрация" используется для краткости и обозначает как администрацию электросвязи, так и признанную эксплуатационную организацию.

Соответствие положениям данной Рекомендации является добровольным делом. Однако в Рекомендации могут содержаться определенные обязательные положения (для обеспечения, например, возможности взаимодействия или применимости), и тогда соответствие данной Рекомендации достигается в том случае, если выполняются все эти обязательные положения. Для выражения требований используются слова "shall" ("должен", "обязан") или некоторые другие обязывающие термины, такие как "must" ("должен"), а также их отрицательные эквиваленты. Использование таких слов не предполагает, что соответствие данной Рекомендации требуется от каждой стороны.

ПРАВА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

МСЭ обращает внимание на то, что практическое применение или реализация этой Рекомендации может включать использование заявленного права интеллектуальной собственности. МСЭ не занимает какую бы то ни было позицию относительно подтверждения, обоснованности или применимости заявленных прав интеллектуальной собственности, независимо от того, отстаиваются ли они членами МСЭ или другими сторонами вне процесса подготовки Рекомендации.

На момент утверждения настоящей Рекомендации МСЭ не получил извещение об интеллектуальной собственности, защищенной патентами, которые могут потребоваться для реализации этой Рекомендации. Однако те, кто будет применять Рекомендацию, должны иметь в виду, что это может не отражать самую последнюю информацию, и поэтому им настоятельно рекомендуется обращаться к патентной базе данных БСЭ.

СОДЕРЖАНИЕ

Рекомендация МСЭ-Т X.744.1

Служба управления программными средствами TMN на основе CORBA

1 Область применения

Настоящая Рекомендация является одной из серии Рекомендаций, определяющих требования к интерфейсу CORBA для обеспечения взаимосвязи между операционными системами (ОС) и сетевыми элементами (СЭ), между ОС и связующим устройством (СУ), между ОС и адаптером Q (QA), а также между различными ОС в сети управления электросвязью (TMN) [1].

Функция управления программными средствами охватывает управление системой доставки программных средств, а также управление программными средствами в пределах системы.

Существуют два аспекта программного обеспечения, которые следует рассмотреть по отдельности. С точки зрения программного обеспечения эти аспекты можно назвать пассивными и активными.

Пассивный аспект программного обеспечения касается данных, хранимых в системе управления, а также способа их доставки и инсталляции. В общем случае "данные" представляют собой хранимую информацию типа файлов и таблиц, но к ним могут относиться также файлы, содержащие исполняемый код. К области применения настоящей Рекомендации относится пассивная функция программного обеспечения.

Активная функция программного обеспечения касается управления ресурсами, которые используют программные средства. Между этой точкой зрения и обычной точкой зрения на управление ресурсами нет фактических различий. В область применения настоящей Рекомендации не входит активный аспект программного обеспечения. Однако взаимоотношения между управляемыми объектами, представляющими ресурсы, которые используют программные средства, и управляемыми объектами, представляющими программные средства, которые используют ресурсы (т. е. пассивный аспект программного обеспечения), входят в предмет рассмотрения настоящей Рекомендации.

В область применения настоящей Рекомендации входят:

- инициализация передачи программных средств;
- управление почтовой передачей программных средств;
- активизация программного обеспечения (включая обновление версий и латание);
- деактивизация программного обеспечения;
- реверсивные изменения программного обеспечения;
- проверка правильности программного обеспечения;
- запросы программного обеспечения;
- резервирование программных средств;
- восстановление программных средств.

В область применения настоящей Рекомендации не входят:

- механизм передачи программных средств;
- физическое хранилище программных средств (отображение программных средств в физическом хранилище файлов, таком как гибкие диски, жесткие диски и т. п.);
- форматирование программных средств;
- присвоение имен программным продуктам;
- упорядочение команд управления программными средствами;
- мониторинг программного обеспечения;
- управление процессами системы.

В настоящей Рекомендации интерфейсы утонченной и грубой (т. е. фасадной) структур определены для функций управления программными средствами. Интерфейсы утонченной структуры и грубой структуры обеспечиваются одинаковой поддержкой со стороны управления программными средствами. И те и другие могут быть использованы вместе с управляемыми объектами утонченной структуры и грубой структуры (например, определенными в Рекомендации МСЭ-Т M.3120 [2]).

Современные сети электросвязи охватывают большое и все возрастающее количество ОС и сетевых элементов различных поставщиков. Количество и многообразие сетей и услуг растут, создавая многообразие потребностей в управлении. Этот рост приводит к возрастанию количества уникальных связных интерфейсов между ОС и сетевыми элементами. Промышленность электросвязи получает преимущества от стандартизации этих интерфейсов, предназначенных для достижения взаимодействия между широким диапазоном ОС и сетевых элементов/адаптеров Q, использующих в необходимых случаях связующие устройства, а также между ОС.

Основная цель настоящей Рекомендации состоит в том, чтобы предоставить набор сообщений прикладного уровня и соответствующих поддерживающих объектов для обеспечения взаимосвязи через интерфейсы CORBA. Поскольку желательно обеспечить общие решения TMN, предполагаются, что эти сообщения и поддерживающие объекты будут применимы к другим TMN или относящимся к TMN интерфейсам.

2 Ссылки

Перечисленные ниже Рекомендации МСЭ-Т и другие документы содержат положения, которые путем ссылок на них в тексте образуют положения настоящей Рекомендации. Во время публикации указанные издания были действительными. Все Рекомендации и другие документы подлежат пересмотрам; поэтому пользователям настоящей Рекомендации рекомендуется применять самые последние издания этих Рекомендаций и других перечисленных ниже документов. Список действующих в данное время Рекомендаций МСЭ-Т публикуется регулярно. Ссылка на документ в настоящей Рекомендации не придает ему как самостоятельному документу статус Рекомендации.

[1] ITU-T Recommendation M.3010 (2000), *Principles for a telecommunications management network*.

[2] ITU-T Recommendation M.3120 (2001), *CORBA generic network and network element level information model*.

[3] ITU-T Recommendation Q.816 (2001), *CORBA-based TMN services*.

[4] ITU-T Recommendation Q.816 (2001), *CORBA-based TMN services* plus Amendment 1 (2001): *OMG services profile*.

[5] ITU-T Recommendation Q.816 (2001), *CORBA-based TMN services* plus Amendment 2 (2002): *User guide for local name resolution*.

[6] ITU-T Recommendation Q.816 (2001), *CORBA-based TMN services* plus Corrigendum 1 (2001).

[7] ITU-T Recommendation Q.816.1 (2001), *CORBA-based TMN services*: *Extensions to support coarse-grained interfaces.*

[8] ITU-T Recommendation Q.822.1 (2001), *CORBA-based TMN performance management service*.

[9] ITU-T Recommendation X.701 (1997), *Information technology – Open Systems Interconnection – Systems management overview*.

[10] ITU-T Recommendation X.720 (1992), *Information technology – Open Systems Interconnection – Structure of management information: Management information model.*

[11] ITU-T Recommendation X.721 (1992), *Information technology – Open Systems Interconnection – Structure of management information: Definition of management information*.

[12] ITU-T Recommendation X.722 (1992), *Information technology – Open Systems Interconnection – Structure of management informations: Guidelines for the definition of managed objects*.

[13] ITU-T Recommendation X.723 (1993), *Information technology – Open Systems Interconnection – Structure of management information: Generic management information*.

[14] ITU-T Recommendation X.731 (1992), *Information technology – Open Systems Interconnection – Systems management: State management function*.

[15] ITU-T Recommendation X.740 (1992), *Information technology – Open Systems Interconnection – Systems management: Security audit trail function*.

[16] ITU-T Recommendation X.741 (1995), *Information technology – Open Systems Interconnection – Systems management: Objects and attributes for access control*.

[17] ITU-T Recommendation X.742 (1995), *Information technology – Open Systems Interconnection – Systems management: Usage metering function for accounting purposes*.

[18] ITU-T Recommendation X.744 (1996), *Information technology – Open Systems Interconnection – Systems management: Software management function*.

[19] ITU-T Recommendation X.744 (1996), *Information technology – Open Systems Interconnection – Systems management: Software management function* plus Technical Corrigendum 1 (1998).

[20] ITU-T Recommendation X.744 (1996), *Information technology – Open Systems Interconnection – Systems management: Software management function* plus Technical Corrigendum 2 (2000).

[21] ITU-T Recommendation X.744 (1996), *Information technology – Open Systems Interconnection – Systems management: Software management function* plus Technical Corrigendum 3 (2001).

[22] ITU-T Recommendation X.745 (1993), *Information technology – Open Systems Interconnection – Systems management: Test management function*.

[23] ITU-T Recommendation X.746 (2000), *Information technology – Open Systems Interconnection – Systems management: Scheduling function*.

[24] ITU-T Recommendation X.780 (2001), *TMN guidelines for defining CORBA managed objects*.

[25] ITU-T Recommendation X.780 (2001), *TMN guidelines for defining CORBA managed objects* plus Corrigendum 1 (2001).

[26] ITU-T Recommendation X.780 (2001), *TMN guidelines for defining CORBA managed objects* plus Corrigendum 2 (2002).

[27] ITU-T Recommendation X.780 (2001), *TMN guidelines for defining CORBA managed objects* plus Amendment 1 (2002), *System objects and user guide for bulk attribute retrieval*.

[28] ITU-T Recommendation X.780.1 (2001), TMN *guidelines for defining coarse-grained CORBA managed object interfaces*.

[29] ITU-T Recommendation X.780.1 (2001), *TMN guidelines for defining coarse-grained CORBA managed object interfaces* plus Amendment 1 (2002), *System façades and user guide for bulk attribute retrieval*.

[30] ITU-T Recommendation X.780.1 (2001), *TMN guidelines for defining coarse-grained CORBA managed object interfaces* plus Corrigendum 1 (2002).

3 Определения

В настоящей Рекомендации использованы следующие термины из других Рекомендаций:

3.1 В настоящей Рекомендации использованы следующие термины из Рекомендации МСЭ-Т M.3010 [1]:

a) Информационная модель управления

b) Администратор

3.2 В настоящей Рекомендации использованы следующие термины из Рекомендации МСЭ-Т X.701 [9]:

a) Класс управляемых объектов

3.3 В настоящей Рекомендации использованы следующие термины из Рекомендации МСЭ-Т X.744 [18]:

a) Резервирование

b) Доставка

c) Выполнение

d) Инсталляция

e) Восстановление

f) Возврат

g) Использование

h) Проверка достоверности

3.4 В настоящей Рекомендации использованы следующие термины из Рекомендации МСЭ-Т X.780 [24]:

a) Унаследованная иерархия

b) Дерево наименований

c) Подчиненные объекты

d) Старшие объекты

4 Сокращения

В настоящей Рекомендации используются следующие сокращения:

CORBA Брокерская архитектура запроса общих объектов

IDL Язык определения интерфейсов

MD Связующее устройство

NE Сетевой элемент

OMG Группа управления объектами

ORB Брокер запроса объектов

OS Операционная система

QA Q-адаптер

TMN Сеть управления электросвязью

UID Универсальный идентификатор

5 Требования к управлению программными средствами

Управление программными средствами должно быть способно удовлетворять следующим требованиям в зависимости от возможных предписанных управляющих действий и существующих условий:

a) обладать способностью запрашивать доставку программных средств конкретной управляемой системе;

b) обладать способностью управлять инсталляцией программных средств в управляемой системе, включая инсталляцию "заплат" (например, усовершенствований) и возврата к предыдущей версии программного обеспечения;

c) обладать способностью инициировать исполнение программы;

d) обладать способностью запрашивать информацию относительно атрибутов всех программных средств, содержащихся в управляемой системе;

e) обладать способностью создавать и удалять программные средства, содержащиеся в управляемой системе;

f) обладать способностью проверять правильность программных средств, содержащихся в управляемой системе, с целью проверки их целостности;

g) обладать способностью ограничивать использование программных ресурсов в управляемой системе для административных целей;

h) обладать способностью резервирования программных элементов и восстанавливать ранее зарезервированные программные элементы.

Эта модель не должна препятствовать использованию регистрации, учета, аудиторских проверок и лицензирования на каждой стадии этого процесса.

Во всех случаях управляющая система должна быть проинформирована об успешности или безуспешности выполнения операции

Взаимоотношения, существующие между программными управляемыми объектами, отражают способ использования одними программными ресурсами других программных ресурсов. Эти взаимоотношения известны как "использование", и они применятся для информирования о том, какие версии программного обеспечения должны использоваться в конкретное время. Управление этими взаимоотношениями – вопрос дальнейших исследований.

6 Модель функции управления программными средствами

Услуги управления программными средствами касаются в основном управляющих частей программного обеспечения. В этом контексте программные средства включают в себя данные, управляющую информацию и инструкции по исполнению. Административное представление программных средств с точки зрения управления может быть выражено управляемым объектом класса "программный модуль". Управление программными средствами, представляющими исполняемые инструкции, имеют дополнительные характеристики и представлены управляемым объектом "исполняемая программа" (который является подклассом программного модуля).

Другой аспект управления программными средствами состоит в управлении доставкой программных средств управляемой системе. Программные средства не обязательно управляются в тех же единицах, в которых они доставляются. Например, возможно, что многие различные программные модули могут быть доставлены управляемой системе вместе (например, в CD-ROM). Точно так же, если большое число программных модулей должно быть доставлено через сеть связи, может оказаться необходимым разделить их на небольшие части в целях доставки. Другой случай относится к доставке только тех изменений, которые требуются для существующих программных объектов, например "заплат". Управление доставкой осуществляется в понятиях управляемого объекта "дистрибьютор программных средств".

6.1 Возможности управления программными средствами

6.1.1 Создание

Программный модуль может быть создан в управляемой системе путем использования операции "создание". Программный модуль может быть создан в состоянии "создание" или "доставка" в зависимости от режима работы. Программный модуль может быть создан также в результате выполнения операции "доставка" в управляемом объекте "дистрибьютор программных средств" или же в результате некоторого локального действия. По результатам создания управляемый объект "программный модуль" может выдать уведомление "создание объекта".

6.1.2 Удаление

Программный модуль может быть удален из управляемой системы с использованием операции "удаление". Побочным результатом операции "удаление" в программном модуле может быть удаление соответствующих нижерасположенных ресурсов. По результатам удаления управляемый объект "программный модуль" может выдать уведомление "удаление объекта".

6.1.3 Доставка

Может быть запрошена доставка скоординированного набора программных модулей. Результатом доставки может быть успешность или безуспешность. Доставка осуществляется путем передачи операции "доставка" объекту "дистрибьютор программных средств". Результатом операции "доставка" является доставка копий заданных программных модулей в заданную систему, находящуюся в состоянии "доставка". Побочным результатом может быть создание одного или нескольких соответствующих объектов (например, программных модулей).

Компоновка программных модулей и выбор механизма передачи являются локальным вопросом и не входят в предмет рассмотрения настоящей Рекомендации. Например, эта информация может быть предварительно сконфигурирована или определена в операции "доставка" наряду с другой соответствующей информацией.

6.1.4 Исполняемая программа

Управляющая система может запросить выполнение исполняемой программы через операцию "исполняемая программа". Настоящая Рекомендация не определяет, каким образом может в дальнейшем управляться выполнение исполняемой программы, а обеспечивает только механизм для инициализации выполнения исполняемой программы.

6.1.5 Получение

Можно отыскать информацию относительно программных средств (какие программные средства имеются в наличии, какие из них доступны для использования, взаимоотношения между программными средствами и др.) с использованием операции "получить". В случае успеха результат операции "получить" будет содержать запрошенную информацию.

6.1.6 Инсталляция

Инсталляция настривает программные средства на их использование. В принципе это может потребовать значительного объема обрабатывающих ресурсов и времени, проверки наличия всех частей заданной версии программного обеспечения в управляемой системе (независимо от того, доставлены они как часть обновления или уже имеются) и их сборки для готового использования. Инсталляция выполняется через операцию "инсталляция", направленную на управляемый объект "программный модуль", где он обладает факультативными характеристиками.

"Заплата" представляет собой модификацию программных средств и может быть представлена управляемым объектом "программный модуль". Следовательно, "заплата" может быть доставлена, инсталлирована, использована, скопирована и т. п. с использованием управления программными средствами.

Специальный случай инсталляции состоит в "заплате", когда имеется уже доставленная "заплата" и применением "заплаты" являются инсталляция и выработка обновленной версии программных средств, готовой к использованию. Обновленная версия воспринимает обновленный номер версии для идентификации, доставленный с "заплатой". Результат инсталляции указывает успешность или безуспешность ее выполнения.

После завершения доставки программных средств может начинаться инсталляция. Может оказаться необходимым скоординировать инсталляцию; однако такое координирование не входит в предмет рассмотрения настоящей Рекомендации. Примерами координации служат:

- выполнение доставки нескольких программных модулей до инсталляции других программных модулей;
- проверка того, что действующие программные средства не используются до активизации нового средства;
- проверка того, что конкретные версии других программных средств (не) инсталлированы;
- синхронизация инсталляции с другими открытыми системами, предусматривающая одновременную, но не связанную друг с другом инсталляцию в нескольких открытых системах, предусматривающую способ инсталляции в нескольких открытых системах для их объединения в единый этап работы, и т. д.

6.1.7 Возврат

Операция "возврат" используется для того, чтобы возвратить инсталлированный управляемый объект "программный модуль" обратно в неинсталлированное состояние либо обусловить удаление одной или нескольких использованных "заплат". Операция "возврат" выполняется методом, ориентированным на управляемый объект "программный модуль", где он обладает факультативными характеристиками.

Может оказаться необходимым иметь дополнительную информацию относительно способа возврата управляемого объекта "программный модуль". Однако этот вопрос не входит в предмет рассмотрения настоящей Рекомендации.

6.1.8 Установка состояния

Как только программные средства приобретены для использования (инсталлированы), к ним можно обращаться через установку административного состояния на выполнение операции "разблокировка", ориентированную на управляемые объекты "программный модуль". Можно удалить состояние доступности заданных программных средств путем установки административного состояния управляемого объекта "программный модуль" в состояние "выключено" или "заблокировано".

6.1.9 Проверка правильности

Целостность программных средств может быть проверена с использованием операции "проверка правильности". Можно также убедиться в том, что ранее доставленные программные средства остаются в состоянии использования. Результат проверки указывает, подтверждена ли правильность программных средств. Операция "проверка правильности" выполняется через операцию, ориентированную на управляемый объект "программный модуль", где он обладает факультативными характеристиками.

6.1.10 Уведомления

Все операции, применимые к программному модулю, могут потребовать, чтобы результат был передан управляющей системе наряду с выдачей подтверждения о выполнении операции. Может оказаться также необходимым уведомить другую управляющую систему о завершении одной из этих операций, когда запрос на операцию был выдан не из этой управляющей системы.

6.1.11 Резервирование

Операция резервирования может использоваться управляющей системой для запроса резервирования заданного объекта. Операция резервирования применительно к программному управляемому объекту приводит к необходимости дублирования нижерасположенных ресурсов. Она не оказывает прямого влияния на исходные программные ресурсы.

6.1.12 Восстановление

Операция "восстановление" может использоваться управляющей системой для запроса восстановления ранее зарезервированного заданного объекта.

6.2 Взаимоотношения между управляемыми объектами

На рисунке 1 показана унаследованная иерархия управляемых объектов управления программными средствами, а на рисунке 2 показана иерархия дерева наименований управления программными средствами. Следует заметить, что системные управляемые объекты могут иметь другую взаимосвязь наименований, чем в управляемом объекте "управляемый элемент".

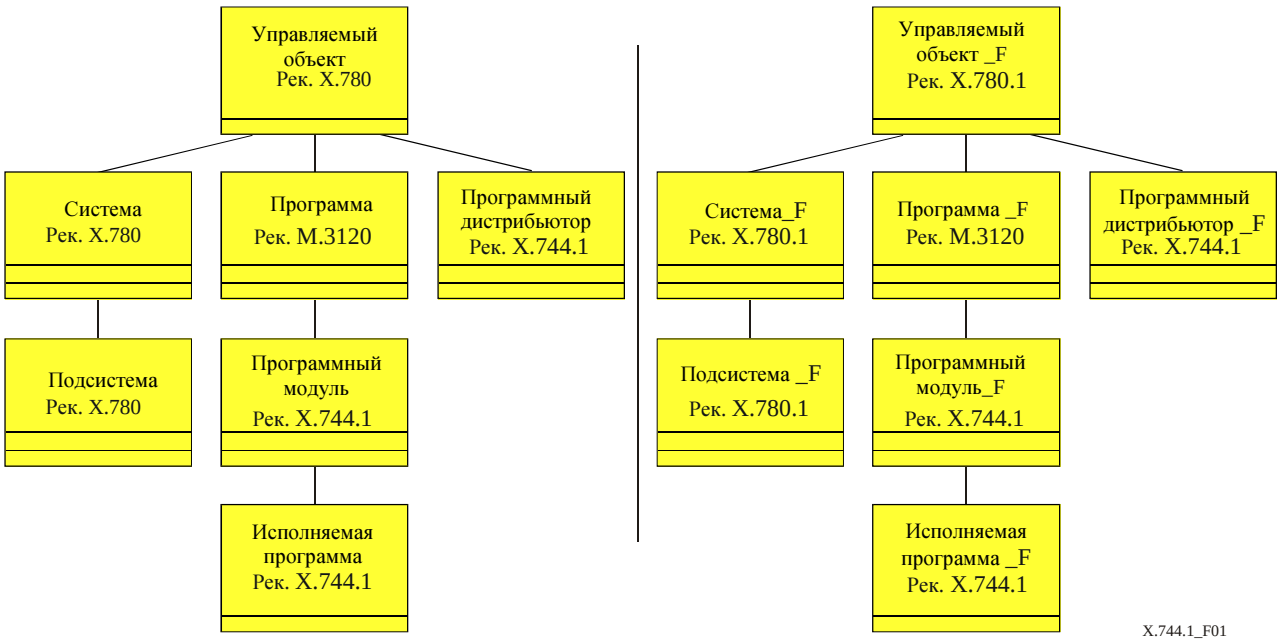

Рисунок 1/X.744.1 – Унаследованная иерархия управления программными средствами

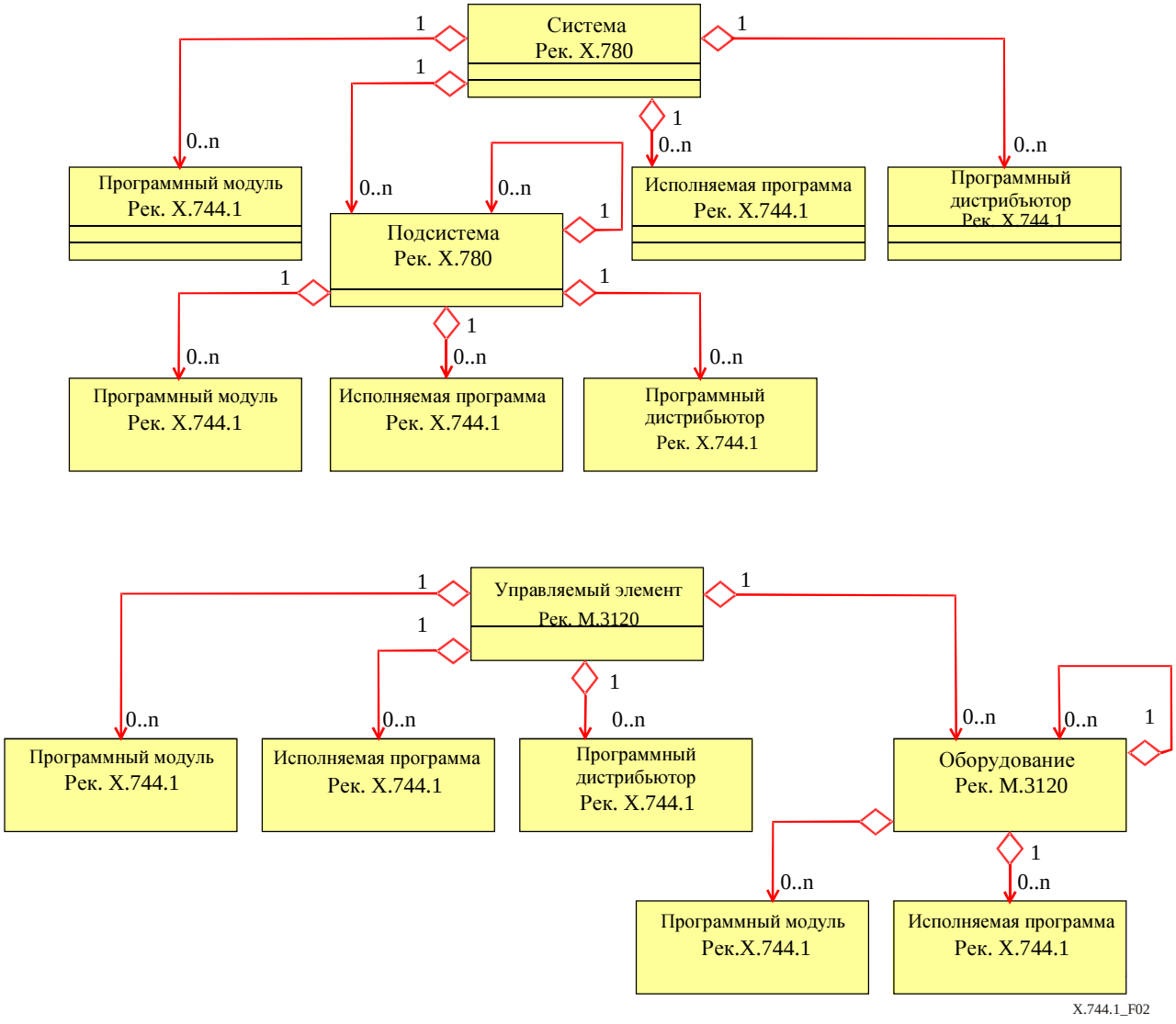

Рисунок 2/X.744.1 – Иерархия дерева наименований управления программными средствами

6.3 Управляемый объект "программный модуль"

6.3.1 Создание управляемых объектов "программный модуль"

При начальном создании управляемого объекта "программный модуль" он представляет собой ресурс, который готов принять доставленную часть программных средств.

Существуют три механизма или операции, посредством которых может быть создан управляемый объект "программный модуль":

1) Программные средства могут быть продублированы в заданной управляемой системе без выполнения какой-либо операции (например, локальными средствами).
2) Использование операции "создание" в пункте программного модуля для создания управляемого объекта "программный модуль" в состоянии "создание" или "доставка".
3) Использование операции "доставка", ориентированной на управляемый объект "дистрибьютор программных средств" в исходной системе. Это может привести к созданию управляемых объектов "программный модуль" у заданного адресата в состоянии "доставка". Операция "доставка", относящаяся к управляемому объекту "дистрибьютор программных средств", вызывает доставку программных средств заданному адресату. После того как все необходимые программные средства будут успешно доставлены, управляемый объект "дистрибьютор программных средств" выдает уведомление "результат доставки" для информирования о выполнении доставки.

Операция "создание" приводит к созданию управляемого объекта "программный модуль" у заданного адресата. Успешность создания может быть указана наличием управляемого объекта "программный модуль" у заданного адресата (может быть также выдано уведомление "создание объекта").

Будучи ранее создан, но еще не доставлен, управляемый объект "программный модуль" может выдать после завершения доставки уведомление "изменение значения атрибута", указав тем самым, что внутреннее состояние изменилось с "создано" на "доставлено". Управляемый объект "программный модуль" может быть установлен также в состояние "доставлено".

Как только программные средства войдут в состояние "доставлено", следующей стадией для программных средств должно быть "инсталлировано". При инсталляции программные средства настраиваются на использование. Например, инсталляция может охватить подготовку и настройку программного модуля на усовершенствование, возможно, создав копию текущих ресурсов программных средств и внеся изменения в копию. Инсталляция может охватить установку конфигурации и взаимоотношения зависимости между программным модулем и другими программными объектами.

Как только программные средства будут инсталлированы, они могут быть использованы другими управляемыми объектами. Может оказаться необходимым сделать их доступными для использования, установив административное состояние в значение "разблокировано", если оно находилось в заблокированном состоянии. Эта операция реализуется путем использования операции "установить административное состояние" в атрибуте "административное состояние".

Обеспечение недоступности программных средств для использования осуществляется путем их административного блокирования (т. е. установкой административного состояния в значение "заблокировано" или "выключено"). Это препятствует любым новым процессам использовать программные средства. Когда административное состояние устанавливается в значение "заблокировано", то для всех действующих в данное время процессов использование этих программных средств прерывается. Если административное состояние устанавливается в значение "выключено", то действующему в данное время процессу разрешается продолжать, а всем новым процессам не разрешается. Если выполнение всех действующих в данное время процессов, использующих данное программное средство, закончилось, административное состояние становится заблокированным. Эта операция реализуется путем использования операции "установить административное состояние" в атрибуте "административное состояние".

Операция "возврат" может использоваться для возвращения обратно результата предыдущей операции "инсталляция" либо для удаления установленной "заплаты".

Операция "проверка правильности" вызывает проверку целостности программных средств, возможно, путем привлечения алгоритма, генерирующего контрольную сумму, выполняющего проверку на вирусы и др.

Резервирование программных средств с использованием операции "резервирование" рассматривается как дублирование программных средств, например для использования в случае выхода из строя текущей версии. Резервирование программных средств может быть отменено с использованием операции "восстановление".

Следует заметить, что управляемой системе может потребоваться самой обеспечивать факультативный атрибут "налагаемые заплаты", если он не обеспечивается в классе управляемых объектов. Атрибут "налагаемые заплаты" должен обеспечиваться, если обеспечивается услуга "инсталляция" или "возврат" (даже в том случае, если атрибут "налагаемые заплаты" не входит в класс управляемых объектов).

6.3.2 Состояния управляемого объекта "программный модуль"

Управляемый объект "программный модуль" может находиться в одном из нескольких состояний в зависимости от последней операции, которая по нему выполнялась. К возможным состояниям (и их значениям) относятся:

– Создано – Доставка программного продукта не выполнена, однако некоторые произвольные ресурсы управляемой системы выделены программному модулю.

– Доставлено – Программный модуль успешно доставлен управляемой системе.

– Инсталлировано – Программные средства успешно инсталлированы в управляемой системе.

В таблице 1 показано преобразование состояний управляемого объекта "программный модуль" в состояния и значения состояний, определенные в Рекомендации МСЭ-Т X.731 [14].

Таблица 1/X.744.1 – Внутренние состояния программного модуля

Состояние программного модуля	**{Требуется инициализация} значения процедурного состояния**	**{Не инсталлировано} значение состояния доступности**	**Результирующее операционное состояние**
Создано	Имеется	Имеется	Деактивизировано
Доставлено	Отсутствует	Имеется	Деактивизировано
Инсталлировано	Отсутствует	Отсутствует	Деактивизировано или активизировано

Для управляемого объекта "программный модуль" административное состояние, операционное состояние, процедурные состояния и состояния доступности являются обязательными состояниями и атрибутами состояний, а состояние "использование" является факультативным атрибутом.

Внутренние состояния "создано", "доставлено" и "инсталлировано" взаимно исключающие, т. е. в любой заданный момент времени программный модуль может находиться только в одном из этих состояний. Разрешены только те переходы состояний, которые указаны в таблице 2. В таблице 2 указаны только те операции, которые могут влиять на состояния, другие операции, котрые не вызывают изменения состояний, исключены.

Таблица 2/X.744.1 – Матрица переходов внутренних состояний программного модуля

Матрица переходов состояний		Результирующее состояние			
		(Не существует)	**Создано**	**Доставлено**	**Инсталлировано**
Начальное состояние	(Не существует)	Изменение состояния невозможно	Создание или локальные средства[a)]	Создание, локальные средства или доставка дистрибьютору программных средств[a)]	Изменение состояния невозможно
	Создано	Удалено	–	Локальные средства[a]	Изменение состояния невозможно
	Доставлено	Удалено	Изменение состояния невозможно	–	Инсталляция или локальные средства[a)]
	Инсталлировано	Удалено	Изменение состояния невозможно	Возврат[a)]	Инсталляция, возврат или любая операция, которая не приводит к изменению состояния (т. е. резервирование, проверка правильности и восстановление)[a)]
[a)] В зависимости от поведения объекта или локальные средства.					

Независимыми от состояний "создано", "доставлено" и "инсталлировано" являются состояния "проверка правильности" и "неисправно".

Вторичными состояниями являются "проверка правильности" и "неисправно", которые могут иметь место дополнительно к внутренним состояниям. Состояние "проверка правильности" преобразуется в значение "состояние доступность", которое включает значение {в процессе тестирования}. Управляемый объект программного модуля может войти в состояние "проверка правильности" и выйти из него независимо от значений его других состояний. Например, в случае неисправности управляемый объект "программный модуль" может находиться в состоянии "проверка правильности" (см. таблицу 3).

Таблица 3/X.744.1 – Состояние "проверка правильности" программного модуля

Состояние программного модуля	**{В процессе тестирования} значение состояния "доступность"**	**Результирующее операционное состояние**
Проверка правильности	Имеется	Деактивизировано или активизировано

Находясь в состоянии "создано", "доставлено" или "инсталлировано", управляемый объект "программный модуль" может находиться также в состоянии "неисправное". Состояние "неисправное" преобразуется в значение "состояние доступность", включая значение {"неисправное"}. Конкретные причины входа и выхода из состояния "неисправное" являются локальным вопросом. Управляемый объект "программный модуль" может входить в состояние "неисправное" или выходить из него независимо от значений его других состояний. Например, побочным результатом операции "проверка правильности" может быть переход в состояние "неисправное" из состояния "проверка правильности" либо программный модуль может выйти из состояния "неисправное" в результате операции "возврат" (см. таблицу 4).

Таблица 4/X.744.1 – Состояние "неисправное" программного модуля

Состояние программного модуля	**{Неисправное} значение "состояния доступности"**	**Результирующее операционное состояние**
Неисправное	Имеется	Деактивизировано

6.3.3 Операции над управляемым объектом "программный модуль"

С управляемым объектом "программный модуль" связано большое число операций, используемых для изменения его состояния.

К этим операциям относятся:

– Резервирование – вызывает резервирование программных средств для заданного адресата.

– Создание – приводит к тому, что в управляемой системе начинает существовать новый управляемый объект "программный модуль".

– Удаление – приводит к удалению управляемого объекта "программный модуль" из управляемых объектов и может иметь в качестве дополнительного результата удаление соответствующих ресурсов.

– Инсталляция – подготавливает программные средства для использования.

– Восстановление – приводит к восстановлению зарезервированных программных средств у заданного адресата.

– Возврат – вызывает возврат применения "заплаты" или инсталляции.

– Проверка правильности – проверяет целостность программных средств.

Операции "создание" и "удаление" преобразуются в стандартные операции "создание" и "удаление", определенные в Рекомендации МСЭ-Т X.780 [24], остальные же операции преобразуются в другие операции. Кроме того, могут быть запрошены и модифицированы значения атрибутов с использованием операций "получить" и "установить", определенных в Рекомендации МСЭ-Т X.780.

6.3.3.1 Услуга резервирования

Услуга резервирования используется управляющей системой для того, чтобы запросить выполнение резервирования информации, представленной заданным экземпляром объекта (т. е. управляемый объект, представляющий программные средства, резервируется).

Эта услуга использует операцию резервирования в классе управляемых объектов "программный модуль". Операция резервирования может выполняться независимо от состояния класса управляемых объектов "программный модуль". Параметр "адресат резервирования" указывает адресата, для которого должна резервироваться информация. Возможными адресатами могут быть:

– Локальный управляемый объект – в этом случае операция резервирования может выполняться внутри управляемой системы. Информация должна резервироваться для представленного экземпляра управляемого объекта.

– Удаленная система – в этом случае зарезервированная информация должна быть передана в автономном режиме удаленной системе локальными средствами.

Могут быть выделены следующие исключения:

– Рекомендация МСЭ-Т X.780 [24] **Ошибка прикладной программы**.

– **НЕТпакетаРезервнойИнформации** – если пакет резервной информации не предусмотрен в данном экземпляре.

– **ОшибкаОбработкиПрограммногоРезервирования** – недействительный параметр "адресат резервирования" для операции резервирования.

– **ОшибкаЗапросаОдновременныхОпераций** – управляемый объект "программный модуль" этого класса уже задержал запрос операции резервирования или восстановления (либо другой операции). Критерий обеспечения одновременных запросов для конкретного класса управляемых объектов "программный модуль" зависит от системы.

Поскольку запрос резервирования может занять значительное время, операция резервирования должна выдаваться немедленно. После выполнения резервирования заданного объекта должно быть выдано уведомление "отчет о резервировании".

6.3.3.2 Услуга инсталляции

Услуга инсталляции используется управляющей системой, чтобы дать инструкцию управляемой системе инсталлировать экземпляр объекта "доставленный или инсталлированный программный модуль". В применимых случаях услуга инсталляции может обновить значение атрибута "введенные заплаты".

Атрибут "заданные программные средства" должен указывать источник подлежащих инсталляции программных средств. С точки зрения вводимых заплат этот источник должен быть уникальным. Этим источником может один или несколько из следующих объектов:

– Идентификатор "заплаты" – специфичный для системы идентификатор.
– Указатель "заплаты" – класс (или подкласс) управляемого объекта "программный модуль".

Операция "инсталляция" должна выдавать значение атрибута "введенные заплаты" того экземпляра объекта "программный модуль", на который ориентирована услуга.

Могут быть выделены следующие исключения:

– Рекомендация МСЭ-Т X.780 [24] **Ошибка прикладной программы**.
– **НЕТпакетаИнсталляции** – если пакет инсталляции не предусмотрен в данном экземпляре.
– **ОшибкаОбработкиИнсталляцииПрограммногоСредства** – недействительный параметр "информация инсталляции" операции "инсталляция".
– **НеправильноеОперационноеСостояние –** если программный модуль не может быть инсталлирован по причине недействительного условия состояния. Могут быть инсталлированы только те управляемые объекты "программный модуль", которые находятся в состоянии "доставлено" или "инсталлировано". Кроме того, операционное состояние должно быть активизировано, административным состоянием должно быть "разблокировано", а состоянием "использование" должно быть "активное" или "холостое".
– **ОшибкаЗапросаОдновременныхОпераций** – Управляемый объект программного модуля этого класса уже задержал запрос операции "инсталляция" или "возврат" (либо другой операции). Критерий обеспечения одновременных запросов для конкретного класса управляемых объектов "программный модуль" зависит от системы.

6.3.3.3 Услуга "восстановление"

Услуга "восстановление" используется управляющей системой для выполнения запроса на восстановление информации, представленной заданным экземпляром объекта. Это восстановление от предыдущего резервирования.

Эта услуга восстановления использует операцию "восстановление" в классе управляемых объектов "программный модуль". Операция "восстановление" может выполняться независимо от состояния класса управляемых объектов "программный модуль". Параметр "источник восстановления" должен указывать адресат из перечисленных ниже, для которого должна быть восстановлена информация:

– Локальный управляемый объект – локальный управляемый объект того же класса, что и класс данной используемой операции. В этом случае операция "восстановление" может выполняться внутри управляемой системы.
– Удаленная система – в этом случае информация восстановления может передаваться из удаленной системы в автономном режиме некоторыми локальными средствами.

Поведение конкретного экземпляра программного модуля или локальных средств будет определять состояние программного модуля при его восстановлении.

Могут быть выделены следующие исключения:

– Рекомендация МСЭ-Т X.780 [24] **Ошибка прикладной программы**.
– **НЕТпакетаИнформацииПакетаВосстановления** – если пакет восстановления информации не предусмотрен в данном экземпляре.
– **ОшибкаОбработкиПрограммногоВосстановления** – недействительный параметр "источник ресурса" для операции восстановления.

– **ОшибкаЗапросаОдновременныхОпераций** – Управляемый объект "программный модуль" этого класса уже задержал запрос операции резервирования или восстановления (либо другой операции). Критерий обеспечения одновременных запросов для конкретного класса управляемых объектов "программный модуль" зависит от системы.

Поскольку запрос восстановления может занять значительное время, операция "восстановление" должна выдаваться немедленно. После выполнения восстановления заданного объекта должно быть выдано уведомление "отчет о восстановлении".

6.3.3.4 Услуга "возврат"

Услуга "возврат" используется управляющей системой (например, ОС) для того, чтобы выдать инструкцию управляемой системе вернуть в прежнее состояние "заплату" или набор "заплат" программных средств, представленных управляемым объектом "программный модуль".

Эта услуга использует операцию "возврат" в классе управляемых объектов "программный модуль". Параметр "информация возврата" должен указывать одну или несколько ранее использованных "заплат", которые были инсталлированы. Каждый идентификатор использованной "заплаты" представляет собой выбор одного из специфичных для системы идентификаторов или экземпляров объекта "программный модуль" в зависимости от значений, установленных вначале в предыдущей(их) операции(ях) "инсталляция" относительно данного экземпляра объекта "программный модуль".

Операция "возврат" должна установить значение атрибута "используемые заплаты" экземпляра объекта "программный модуль", на который была ориентирована услуга.

Если услуга "возврат" успешно вернула в предыдущее состояние все "заплаты", которые были перед этим инсталлированы (т. е. атрибут "введенные заплаты" пустой), то внутреннее состояние программного модуля будет изменено с "инсталлировано" на "доставлено".

Могут быть выделены следующие исключения:

– Рекомендация МСЭ-Т X.780 [24] **Ошибка прикладной программы**.
– **НЕТпакетаВозврата** – если пакет "возврат" не предусмотрен в данном экземпляре.
– **НеправильноеОперационноеСостояние** – Программный модуль находится в недействительном состоянии для операции "возврат". Для выполнения операции "возврат" программный модуль должен находиться во внутреннем состоянии "инсталлировано", административным состоянием должно быть "разблокировано", операционным состоянием должно быть "активизировано", а состоянием "использование" должно быть "активное" или "холостое".
– **ОшибкаОбработкиПрограммногоВозврата** – недействительный параметр "возврат информации" для операции "возврат".
– **ОшибкаЗапросаОдновременныхОпераций** – Управляемый объект "программный модуль" этого класса уже задержал запрос операции инсталляции или возврата (либо другой операции). Критерий обеспечения одновременных запросов для конкретного класса управляемых объектов программного модуля зависит от системы.

6.3.3.5 Услуга "проверка правильности"

Эта услуга использует операцию "проверка правильности" в классе управляемых объектов "программный модуль". Параметр "информация проверки правильности" должен указывать тип необходимой проверки. К возможным типам относятся:

– Зарегистрированная проверка правильности – в этом случае информация проверки правильности обеспечивается через другой заданный управляемый объект.
– Проверка правильности по умолчанию – в этом случае должно быть использовано значение проверки правильности по умолчанию для данного конкретного экземпляра управляемого объекта.

Могут быть выделены следующие исключения:

– Рекомендация МСЭ-Т X.780 [24] **Ошибка прикладной программы**.
– **НЕТпакетаПроверкиПравильности** – если пакет "проверка правильности" не предусмотрен в данном экземпляре.
– **ОшибкаОбработкиПроверкиПравильностиПрограммногоСредства** – один или несколько недействительных аргументов операции "проверка правильности".

– **ОшибкаОперационногоСостояния** – программный модуль находится в недействительном состоянии для операции "проверка правильности". Для выполнения операции "проверка правильности" внутренним состоянием программного модуля должно быть "доставлено" или "инсталлировано".
– **ОшибкаЗапросаОдновременныхОпераций** – Управляемый объект "программный модуль" этого класса уже задержал запрос операции "проверка правильности" (либо другой операции). Критерий обеспечения одновременных запросов для конкретного класса управляемых объектов "программный модуль" зависит от системы.

Поскольку запрос проверки правильности может занять значительное время, операция "восстановление" должна выдаваться немедленно. После выполнения проверки правильности заданного объекта должно быть выдано уведомление "отчет проверки правильности".

6.3.4 Уведомления управляемого объекта "программный модуль"

Управляемый объект "программный модуль" получает следующие уведомления, включая уведомления, унаследованные из управляемого объекта программных средств [2] (см таблицу 5):

Таблица 5/X.744.1 – Уведомления программного модуля

Уведомление	Ссылка	Условный пакет (если "условный")
Изменено значение атрибута	[24]	"itut_m3120::пакетУведомленияИзмененоЗначениеАтрибута"
Отчет о резервировании	6.3.4.1	"itut_x744d1::пакетРезервированиеИнформации" и/или "itut_x744d1::пакетАворезервированиеИнформации"
Создан объект	[24]	"itut_m3120::созданиеПакетаУведомленийОбУдалениях"
Удален объект	[24]	"itut_m3120:: созданиеПакетаУведомленийОбУдалениях"
Ошибка обработки	[24]	Обязательно
Отчет о восстановлении	6.3.4.2	"itut_x744d1:: пакетВосстановлениеИнформации" и/или "itut_x744d1:: пакетАвтовосстановлениеИнформации"
Изменено состояние	[24]	"itut_m3120:: пакетУведомлениеОбИзмененииСостояния"
Отчет о проверке правильности	6.3.4.3	"itut_x744d1:: пакетПроверкиПравильности"

Изменения перечисленных ниже атрибутов (если они определены) могут привести к выдаче следующих уведомлений (если они предусмотрены) об изменениях значений атрибутов:

– Объекты, испытывающие воздействие;
– Аварийное состояние;
– Введенные заплаты;
– Состояние доступности;
– Список текущих проблем;
– Адресат будущего авторезервирования;
– Предельное значение запуска будущего авторезервирования;
– Будущее автовосстановление разрешено;
– Источник будущего автовосстановления;
– Процедурное состояние;
– Метка пользователя;
– Версия.

Изменения перечисленных ниже состояний (если они определены) могут привести к выдаче уведомлений об изменении состояния:

– Операционное состояние;
– Административное состояние;
– Состояние использования.

6.3.4.1 Уведомление "отчет о резервировании"

Уведомление "отчет о резервировании" выдается для информирования о резервировании управляемого объекта. Резервирование может быть инициировано автоматически (в соответствии с критерием, установленным в атрибутах "предельное значение запуска будущего авторезервирования" и "будущее авторезервирование") через запрос управления (посредством операции резервирования) либо по инициативе управляемой системы.

Адресат резервирования может быть локальным (т. е. резервирование другого объекта "программный модуль" в локальной управляемой системе) либо это может быть удаленная система с обращением к ней в автономном режиме путем использования конкретного протокола передачи файлов (например, FTAM). О результате резервирования сообщается в этом уведомлении.

6.3.4.2 Уведомление "отчет о восстановлении"

Уведомление "отчет о восстановлении" выдается для информирования о восстановлении управляемого объекта после предыдущего резервирования. Восстановление может быть инициировано автоматически (в соответствии с атрибутами "тип источника будущего автовосстановления" и "тип допустимого будущего автовосстановления" и со специфичным для системы критерием) через запрос управления (посредством операции "восстановление") либо оно может быть инициировано управляемой системой.

Источник восстановления может быть локальным (т. е. восстановление осуществляется от другого объекта "программный модуль" в локальной управляемой системе) либо это может быть удаленная система с обращением к ней в автономном режиме путем использования конкретного протокола передачи файлов (например, FTAM). О результате восстановления сообщается в этом уведомлении.

6.3.4.3 Уведомление "отчет о проверке правильности"

Уведомление "отчет о проверке правильности" выдается для информирования о результатах выполнения операции "проверка правильности".

6.4 Управляемый объект "исполняемая программа"

Класс управляемых объектов "исполняемая программа" является подклассом класса управляемых объектов "программный модуль" (см. 6.3.1) с дополнительными характеристиками, описывающими набор функций при исполнении программы. Исполняемая программа представляет собой программу, которая может выполняться некоторым способом, например локальными средствами или дистанционно через операцию "исполнение". Имеется возможность вызвать выполнение исполняемой программы командой управления; однако такая возможность выполнения программы может быть осуществлена только локальным действием.

Несмотря на ограниченную область распространения настоящей Рекомендации, данная модель не препятствует включению в подклассы исполняемой программы такой информации, как наличие у данной программы одного или нескольких пользователей, условия активности или занятости программных средств или допустимость подключения максимального количества пользователей.

6.4.1 Дополнительные состояния для управляемого объекта "исполняемая программа"

Состояние "использование", определенное в Рекомендации МСЭ-Т X.731 [14], указывает, используется ли в данное время исполняемая программа. Для класса управляемых объектов "исполняемая программа" состояние "использование" является обязательным атрибутом помимо атрибутов, которые уже имеются в управляемом объекте "программный модуль" (см. 6.3.2). В классе управляемых объектов "исполняемая программа" для состояния "использование" разрешены значения "холостое", "активное" и "занято".

В состоянии "использование" значение "холостое" означает отсутствие активных исполнений. Значения "активное" и "занято" специфичны для реализации и не входят в предмет рассмотрения настоящей Рекомендации. Например, значение "активное" может означать наличие некоторого действующего исполнения, а значение "занято" может означать, что программа достигла своих

максимальных возможностей и любой запрос на исполнение (путем использования услуги "программа исполнения" или выданный некоторыми локальными средствами) может быть отклонен или поставлен в очередь для последующего исполнения. Спецификации могут предпочесть ограничить количество пользователей для достижения максимальной пропускной способности.

6.4.2 Дополнительные операции для управляемого объекта "исполняемая программа"

Ниже перечислены операции над управляемым объектом "исполняемая программа" помимо операций, определенных для управляемого объекта "программный модуль" (см. 6.3.3):

– Исполняемая программа – вызывает исполнение программного управляемого объекта.

Операция "исполняемая программа" используется для инициализации выполнения программы, представленной исполняемой программой. Для выполнения исполняемой программы она должна быть инсталлирована.

Поведение конкретного экземпляра исполняемой программы или локальных средств будет определять состояние исполняемой программы при ее выполнении.

6.4.2.1 Услуга "исполняемая программа"

Услуга "исполняемая программа" используется управляющей системой для инициализации выполнения программы, представленной объектом "исполняемая программа".

Параметр "дополнительная информация" должен указывать параметры, используемые при выполнении программы.

После успешной инициализации "ответ исполняемой программы" может указывать:

– Идентификатора процесса.
– Владельца процесса.
– Начальное стартовое время.
– Обеспечиваемые параметры дополнительной информации.

Могут быть выделены следующие исключения:

– Рекомендация МСЭ-Т X.780 [24] **Ошибка прикладной программы**.
– **НЕТпакетаИсполняемойПрограммы** – если пакет исполняемой программы не предусмотрен в данном экземпляре.
-- **ОшибкаОперационногоСостояния** – исполняемая программа находится в недействительном состоянии для операции "выполнение программы". Для выполнения операции "выполнение программы" исполняемая программа должна находиться во внутреннем состоянии "инсталлировано", административное состояние должно быть в значении "разблокировано", операционное состояние должно быть "активизировано", а состоянием "использование" должно быть "активное" или "холостое".
– **ОшибкаОбработкиИсполняемойПрограммы** – один или несколько недействительных аргументов операции "выполнение программы".

6.5 Управляемый объект "дистрибьютор программных средств"

Управляемый объект "дистрибьютор программных средств" – это статический объект, представляющий механизм(ы) доставки управляемой системы. Это – управляемый объект, который распространяет программные средства по заданной управляемой системе при получении им операции "доставка" из управляющей системы. Параметры операции "доставка" могут использоваться для указания набора программных средств, подлежащих доставке, заданного адресата доставки и для выбора механизма передачи. Несмотря на то что этот класс объектов может использоваться для инициализации доставки через различные механизмы передачи, он не является моделью ни одного из таких механизмов передачи. Эти модели оставлены для дальнейшей специализации.

При завершении распространения управляемый объект выдает уведомление "результат доставки" с результатом распространения.

6.5.1 Операции над управляемым объектом "дистрибьютор программных средств"

Ниже перечислены операции над управляемым объектом "дистрибьютор программных средств":

– Создание – приводит к созданию нового управляемого объекта "дистрибьютор программных средств". Должно выдаваться уведомление "создан объект".

– Доставка – побуждает управляемый объект "дистрибьютор программных средств" создать конкретные программные средства (методом, не входящим в предмет рассмотрения настоящей Рекомендации) в заданной управляемой системе и в качестве побочного результата – выделить соответствующие ресурсы, которые относятся к тем управляемым объектам "программный модуль", которые должны быть созданы в заданном управляемом пункте.

– Удаление – приводит к удалению управляемого объекта "дистрибьютор программных средств" из управляемой системы. Должно выдаваться уведомление "удален объект".

6.5.1.1 Услуга доставки

Услуга доставки используется управляющей системой для запроса распространения программных средств или набора программных средств. Информация операции "доставка" идентифицирует программные средства, подлежащие распространению. Результат операции "доставка" состоит в том, что копия заданных элементов программных средств доставляется заданной системе при внутреннем состоянии "доставка".

Пакетирование программных средств и выбор механизма передачи являются локальным вопросом и не входят в предмет рассмотрения настоящей Рекомендации. Например, эта информация может быть предварительно сконфигурирована в операции "доставка" наряду с другой уместной информацией.

Результат успешного выполнения этого запроса состоит в том, что программные средства, подлежащие распространению, копируются в заданную систему; это может привести к созданию объектов программного модуля и/или исполняемой программы. После выполнения команды выдается уведомление "результат доставки".

При выполнении операции "доставка" используются следующие параметры:

– Идентификатор доставки – этот факультативный параметр указывает уникальный идентификатор данной операции "доставка".

– Заданные программные средства – указывает источник программных средств, подлежащих доставке.

– Заданная система – указывает факультативный заданный адресат программных средств, подлежащих доставке. Если этот параметр не указывает заданного адресата, система использует локальные средства для определения заданного адресата.

– Информация передачи – специфичный для прикладной программы механизм передачи.

– Дополнительная информация – дополнительная специфичная для прикладной программы информация.

Могут быть выделены следующие исключения:

– Рекомендация МСЭ-Т X.780 [24] **Ошибка прикладной программы**.

– **ОшибкаОперационногоСостояния** – при выполнении операции "доставка" дистрибьютор программных средств находится в недействительном состоянии. Для выполнения операции "доставка" дистрибьютор программных средств должен находиться в административном состоянии "разблокировано" и в операционном состоянии "активизировано".

– **ОшибкаОбработкиПрограммныхСредств** – один или несколько недействительных аргументов операции "доставка".

6.5.2 Уведомления управляемого объекта "дистрибьютор программных средств"

Управляемый объект "дистрибьютор программных средств" имеет следующие уведомления (см. таблицу 6):

Таблица 6/X.744.1 – Уведомления дистрибьютора программных средств

Уведомление	Ссылка	Условный пакет (в случае "условный")
Результат доставки	6.5.2.1	Обязательно
Создание объекта	[24]	Обязательно
Удаление объекта	[24]	Обязательно
Изменение состояния	[24]	Обязательно

Изменения следующих состояний могут привести к выдаче уведомления "изменение состояния":

– операционное состояние;

– административное состояние.

6.5.2.1 Уведомление "результат доставки"

Уведомление "результат доставки" выдается управляемым объектом при завершении операции "доставка". Оно содержит окончательные результаты операции и может указывать успешность или безуспешность выполнения.

6.6 Использование универсальных идентификаторов (UID)

Настоящая Рекомендация использует универсальные идентификаторы (UID) в соответствии с положениями Рекомендации МСЭ-Т X.780 [24]. Универсальные идентификаторы позволяют осуществлять локальные расширения как констант, так и значений данных. Примерами их использования может служить разрешение на использование локально определяемых аргументов в операции "исполняемая программа" и на расширение перечисленных типов файла "тип файла".

Универсальные идентификаторы используют тип UID, тип расширения управления и типы атрибутов "тип набора дополнительной информации". Синтаксис IDL для этих типов (из Рекомендации МСЭ-Т X.780) имеет вид:

```
struct UIDType {
  string moduleName; // module where value is defined
  short value; // constant within the module
};

struct ManagementExtensionType {
  UIDType id; // identifies the type of info
  any info; // type will depend on id
};

typedef sequence <ManagementExtensionType> AdditionalInformationSetType;
```

Для осуществления локального расширения универсального идентификатора в новом IDL должны быть определены постоянные модули UID. Постоянные модули UID будут содержать постоянное значение для каждой обеспечиваемой константы (с типом UID) или типа данных (с типом расширения управления и типом набора дополнительной информации). Примеры постоянных модулей, используемых в этой модели, приведены в 9.19 и 9.20.

Подробное описание универсальных идентификаторов приведено в Рекомендации МСЭ-Т X.780 .

6.7 Взаимоотношения

Идентифицированы многие взаимоотношения между программными управляемыми объектами и взаимоотношения между программными управляемыми объектами и другими управляемыми объектами. К ним относятся:

– Зависимость – это взаимоотношение может использоваться для моделирования того факта, что один программный управляемый объект зависит некоторым образом от наличия другого программного управляемого объекта. Такие взаимоотношения могут использоваться для моделирования "заплат".

– Конфигурация – это взаимоотношение может использоваться для отражения того факта, что один управляемый объект "программный модуль" может повлиять на поведение другого управляемого объекта "программный модуль". Например, дополнительный шрифт можно смоделировать как взаимоотношение конфигураций. Это взаимоотношение может быть также использовано для моделирования расширений и "заплат".

– Использование – это взаимоотношение может использоваться для того, чтобы показать, какие другие управляемые объекты используют программные управляемые объекты. Такие управляемые объекты, вероятно, должны представлять процессы, протекающие в управляемой системе.

Неисправность управляемого объекта может вызвать также неисправность в любом зависимом от него управляемом объекте. Однако она не должна привести управляемые объекты к нарушениям взаимоотношений с конфигурацией (хотя их поведение может измениться). Обнаружение неисправных программных средств может потребовать доставки и инсталляции новой копии программных средств или доставки и инсталляции обновленной версии.

Однако спецификация этих взаимоотношений не входит в предмет рассмотрения настоящей Рекомендации, поскольку эти взаимоотношения зависят от применения.

7 Взаимоотношения с другими функциями

Другие "функции управления систем" обеспечивают следующие функции:

– Рабочие характеристики программных средств, рассматриваемые в Рекомендации МСЭ-Т Q.822.1 [8];

– Программные средства аудиторской проверки, охватываемые функцией "отслеживание проверки защиты" (см. Рекомендацию МСЭ-Т X.740 [15]);

– Обеспечение защиты программных средств, охватываемое объектами и атрибутами управления доступом (см. Рекомендацию МСЭ-Т X.741 [16]);

– Учет использования программных средств, охватываемый функцией "измерение использования" для целей учета (см. Рекомендацию МСЭ-Т X.742 [17]);

– Тестирование программных средств (включая инсталляцию среды тестирования, тестовый прогон программных средств, установку контрольных точек, а также приостановку и возобновление функциональной среды тестирования программных средств), охватываемых функцией управления тестированием (см. Рекомендацию МСЭ-Т X.745 [22]);

– Планирование программных функций и операций, охватываемых функцией "планирование" (см. Рекомендацию МСЭ-Т X.746 [23]).

8 Согласованность и соответствие

В этом разделе определяется критерий, которому должны удовлетворять другие стандарты, претендующие на соответствие настоящей Рекомендации, и функции, которые должны быть реализованы в системах, претендующих на соответствие настоящей Рекомендации.

8.1 Системное соответствие

8.1.1 Точки соответствия

В этом подразделе описываются точки соответствия, которые должны обеспечиваться системами, претендующими на соответствие настоящей спецификации:

1) Реализация, претендующая на соответствие этим требованиям, должна:
   - обеспечивать либо:
     - профиль базового соответствия по Рекомендации МСЭ-Т Q.816 [3]. В этом случае каждый экземпляр управляемого объекта должен быть инсталлирован объектом CORBA; либо
     - профиль базового соответствия по Рекомендации МСЭ-Т Q.816.1 [7]. В этом случае каждый экземпляр управляемого объекта должен быть инсталлирован объектом Facade CORBA (см. Рекомендацию МСЭ-Т Q.816.1 и Рекомендацию МСЭ-Т X.780.1 [28]);
   - обеспечивать либо:
     - требования к программному модулю (без обеспечения исполняемой программы или дистрибьютора программных средств); либо
     - требования к исполняемой программе (без обеспечения программного модуля или дистрибьютора программных средств); либо
     - требования к дистрибьютору программных средств (без обеспечения программного модуля или исполняемой программы). Это может иметь место до доставки программных средств; либо
     - требования к программному модулю и к исполняемой программе (без обеспечения дистрибьютора программных средств); либо
     - требования к программному модулю и к дистрибьютору программных средств (без обеспечения исполняемой программы); либо
     - требования к исполняемой программе и к дистрибьютору программных средств (без обеспечения программного модуля); либо
     - требования к программному модулю, к исполняемой программе и к дистрибьютору программных средств;
   - использовать модули IDL, перечисленные в разделе 9.
2) Реализация, претендующая на соответствие требованиям к программному модулю, должна:
   - обеспечивать управляемый объект "программный модуль", определенный в 6.3;
   - обеспечивать создание по меньшей мере одного управляемого объекта класса управляемых объектов "программный модуль".
3) Реализация, претендующая на соответствие требованиям к исполняемой программе, должна:
   - обеспечивать управляемый объект "исполняемая программа", определенный в 6.4.
   - обеспечивать создание по меньшей мере одного управляемого объекта класса управляемых объектов "исполняемая программа".
4) Реализация, претендующая на соответствие требованиям к дистрибьютору программных средств, должна:
   - обеспечивать управляемый объект "дистрибьютор программных средств", определенный в 6.5;
   - обеспечивать создание по меньшей мере одного управляемого объекта класса управляемых объектов "дистрибьютор программных средств".

8.2 Руководства по заявке о соответствии

Пользователи этого основополагающего документа должны быть внимательны при составлении заявки о соответствии. Поскольку модули IDL используются в виде пространства наименований, они могут, согласно правилам OMG IDL, быть разделены по файлам. Таким образом, при расширении модуля его имя не должно меняться. Вместо этого просто добавляется новый файл IDL. Следовательно, простого указания имени модуля в заявке о соответствии будет недостаточно для

идентификации набора интерфейсов IDL. Заявка о соответствии должна идентифицировать документ и год его публикации, чтобы идентифицировать правильную версию IDL.

9 Листинг модулей IDL по Рекомендации МСЭ-Т X.744.1

```
#ifndef _itut_x744_1_idl_
#define _itut_x744_1_idl_

#include <itut_x780.idl>
#include <itut_x780_1.idl>
#include <itut_x780ct.idl>
#include <itut_m3120.idl>

#pragma prefix "itu.int"

/**
Этот код IDL (начиная со строки "#ifndef … " и до конца этого раздела) предназначен
для хранения в файле под названием "itut_x744_1.idl", расположенном на пути поиска,
используемого компилятором IDL в вашей системе. Должен использоваться компилятор,
который обеспечивает версию CORBA, определенную в Рекомендации МСЭ-Т Q.816.
*/
/**
Этот модуль itut_x744d1 содержит определение интерфейса IDL по Рекомендации МСЭ-Т
X.744. Определения IDL в этом файле – это интерфейсы объектов.
*/

module itut_x744d1
{

/**
```

9.1 Импорты

```
*/
     /**
     Типы, импортированные из Рекомендации МСЭ-Т X.780
     */

     typedef itut_x780::AdditionalInformationSetType AdditionalInformationSetType;
     typedef itut_x780::AdministrativeStateType AdministrativeStateType;
     typedef itut_x780::ApplicationErrorInfoType ApplicationErrorInfoType;
     typedef itut_x780::AvailabilityStatusSetType AvailabilityStatusSetType;
     typedef itut_x780::DeletePolicyType DeletePolicyType;
     typedef itut_x780::ExternalTimeType ExternalTimeType;
     typedef itut_x780::GeneralizedTimeType GeneralizedTimeType;
     typedef itut_x780::Istring Istring;
     typedef itut_x780::ManagementExtensionType ManagementExtensionType;
     typedef itut_x780::MONameType MONameType;
     typedef itut_x780::NameBindingType NameBindingType;
     typedef itut_x780::NullType NullType;
     typedef itut_x780::OperationalStateType OperationalStateType;
     typedef itut_x780::ProceduralStatusSetType ProceduralStatusSetType;
     typedef itut_x780::StringSetType StringSetType;
     typedef itut_x780::UIDType UIDType;
     typedef itut_x780::UsageStateType UsageStateType;

     /**
     Типы, импортированные из Рекомендации МСЭ-Т M.3120
     */
```

```
    typedef itut_m3120::AlarmStatusType AlarmStatusType;
    typedef itut_m3120::AlarmSeverityAssignmentProfileNameType
            AlarmSeverityAssignmentProfileNameType;
    typedef itut_m3120::ArcProbableCauseSetType ArcProbableCauseSetType;
    typedef itut_m3120::ArcIntervalProfileNameType ArcIntervalProfileNameType;
    typedef itut_m3120::ArcTimeType ArcTimeType;

/**
```

9.2 Прямые декларации

```
*/
    /**
    Прямые декларации интерфейсов
    */

    interface ExecutableSoftware;
    interface ExecutableSoftware_F;
    interface ExecutableSoftwareFactory;
    interface SoftwareDistributor;
    interface SoftwareDistributor_F;
    interface SoftwareDistributorFactory;
    interface SoftwareUnit;
    interface SoftwareUnit_F;
    interface SoftwareUnitFactory;

    /**
    Прямые декларации типов значений
    */

    valuetype ExecutableSoftwareValueType;
    valuetype SoftwareDistributorValueType;
    valuetype SoftwareUnitValueType;

/**
```

9.3 Структуры и определения типов

```
*/

    /**
    Patch ::= CHOICE {
      patchId GraphicString, -- идентификатор, специфичный для системы --
      patchPointer ObjectInstance } -- класса объектов "программный модуль"  -
    AppliedPatches ::= SEQUENCE OF Patch
    */

    enum PatchChoice
    {
      patchIdChoice, // идентификатор, специфичный для системы
      patchPointerChoice // класса объектов "программный модуль" (или "исполняемая
                         // программа")
    };

    union PatchType switch (PatchChoice)
    {
      case patchIdChoice:
         Istring patchId;
      case patchPointerChoice:
         MONameType patchPointer;
    };
```

```
/**
Тип атрибута "введенная заплата"
*/

typedef sequence <PatchType> AppliedPatchesSeqType;

/**
BackupDestination ::= CHOICE {
  localObject ObjectInstance,
  inLine NULL, -- встроено в уведомление о дополнительной информации --
  offLine GraphicString -- удаленной системой, например FTAM --}

Встроенный вариант не определяется в Рекомендации МСЭ-Т X.744.1
*/

enum BackupDestinationChoice
{
  localObjectChoice,
  offLineChoice // удаленной системой, например FTAM
};

union BackupDestinationType switch (BackupDestinationChoice)
{
  case localObjectChoice:
     MONameType localObject;
  case offLineChoice:
     Istring offLine;
};

/**
CheckSum ::= BIT STRING

Тип атрибута "контрольная сумма". Алгоритм вычисления контрольной суммы
определяется локально
*/

typedef long CheckSumType;

/**
Тип атрибута "дата создания"
*/

typedef GeneralizedTimeType DateOfCreationType;

/**
Date ::= CHOICE {
  time   GeneralizedTime,
  noSuchInformationNULL}
*/

enum DateChoice
{
  timeChoice,
  noSuchInformationChoice
};

union DateType switch (DateChoice)
{
  case timeChoice:
     GeneralizedTimeType time;
  case noSuchInformationChoice:
     NullType noInformation;
};
```

```
/**
Тип атрибута "местоположение файла"
*/

typedef sequence <Istring> FileLocationSetType;

/**
FileType ::= INTEGER{
  unstructuredText (0), -- FTAM-1
  unstructuredBinary (1), -- FTAM-3
  blockSpecial (2)}
```

"Тип файла" преобразован в UIDType, который первоначально основан на модуле FileTypeConst. Это дает возможность приложениям добавлять собственные типы файлов

```
Тип атрибута "тип файла"
*/

typedef UIDType FileTypeType;

/**
Тип атрибута "порог пуска будущего авторезервирования"

Следует заметить, что этот гибкий тип используется для пороговых типов,
соответствующих Рекомендации МСЭ-Т Q.822.1, но это не истинный порог Q.822.1
*/

typedef float FutureAutoBackupTriggerThresholdType;

/**
Тип атрибута "будущее допусимое авторезервирование". ИСТИННО означает, что оно
разрешено
*/

typedef boolean FutureAutoRestoreAllowedType;

/**
AutoRestoreSource ::= CHOICE {
  localObject ObjectInstance,
  remoteSystem GraphicString – независимо от удаленной системы
}
*/

enum AutoRestoreSourceChoice
{
  autoRestoreSourceLocalObjectChoice,
  autoRestoreSourceRemoteSystemChoice // независимо от удаленной системы
};

union AutoRestoreSourceType switch (AutoRestoreSourceChoice)
{
  case autoRestoreSourceLocalObjectChoice:
     MONameType localObject;
  case autoRestoreSourceRemoteSystemChoice:
     Istring offLine;
};

/**
Тип атрибута "источник будущего автовосстановления"
*/

typedef AutoRestoreSourceType FutureAutoRestoreSourceType;
```

```
/**
Тип атрибута "идентификатор создателя"
*/

typedef Istring IdentityOfCreatorType;

/**
Тип атрибута "идентификатор последнего модификатора"
*/

typedef Istring IdentityOfLastModifierType;

/**
InformationSize ::=     CHOICE {
  numberOfBits[0] INTEGER,
  numberOfBytes    [1] INTEGER}
*/

enum InformationSizeChoice
{
  numberOfBitsChoice,
  numberOfBytesChoice
};

union InformationSizeType switch (InformationSizeChoice)
{
  case numberOfBitsChoice:
     long bits;
  case numberOfBytesChoice:
     long bytes;
};

/**
LastBackupDestination ::= CHOICE {
  notBackedUp NULL,
  localObject ObjectInstance,
  managingSystem AE-title,
  remoteSystem GraphicString}

Встроенный вариант не определяется в  Рекомендации МСЭ-Т X.744.1
*/

enum LastBackupDestinationChoice
{
  lastBackupDestinationLocalObjectChoice,
  lastBackupDestinationOffLineChoice,
  lastBackupDestinationNotBackedUpChoice
};

/**
Тип атрибута "адресат последнего резервирования"
*/

union LastBackupDestinationType switch (LastBackupDestinationChoice)
{
  case lastBackupDestinationLocalObjectChoice:
     MONameType localObject;
  case lastBackupDestinationOffLineChoice:
     Istring offLine;
  case lastBackupDestinationNotBackedUpChoice:
     NullType noInformation;
};
```

```
/**
LastRestoreSource ::= CHOICE {
  notRestored NULL,
  localObject ObjectInstance,
  managingSystem AE-title,
  remoteSystem GraphicString}

Встроенный вариант не определяется в  Рекомендации МСЭ-Т X.744.1
*/

enum LastRestoreSourceChoice
{
  lastRestoreSourceLocalObjectChoice,
  lastRestoreSourceOffLineChoice,
  lastRestoreSourceNotRestoredChoice
};

/**
Тип атрибута "отправитель последнего восстановления"
*/

union LastRestoreSourceType switch (LastRestoreSourceChoice)
{
  case lastRestoreSourceLocalObjectChoice:
     MONameType localObject;
  case lastRestoreSourceOffLineChoice:
     Istring offLine; // off-line from remote system
  case lastRestoreSourceNotRestoredChoice:
     NullType noInformation;
};

/**
Тип атрибута "поле примечаний"
*/

typedef Istring NoteFieldType;

/**
Тип атрибута "дата доставки"
*/

typedef DateType DateDeliveredType;

/**
Тип атрибута "дата инсталляции"
*/

typedef DateType DateInstalledType;

/**
Тип атрибута "дата последней модификации"
*/

typedef DateType DateOfLastModificationType;

/**
  Тип атрибута "размер файла"
*/

typedef InformationSizeType FileSizeType;
```

```
/**
Тип атрибута "адресат будущего авторезервирования"
*/

typedef BackupDestinationType FutureAutoBackupDestinationType;

/**
Тип атрибута "время последнего резервирования"
*/

typedef DateType LastBackupTimeType;

/**
Тип атрибута "время последнего восстановления"
*/

typedef DateType LastRestoreTimeType;

/**
BackupResult ::= CHOICE {
  inLine [0]  CHOICE {
     successBIT STRING,
     fail-pduSizeLimitation  [3]  NULL,
     fail-securityLicensing  [4]  NULL,
     fail-unknown  [5]  NULL},
  local [1] SEQUENCE {
     destination ObjectInstance, -- в управляемой системе --
     success BOOLEAN -- ИСТИННО при успешном выполнении --
     },
  offLine [2] SEQUENCE {
     destination GraphicString, -- удаленная система
     result CHOICE {
      success  [6]  NULL,
      fail-securityLicensing [7]  NULL,
      fail-unknown  [8]  NULL}
}}

Встроенный вариант не определяется в  Рекомендации МСЭ-Т X.744.1
*/

enum BackupResultChoice
{
  backupResultFailureChoice, // обнаружена ошибка некотого типа при
                             // резервировании объекта
  backupResultLocalChoice,
  backupResultOffLineChoice
};

union BackupResultType switch (BackupResultChoice)
{
  case backupResultFailureChoice:
     ApplicationErrorInfoType error; // из Рекомендации МСЭ-Т X.780
  case backupResultLocalChoice:
     MONameType localObject; // в управляемой системе
  case backupResultOffLineChoice:
     Istring offLine; // в управляемой системе
};
```

```
  /**
  DeliverId ::= CHOICE {
  globalValue  OBJECT IDENTIFIER,
  localValue INTEGER}
*/

typedef long DeliverIdType;
/**
Тип DeliverIdTypeOpt является факультативным. Если дискриминатор имеет
значение "истинно", имеет место указанное значение. В противном случае
значением является NullType.
*/

union DeliverIdTypeOpt switch (boolean)
{
  case TRUE:
     DeliverIdType value;
  case FALSE:
     NullType noInformation;
};

/**
DeliverResult ::= INTEGER {
  pass (0),
  communicationError (1),
  equipmentError (2),
  qosError (3),
  accessDenied (4),
  notFound (5),
  insufficientSpace (6),
  alreadyDelivered (7),
  inProgress (8),
  unknown (9) }

Тип Deliver Result преобразуется в UIDType, который в исходном виде основан на
модуле DeliverResultConst. Это дает возможность приложениям добавлять свои
собственные результаты доставки.
*/

typedef UIDType DeliverResultType;

/**
Destination ::= CHOICE {
  single  AE-title,
  multiple  SET OF AE-title}
-- Следует заметить, что синтаксис подлежащего использованию наименования AE-title
-- взят из Рекомендации МСЭ-Т X.227 | ИСО/МЭК 8650-1 Поправка 1 и не является
"ЛЮБОЙ".
*/

typedef sequence <MONameType> MONameSetType;

enum DestinationChoice
{
  singleChoice,
  multipleChoice
};

union DestinationType switch (DestinationChoice)
{
  case singleChoice:
     MONameType single;
  case multipleChoice:
     MONameSetType multipleValues;
```

```
};
/**
ExecuteProgramReply ::= SEQUENCE {
  processId INTEGER,
  processOwner Identity,
  startTime GeneralizedTime,
  additionalInfo    SET OF ManagementExtension OPTIONAL }
*/

typedef Istring IdentityType;

struct ExecuteProgramReplyType
{
  long processId;
  IdentityType processOwner;
  GeneralizedTimeType startTime;
  AdditionalInformationSetType additionalInfo;
};

typedef AppliedPatchesSeqType InstallReplyType;

/**
Кто выдал запрос?
*/

enum RequestType
{
  automaticRequest,
  managementRequest,     //т. е. метод исполнения
  managedSystemRequest
};

/**
Каковы результаты операции "восстановление"?
*/

enum RestoreResultChoice
{
  restoreResultFailureChoice,    // при восстановлении объекта обнаружены ошибки
                                 // некоторого типа
  restoreResultLocalChoice,
  restoreResultOffLineChoice
};

union RestoreResultType switch (RestoreResultChoice)
{
  case restoreResultFailureChoice:
     ApplicationErrorInfoType error; // из Рекомендации МСЭ-Т X.780
  case restoreResultLocalChoice:
     MONameType localObject; // в управляемой системе
  case restoreResultOffLineChoice:
     Istring offLine; // в управляемой системе
};
```

```
/**
RestoreSource ::= CHOICE {
  localObject ObjectInstance,
  inLine BIT STRING,
  offLine GraphicString
     -- удаленная система через некоторый протокол передачи, например FTAM --
}

Встроенный вариант не определяется в  Рекомендации МСЭ-Т X.744.1
*/

enum RestoreSourceChoice
{
  restoreSourceLocalObjectChoice,
  restoreSourceOffLineChoice
     / / удаленная система через некоторый протокол передачи, например FTAM
};

union RestoreSourceType switch (RestoreSourceChoice)
{
  case restoreSourceLocalObjectChoice:
     MONameType localObject;
  case restoreSourceOffLineChoice:
     Istring offLine;
};

/**
RevertInfo ::= SEQUENCE OF CHOICE {
  patchId GraphicString, -- специфичный для системы идентификатор --
  patchPointer ObjectInstance } --  класс объектов "исполняемая программа" - -
*/

enum RevertChoice
{
  revertPatchIdChoice, // специфичный для системы идентификатор
  revertPatchPointerChoice // класс (или подкласс) объектов "программный модуль"
};

union RevertType switch (RevertChoice)
{
  case revertPatchIdChoice:
     Istring patchId;
  case revertPatchPointerChoice:
     MONameType patchPointer;
};

typedef sequence <RevertType> RevertInfoSetType;

/**
RevertReply ::= SEQUENCE {
  revertedPatches   [0] AppliedPatches,
  additionalInfo    [1] SET OF ManagementExtension OPTIONAL }
*/

typedef AppliedPatchesSeqType RevertReplyType;

/**
DistributedSoftware ::= CHOICE {
  distibutedSoftwareId  GraphicString,
  distributedSoftwarePointer ObjectInstance }
*/
```

```
enum DistributedSoftwareChoice
{
  distributedSoftwareIdChoice,
  distributedSoftwarePointerChoice
};

union DistributedSoftwareType switch (DistributedSoftwareChoice)
{
  case distributedSoftwareIdChoice:
     Istring patchId; // специфичный для системы идентификатор
  case distributedSoftwarePointerChoice:
     MONameType patchPointer; // класс (или подкласс) объектов "программный модуль"
};

typedef sequence <DistributedSoftwareType> TargetSoftwareSetType;

/**
Следует заметить. что тип AdditionalInformationSetType является набором типов
ManagementExtensionType
*/

typedef ManagementExtensionType TransferInfoType;

/**
ValidateInfo ::= CHOICE {
  instanceDefaultValidationType  [0] NULL, -- локальный вопрос --
  registeredValidationType    [1] OBJECT IDENTIFIER }
*/

enum ValidateInfoChoice
{
  registeredValidationTypeChoice,
  instanceDefaultValidationTypeChoice // локальный вопрос
};

union ValidateInfoType switch (ValidateInfoChoice)
{
  case registeredValidationTypeChoice:
     MONameType instanceDefaultValidationType;
  case instanceDefaultValidationTypeChoice:
     NullType noInformation;
};

/**
ValidateReply ::= CHOICE {
  validationTerminated   [0] NULL,
  passValidation    [1] NULL,
  passValidationWithResult    [2] SET OF ManagementExtension,
  failValidation    [3] NULL,
  failValidationWithResult    [4] SET OF ManagementExtension }

Операция "проверка правильности завершения" не определяется в Рекомендации
МСЭ-Т X.744.1
*/

enum ValidateResultChoice
{
  passValidationWithResultChoice,
  failValidationWithResultChoice,
  passValidationChoice,
  failValidationChoice
};
```

```
union ValidateResultType switch (ValidateResultChoice)
{
  case passValidationWithResultChoice:
     AdditionalInformationSetType passValidationWithResult;
  case failValidationWithResultChoice:
     AdditionalInformationSetType failValidationWithResult;
  case failValidationChoice:
     ApplicationErrorInfoType error; // из Рекомендации МСЭ-Т X.780
  case passValidationChoice:
     NullType noInformation;
};

/**
BackupReply ::= SEQUENCE {
  reply  [0] CHOICE {
     success NULL, -- для локального резервирования в автономном режиме
     inLine  BIT STRING },
  additionalInfo    [1] SET OF ManagementExtension OPTIONAL }

  В Рекомендации МСЭ-Т X.744.1 операция "резервирование" выдает уведомление о
результатах   вместо   выдачи самих результатов
*/

/**
TerminateValidationInfo ::= ENUMERATED {
  cancel (0), -- аннулирует результат частичной проверки --
  truncate (1) } - - сообщает результат частично выполненной проверки - -

TerminateValidationReply ::= CHOICE {
  noOutStandingValidation     [0] NULL,
  validationCancelled    [1] NULL,
  resultOfPartialValidation   [2] ValidateReply}

  Операция "проверка правильности завершения" не определяется в Рекомендации
  МСЭ-Т X.744.1
*/

/**
Тип BackupDestinationTypeOpt является факультативным. Если дискриминатор имеет
значение "истинно", имеет место указанное значение. В противном случае
значением является NullType.
*/

union BackupDestinationTypeOpt switch (boolean)
{
  case TRUE:
     BackupDestinationType value;
  case FALSE:
     NullType noInformation;
};

/**
Тип RestoreSourceTypeOpt является факультативным. Если дискриминатор имеет
значение "истинно", имеет место указанное значение. В противном случае
значением является NullType.
*/

union RestoreSourceTypeOpt switch (boolean)
{
  case TRUE:
     RestoreSourceType value;
  case FALSE:
     NullType noInformation;
};
```

```
/**
Тип TargetSoftwareSetTypeOpt является факультативным. Если дискриминатор
находится в состоянии "истинно", имеет место указанное значение. В противном
случае значением является NullType.
*/

union TargetSoftwareSetTypeOpt switch (boolean)
{
  case TRUE:
     TargetSoftwareSetType value;
  case FALSE:
     NullType noInformation;
};

/**
Тип RevertInfoSetTypeOpt является факультативным. Если дискриминатор находится
в состоянии "истинно", имеет место указанное значение. В противном случае
значением является NullType.
*/

union RevertInfoSetTypeOpt switch (boolean)
{
  case TRUE:
     RevertInfoSetType value;
  case FALSE:
     NullType noInformation;
};

/**
Тип ValidateInfoTypeOpt является факультативным. Если дискриминатор находится
в состоянии "истинно", имеет место указанное значение. В противном случае
значением является NullType.
*/
union ValidateInfoTypeOpt switch (boolean)
{
  case TRUE:
     ValidateInfoType value;
  case FALSE:
     NullType noInformation;
};
struct ValidateSoftwareProcessingErrorType
{
  ValidateInfoTypeOpt validateInfo;
};
/**
Тип DestinationTypeOpt является факультативным. Если дискриминатор находится в
состоянии "истинно", имеет место указанное значение. В противном случае
значением является NullType.
*/
union DestinationTypeOpt switch (boolean)
{
  case TRUE:
     DestinationType value;
  case FALSE:
     NullType noInformation;
};
/**
Тип TransferInfoTypeOpt является факультативным. Если дискриминатор находится
в состоянии "истинно", имеет место указанное значение. В противном случае
значением является NullType.
*/
```

```
union TransferInfoTypeOpt switch (boolean)
{
  case TRUE:
     TransferInfoType value;
  case FALSE:
     NullType noInformation;
};

struct DeliverSoftwareProcessingErrorType
{
  DeliverIdTypeOpt deliverId;
  TargetSoftwareSetTypeOpt targetSoftware;
  DestinationTypeOpt targetSystem;
  TransferInfoTypeOpt transferInfo;
  AdditionalInformationSetType additionalInfo;
};

const string administrativeStatePackage =
  "itut_x744d1::administrativeStatePackage";
const string appliedPatchPackage = "itut_x744d1::appliedPatchPackage";
const string checkSumPackage = "itut_x744d1::checkSumPackage";
const string createDeleteNotificationsPackage =
  "itut_x744d1::createDeleteNotificationsPackage";
const string executeProgramPackage = "itut_x744d1::executeProgramPackage";
const string fileInformationPackage =
  "itut_x744d1::fileInformationPackage";
const string filePackage = "itut_x744d1::filePackage";
const string informationAutoBackupPackage =
  "itut_x744d1::informationAutoBackupPackage";
const string informationAutoRestorePackage =
  "itut_x744d1::informationAutoRestorePackage";
const string informationBackupPackage =
  "itut_x744d1::informationBackupPackage";
const string informationRestorePackage =
  "itut_x744d1::informationRestorePackage";
const string installPackage = "itut_x744d1::installPackage";
const string noteFieldPackage = "itut_x744d1::noteFieldPackage";
const string revertPackage = "itut_x744d1::revertPackage";
const string stateChangeNotificationPackage =
  "itut_x744d1::stateChangeNotificationPackage";
const string usageStatePackage = "itut_x744d1::usageStatePackage";
const string validationPackage = "itut_x744d1::validationPackage";
/**
```

9.4 Исключения

```
*/
exception NOadministrativeStatePackage {};
exception NOappliedPatchPackage {};
exception NOcheckSumPackage {};
exception NOexecuteProgramPackage {};
exception NOfileInformationPackage {};
exception NOfilePackage {};
exception NOinformationAutoBackupPackage {};
exception NOinformationAutoRestorePackage {};
exception NOinformationBackupPackage {};
exception NOinformationRestorePackage {};
exception NOinstallPackage {};
exception NOnoteFieldPackage {};
exception NOrevertPackage {};
exception NOusageStatePackage {};
exception NOvalidationPackage {};
```

```
exception BackupSoftwareProcessingFailure
{
  BackupDestinationTypeOpt backupDestionation;
};

/**
Используется различными методами для информирования о том, что эта операция
оказалась безуспешной по причине задержки или выполнения операций     */

exception ConcurrentOperationRequestFailure {};

exception RestoreSoftwareProcessingFailure
{
  RestoreSourceTypeOpt attributes;
};

exception InstallSoftwareProcessingFailure
{
  TargetSoftwareSetTypeOpt attributes;
};

exception RevertSoftwareProcessingFailure
{
  RevertInfoSetTypeOpt attributes;
};

exception ValidateSoftwareProcessingFailure
{
  ValidateSoftwareProcessingErrorType attributes;
};

exception ExecuteProgramSoftwareProcessingFailure
{
  AdditionalInformationSetType additionalInfo;
};

exception DeliverSoftwareProcessingFailure
{
  DeliverSoftwareProcessingErrorType attributes;
};

/**
Используется различными методами для информирования о том, что данная операция
не может быть выполнена по причине условий, недействительных для этой операции
*/

exception OperationStateMismatch {};

/**
```

9.5 Программный модуль

```
*/
    /**
    Этот тип значения используется для получения всех атрибутов
    */
    valuetype SoftwareUnitValueType : truncatable itut_m3120::SoftwareValueType
    {
      public AvailabilityStatusSetType availabilityStatus;
         // GET
      public ProceduralStatusSetType proceduralStatus;
         // GET
      public AppliedPatchesSeqType appliedPatches;
```

```
   // GET
   // appliedPatchPackage
public CheckSumType checkSum;
   // GET
   // checkSumPackage
public DateOfCreationType dateOfCreation;
   // GET
   // fileInformationPackage
public IdentityOfCreatorType identityOfCreator;
   // GET
   // fileInformationPackage
public DateOfLastModificationType dateOfLastModification;
   // GET
   // fileInformationPackage
public IdentityOfLastModifierType identityOfLastModifier;
   // GET
   // fileInformationPackage
public DateDeliveredType dateDelivered;
   // GET
   // fileInformationPackage
public DateInstalledType dateInstalled;
   // GET
   // fileInformationPackage
public FileLocationSetType fileLocation;
   // GET
   // filePackage
public FileSizeType fileSize;
   // GET
   // filePackage
public FileTypeType fileType;
   // GET
   // filePackage
public FutureAutoBackupTriggerThresholdType
          futureAutoBackupTriggerThreshold;
   // GET-REPLACE
   // informationAutoBackupPackage
public FutureAutoBackupDestinationType futureAutoBackupDestination;
   // GET-REPLACE
   // informationAutoBackupPackage
public FutureAutoRestoreSourceType futureAutoRestoreSource;
   // GET-REPLACE
   // informationAutoRestorePackage
public FutureAutoRestoreAllowedType futureAutoRestoreAllowed;
   // GET-REPLACE
   // informationAutoRestorePackage
public LastBackupTimeType lastBackupTime;
   // GET
   // informationBackupPackage
public LastBackupDestinationType lastBackupDestination;
   // GET
   // informationBackupPackage
public LastRestoreTimeType lastRestoreTime;
   // GET
   // informationRestorePackage
public LastRestoreSourceType lastRestoreSource;
   // GET
   // informationRestorePackage
public NoteFieldType noteField;
   // GET-REPLACE
   // noteFieldPackage
public UsageStateType usageState;
   // GET
   // usageStatePackage
```

```
    }; // valuetype SoftwareUnitValueType

    /**
    Класс объектов "программный модуль" - это класс управляемых объектов, который
    обеспечивает управляемую информацию, относящуюся к программным средствам (будь
    она в форме исполняемого файла типа программы, или неисполняемого файла типа
    данных, или таблицы перекрестного преобразования). Тип файла, местоположение
    файла и размер файла относятся к атрибутам, идентифицированным в этом классе
    объектов. При наличии пакета fileInformationPackage обязательным начальным
    значением атрибута dateOfCreation будет время создания управляемого объекта.
    Если обеспечиваются операции резервирования, должны обеспечиваться также
    операции восстановления. Пакеты "резервирование информации" и
    "авторезервирование информации"  могут иметь место в тех  управляемых объектах
    "программный модуль", которые имеют также пакет "восстановление информации"
    или пакет "автовосстановление информации". Пакеты "восстановление информации"
    и "автовосстановление информации" могут иметь место только в том управляемом
    объекте "программный модуль", который имеет также пакет "резервирование
    информации" или  "авторезервирование информации". Если имеется пакет
    "уведомление об изменении значения атрибута"  (унаследованного из программных
    средств суперкласса), то при изменении значения одного из перечисленных ниже
    атрибутов должно выдаваться уведомление "изменение значения", определенное в
    Рекомендации МСЭ-Т X.780:
    - Воздействуемые объекты
    - Аварийное состояние
    - Введенные заплаты
    - Состояние доступности
    - Список текущих проблем
    - Адресат будущего авторезервирования
    - Пороговый предел пуска будущего авторезервирования
    - Будущее авторезервирование разрешено
    - Источник будущего автовосстановления
    - Процедурное состояние
    - Метка пользователя
    - Версия

    Поскольку некоторые из перечисленных выше атрибутов содержатся в условных
    пакетах, то поведение при выдаче уведомления "изменение значения" характерно
    только для тех случаев, когда соответствующие условные пакеты имеются в
    управляемом объекте.
  */
/**
```

9.6 Интерфейс программного модуля"

```
*/
    /**
    Управляемый объект "программный модуль"
    */

    interface  SoftwareUnit : itut_m3120::Software
    {
      /**
      Здесь требуется, чтобы операционное состояние и административное состояние
      из управляемого объекта "программные средства" увязывались с управляемым
      объектом "программный модуль". Это означает, что атрибут "пакеты"
      программного модуля всегда должен содержать строку
      "itut_m3120::administrativeOperationalStatesPackage"
      */
```

```
/**
Состояние "доступно" описано в Рекомендации МСЭ-Т X.731. Дополнительная
информация о поведении приведена в 6.3.2
*/

AvailabilityStatusSetType availabilityStatusGet ()
raises (itut_x780::ApplicationError);

/**
Процедурное состояние описано в Рекомендации МСЭ-Т X.731. Дополнительная
информация о поведении  приведена в 6.3.2
*/

ProceduralStatusSetType proceduralStatusGet ()
raises (itut_x780::ApplicationError);

/**
Атрибут Applied Patches идентифицирует те заплаты, которые были введены в
программный модуль, представленный экземпляром объекта "программный модуль".
Заплаты обновляют программные средства. Значение этого атрибута только
читается и автоматически обновляется при выполнении заплаты в программных
средствах. Синтаксисом этого атрибута является последовательность
идентификаторов заплат, где идентификатор заплаты представляет собой выбранный
экземпляр объекта (если заплата представлена управляемым объектом "программный
модуль") или графическую строку (если заплата не представлена  управляемым
объектом "программный модуль").

Следует заметить, что факультативный атрибут Applied Patches может не
обязательно обеспечиваться внутри, когда он не предусмотрен в классе
управляемых объектов.  Атрибут Applied Patches должен обеспечиваться, если
предусмотрена услуга "инсталляция" или "возврат"  (даже в том случае, когда
атрибут Applied Patches не входит в  класс управляемых объектов).

ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает латание программных средств
*/

AppliedPatchesSeqType appliedPatchesGet ()
     raises (itut_x780::ApplicationError,
     NOappliedPatchPackage);

/**
Атрибут Check Sum идентифицирует контрольную сумму программной информации,
представленной  экземпляром объекта "программный модуль". ИМЕЕТ МЕСТО, ЕСЛИ
экземпляр обеспечивает проверку правильности  контрольной суммы
*/

CheckSumType checkSumGet ()
     raises (itut_x780::ApplicationError,
     NOcheckSumPackage);

/**
Атрибут "дата создания" указывает время создания управляемого объекта.
Синтаксис этого атрибута относится к типу ОбобщенноеВремя. ИМЕЕТ МЕСТО, ЕСЛИ
экземпляр обеспечивает информацию файла
*/

DateOfCreationType dateOfCreationGet ()
     raises (itut_x780::ApplicationError,
         NOfileInformationPackage);
```

```
/**
Атрибут identityOfCreator идентифицирует логический объект, который создает
управляемый объект. Он может быть пустой строкой, если идентификатор
создателя неизвестен. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает информацию
файла
*/

IdentityOfCreatorType identityOfCreatorGet ()
    raises (itut_x780::ApplicationError,
        NOfileInformationPackage);

/**
Атрибут dateOfLastModification указывает время последней или самой последней
модификации (например, латание, возврат, инсталляция, доставка) информации,
представленной экземпляром объекта "программный модуль". Действительными
значениями этого атрибута являются ОбобщенноеВремя или НУЛЬ, если информация не
была модифицирована. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает информацию файла
*/

DateOfLastModificationType dateOfLastModificationGet ()
   raises (itut_x780::ApplicationError,
       NOfileInformationPackage);

/**
Атрибут identityOfLastModifier идентифицирует последний или самый последний
модификатор информации, представленной  экземпляром объекта "программный модуль".
Это может быть пустая строка, если идентификатор последнего модификатора
неизвестен. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает информацию файла
*/

IdentityOfLastModifierType identityOfLastModifierGet ()
   raises (itut_x780::ApplicationError,
       NOfileInformationPackage);

/**
Атрибут dateDelivered указывает время, в которое информация, представленная
экземпляром объекта "программный модуль", была доставлена управляемой системе.
Действительными значениями  этого атрибута являются ОбобщенноеВремя или НУЛЬ,
если информация не была доставлена. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
информацию файла
*/
DateDeliveredType dateDeliveredGet ()
   raises (itut_x780::ApplicationError,
       NOfileInformationPackage);

/**
Атрибут DateInstalled указывает время, когда информация, представленная
экземпляром объекта "программный модуль", была инсталлирована. Действительными
значениями  этого атрибута являются ОбобщенноеВремя или НУЛЬ, если информация не
была инсталлирована. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает информацию файла
*/

DateInstalledType dateInstalledGet ()
   raises (itut_x780::ApplicationError,
       NOfileInformationPackage);

/**
Атрибут fileLocation определяет полный(ые) адрес(а) (логический или физический)
объекта "программный модуль". Формат адреса зависит от реализации, соответствуя
соглашениям по адресации файлов для конкретной рассматриваемой управляемой
системы. Пустой набор этого атрибута указывает, что информация, к которой
относится управляемый объект "программный модуль", не была инсталлирована в
управляемой системе. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает представление файла
*/
```

```
FileLocationSetType fileLocationGet ()
   raises (itut_x780::ApplicationError,
       NOfilePackage);

/**
Атрибут fileSize указывает размер управляемого объекта "программный модуль".
ИМЕЕТ МЕСТО, ЕСЛИ  экземпляр обеспечивает представление файла
*/

FileSizeType fileSizeGet ()
   raises (itut_x780::ApplicationError,
       NOfilePackage);

/**
Атрибут fileType указывает тип программного модуля. Возможными типами
программного модуля служат неструктурированный двоичный файл (например,
исполняемый файл), неструктурированный текстовый файл (например,
неисполняемый файл), блоковый специальный файл и др. ИМЕЕТ  МЕСТО, ЕСЛИ
экземпляр обеспечивает представление файла
*/

FileTypeType fileTypeGet ()
   raises (itut_x780::ApplicationError,
       NOfilePackage);

/**
Атрибут futureAutoBackupTriggerThreshold определяет пороговое значение, при
котором будет запускаться автоматическое резервирование информации,
представленной экземпляром объекта. Это пороговое значение определяется как
количество выполненных модификаций информации. Как только информация будет
смодифицирована установленное количество раз, может осуществляться
автоматическое резервирование. Адресат резервирования определяется в
атрибуте futureAutomacticBackupDestination. Такие резервирования выполняются
дополнительно к другим плановым периодическим резервированиям. При
завершении  автоматического резервирования объект должен выдать уведомление
"отчет о резервировании". ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
автоматическое резервирование
*/

FutureAutoBackupTriggerThresholdType
    futureAutoBackupTriggerThresholdGet ()
   raises (itut_x780::ApplicationError,
       NOInformationAutoBackupPackage);

void futureAutoBackupTriggerThresholdSet
   (in FutureAutoBackupTriggerThresholdType
       futureAutoBackupTriggerThreshold)
   raises (itut_x780::ApplicationError,
       NOInformationAutoBackupPackage);

/**
Атрибут futureAutoBackupDestination определяет адресата, для
которого будет резервироваться информация, представленная в этом
экземпляре объекта. Критерий резервирования определен в атрибуте
futureAutoBackupTriggerThreshold экземпляра объекта. Адресатом может быть
другой экземпляр объекта из того же класса объектов, представленного в той
же локальной управляемой системе либо в удаленной открытой системе (при
использовании конкретного протокола передачи файлов, например FTAM). ИМЕЕТ
МЕСТО, ЕСЛИ экземпляр обеспечивает автоматическое резервирование
*/

FutureAutoBackupDestinationType futureAutoBackupDestinationGet ()
   raises (itut_x780::ApplicationError,
       NOInformationAutoBackupPackage);
```

```
void futureAutoBackupDestinationSet
   (in FutureAutoBackupDestinationType futureAutoBackupDestination)
   raises (itut_x780::ApplicationError,
       NOInformationAutoBackupPackage);

   /**
Атрибут futureAutoRestoreSource определяет источник информации, подлежащей
восстановлению в информацию, представленную экземпляром управляемого
объекта. Источником может быть либо локальный управляемый объект, либо
удаленная система. Критерий запуска автоматического восстановления
информации специфичен для системы. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
автоматическое восстановление
*/

FutureAutoRestoreSourceType futureAutoRestoreSourceGet ()
   raises (itut_x780::ApplicationError,
       NOInformationAutoRestorePackage);

void futureAutoRestoreSourceSet
   (in FutureAutoRestoreSourceType futureAutoRestoreSource)
   raises (itut_x780::ApplicationError,
       NOInformationAutoRestorePackage);

/**
Атрибут futureAutoRestoreAllowed определяет, разрешено ли автоматическое
восстановление информации, представленной этим экземпляром управляемого
объекта. Синтаксис этого атрибута булевского типа, где значение ИСТИННО
означает "разрешено", а ЛОЖНО - "не разрешено". Критерий запуска
автоматического восстановления информации специфичен для системы. ИМЕЕТ
МЕСТО, ЕСЛИ экземпляр обеспечивает автоматическое восстановление
*/

FutureAutoRestoreAllowedType futureAutoRestoreAllowedGet ()
   raises (itut_x780::ApplicationError,
       NOInformationAutoRestorePackage);

void futureAutoRestoreAllowedSet
   (in FutureAutoRestoreAllowedType futureAutoRestoreAllowed)
   raises (itut_x780::ApplicationError,
       NOInformationAutoRestorePackage);

/**
Атрибут LastBackupTime указывает время последнего резервирования информации,
представленной  экземпляром управляемого объекта. Действительными значениями
этого атрибута являются ОбобщенноеВремя или НУЛЬ (если не было проведено
резервирование информации). ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
операцию "резервирование"
*/

LastBackupTimeType LastBackupTimeGet ()
   raises (itut_x780::ApplicationError,
       NOInformationBackupPackage);

/**
Атрибут LastBackupDestination идентифицирует адресата, при его наличии, для
которого резервируется информация, представленная управляемым объектом.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию "резервирование"
*/

LastBackupDestinationType LastBackupDestinationGet ()
   raises (itut_x780::ApplicationError,
       NOInformationBackupPackage);

/**
Услуга резервирования используется управляющей системой для запроса
выполнения резервирования информации, представленной заданным экземпляром
```

```
объекта (т. е. управляемым объектом, представляющим резервируемые
программные средства). После успешной проверки правильности аргумента будет
немедленно начата операция "резервирование". После завершения резервирования
заданного объекта выдается уведомление "отчет о резервировании".

@param backupDestination    Этот параметр указывает адресата, для которого
                            будет резервироваться информация. Возможными
                            адресатами могут быть:
                            - Локальный управляемый объект – в этом случае
                            операция "резервирование" будет выполняться
                            внутри управляемой системы. Информация будет
                            резервироваться для обеспечиваемого экземпляра
                            управляемого объекта.
                            - Выбор в автономном режиме – в этом случае
                            информация резервирования будет передаваться
                            удаленной системе в автономном режиме некоторыми
                            локальными средствами.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию "резервирование"
*/

void backup
   (in BackupDestinationType backupDestionation)
   raises (itut_x780::ApplicationError,
       NOInformationBackupPackage,
       BackupSoftwareProcessingFailure,
       ConcurrentOperationRequestFailure);

/**
Атрибут LastRestoreTime указывает время последнего восстановления информации,
представленной  экземпляром управляемого объекта. Действительными значениями
этого атрибута являются ОбобщенноеВремя или НУЛЬ (если не было выполнено
восстановление информации). ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
операцию "восстановление" или автоматическое восстановление
*/

LastRestoreTimeType LastRestoreTimeGet ()
   raises (itut_x780::ApplicationError,
       NOInformationRestorePackage,
       NOInformationAutoRestorePackage);

/**
Атрибут LastRestoreSource идентифицирует источник, при его наличии,
информация из которого, представленная управляемым объектом,
восстановливается. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию
"восстановление" или автоматическое восстановление
*/

LastRestoreSourceType LastRestoreSourceGet ()
   raises (itut_x780::ApplicationError,
       NOInformationRestorePackage,
       NOInformationAutoRestorePackage);

/**
Услуга восстановления используется управляющей системой для запроса
выполнения восстановления информации, представленной заданным экземпляром
объекта. После успешной проверки правильности аргумента будет немедленно
начата операция "восстановление". После завершения резервирования заданного
объекта выдается уведомление "отчет о восстановлении".

@param restoreSource        Этот параметр указывает источник, для которого
                            должна быть восстановлена информация. Возможными
                            источниками могут быть:
                            - Локальный управляемый объект того же класса,
                            что и класс, к которому применима данная
                            операция. В этом случае операция
                            "восстановление" будет выполняться внутри
                            управляемой системы.
```

```
                                - Выбор в автономном режиме – в этом случае
                                информация восстановления будет передаваться
                                удаленной системе в автономном режиме путем
                                использования локально выбранного протокола
                                передачи файлов.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию восстановления
*/

void restore
   (in RestoreSourceType restoreSource)
   raises (itut_x780::ApplicationError,
       NOInformationRestorePackage,
       RestoreSoftwareProcessingFailure,
       ConcurrentOperationRequestFailure);

/**
Услуга инсталляции используется управляющей системой для выдачи управляемой
системе инструкции по инсталляции доставленного  экземпляра объекта
"программный модуль". В применимых случаях услуга инсталляции может обновить
значение атрибута "введенные заплаты".

@param targetSoftware       Этот параметр указывает источник программных
                            средств, подлежащих инсталляции. Этот источник
                            должен быть уникальным с точки зрения атрибута
                            "введенные заплаты". Источником может быть одно
                            или несколько из перечисленного ниже:
                            - Id  заплаты – специфичный для системы
                            идентификатор.
                            - Указатель заплаты -  класс управляемых
                            объектов "программный модуль" (или "исполняемая
                            программа").
@return                     Услуга инсталляции будет автоматически выдавать
                            значение атрибута "введенные заплаты" того
                            экземпляра объекта "программный модуль", на
                            который ориентирована услуга.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию "инсталляция"
*/

InstallReplyType install
   (in TargetSoftwareSetType targetSoftware)
   raises (itut_x780::ApplicationError,
       NOinstallPackage,
       InstallSoftwareProcessingFailure,
       OperationStateMismatch,
       ConcurrentOperationRequestFailure/**
/**
Поле noteField содержит всю информацию и комментарии, относящиеся к
управляемому объекту, включая все специфичные для инсталляции инструкции,
стартовые параметры и значения, информацию, необходимую для активизации
возможностей управляемого объекта, и т. п. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр
обеспечивает его
*/

NoteFieldType noteFieldGet ()
   raises (itut_x780::ApplicationError,
       NOnoteFieldPackage);

void noteFieldSet
   (in NoteFieldType noteField)
   raises (itut_x780::ApplicationError,
       NOnoteFieldPackage);
```

```
/**
Услуга "возврат" используется  управляющей системой (например, ОС) для
выдачи управляемой системе инструкции отменить введенную заплату или набор
заплат программных средств,  представленных управляемым объектом
"программный модуль". Если услуга "возврат" успешно отменяет все заплаты,
которые были до этого инсталлированы (т. е. атрибут "введенные заплаты"
пустой), то внутреннее состояние программного модуля изменится с
"инсталлировано" на "доставлено".
@param revertInfo          Параметр "информация возврата" должен указывать
                           одну или несколько ранее использованных заплат,
                           инсталляция которых была осуществлена.
                           Идентификатор каждой введенной заплаты
                           представляет собой выбор специфичного для
                           системы идентификатора или экземпляр объекта
                           "программный модуль", в зависимости от значений,
                           первоначально установленных в предыдущей
                           операции "инсталляция" в данном экземпляре
                           объекта "программный модуль".
@return                    Услуга "возврат" будет автоматически выдавать
                           значение атрибута "введенные заплаты" того
                           экземпляра объекта "программный модуль", на
                           который ориентирована эта услуга.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает ее
*/
RevertReplyType revert
   (in RevertInfoSetType revertInfo)
   raises (itut_x780::ApplicationError,
       NOrevertPackage,
       OperationStateMismatch,
       RevertSoftwareProcessingFailure,
       ConcurrentOperationRequestFailure);
/**
Состояние "использование" описано в Рекомендации МСЭ-Т X.731. ИМЕЕТ МЕСТО,
ЕСЛИ экземпляр обеспечивает его
*/
UsageStateType usageStateGet ()
   raises (itut_x780::ApplicationError,
       NOusageStatePackage);
/**
Услуга "проверка правильности" используется управляющей системой для запроса
выполнения проверки правильности информации, представленной  экземпляром
объекта  "программный модуль".
@param validateInfo        Этот параметр указывает, какой вид проверки
                           правильности должен быть выполнен. Возможными
                           видами могут быть:
                           - Регистрируемая проверка правильности – в этом
                           случае информация проверки правильности
                           определяется через другой заданный управляемый
                           объект.
                           - Проверка правильности по умолчанию – в этом
                           случае для этого конкретного экземпляра объекта
                           должна использоваться проверка правильности по
                           умолчанию.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает его
*/
void validate
   (in ValidateInfoType validateInfo)
   raises (itut_x780::ApplicationError,
       NOValidationPackage,
       OperationStateMismatch,
       ValidateSoftwareProcessingFailure,
       ConcurrentOperationRequestFailure);
```

```
        /**
        Параметр Alarm Effect On Service по Рекомендации МСЭ-Т X.744 включен в
        состав параметров processingErrorAlarm.

        Теперь обязательна выдача уведомления Processing Error в классе управляемых
        объектов "программные средства".
        */

        MANDATORY_NOTIFICATION(
           itut_x780::Notifications, processingErrorAlarm)

        /**
        Уведомление "отчет о резервировании" выдается для информирования о
        результатах резервирования информации, представленной данным объектом. Оно
        инициируется операцией "резервирование".  Адресат резервирования может быть
        локальным (т. е. резервирование для другого  объекта "программный модуль" в
        локальной управляемой системе) или в автономном режиме для удаленной системы
        путем использования конкретного протокола передачи файлов (например, FTAM).
        Результат резервирования, т. е. успешность или безуспешность, будет сообщен
        в этом уведомлении. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию
        "резервирование" или автоматическое резервирование
        */

        CONDITIONAL_NOTIFICATION(
           itut_x744d1::Notifications, backupReport, InformationBackupPackage)

        CONDITIONAL_NOTIFICATION(
           itut_x744d1::Notifications, backupReport,
           InformationAutoBackupPackage)

        /**
        Уведомление "отчет о восстановлении" выдается для информирования о
        восстановлении в исходное состояние управляемого объекта после предыдущего
        резервирования. Восстановление может быть начато автоматически (в
        соответствии с атрибутами Future Auto Restore Source Type и Future Auto
        Restore Allowed Type и со специфичными  для системы критериями) через запрос
        управления (посредством операции "восстановление") либо по инициативе
        управляемой системы. Источник восстановления может быть локальным (т. е.
        восстановление от другого  объекта "программный модуль" внутри локально
        управляемой системы) либо независимым от удаленной системы путем
        использования конкретного протокола передачи файлов (например, FTAM).
        Результат восстановления может быть передан  в этом уведомлении. ИМЕЕТ
        МЕСТО, ЕСЛИ экземпляр обеспечивает операцию "восстановление"  или
        автоматическое восстановление
        */

        CONDITIONAL_NOTIFICATION(
           itut_x744d1::Notifications, RestoreReport,
           InformationRestorePackage)

        CONDITIONAL_NOTIFICATION(
           itut_x744d1::Notifications, RestoreReport,
           InformationAutoRestorePackage)

        /**
        Уведомление "отчет о проверке правильности" выдается для информирования о
        проверке правильности управляемого объекта. Результат операции "проверка
        правильности" может быть сообщен в этом уведомлении. ИМЕЕТ МЕСТО, ЕСЛИ
        экземпляр обеспечивает операцию "проверка правильности"
        */

        CONDITIONAL_NOTIFICATION(
           itut_x744d1::Notifications, validateReport, validationPackage)

        }; // interface  SoftwareUnit
/**
```

9.7 Интерфейс программного модуля_F

```
*/

     /**
     Управляемый объект "фасад программного модуля" -  см. Рекомендацию МСЭ-Т
     X.780.1
     */

     interface  SoftwareUnit _F : itut_m3120::Software_F
     {
       /**
       Операционное состояние и административное состояние из управляемого объекта
       "программные средства" теперь должно относиться к управляемому объекту
       "программный модуль". Это означает, что  атрибут "пакеты" для   программного
       модуля должен всегда содержать строку
       "itut_m3120::administrativeOperationalStatesPackage"
       */

       /**
       Состояние "доступно" описано в Рекомендации МСЭ-Т X.731. Дополнительная
       информация о поведении приведена в 6.3.2

       */

       AvailabilityStatusSetType availabilityStatusGet
          (in MONameType name)
          raises (itut_x780::ApplicationError);

       /**
       Процедурное состояние описано в Рекомендации МСЭ-Т X.731. Дополнительная
       информация о поведении  приведена в 6.3.2
       */

       ProceduralStatusSetType proceduralStatusGet
          (in MONameType name)
          raises (itut_x780::ApplicationError);

       /**
       Атрибут "введенные заплаты" идентифицирует заплаты, которые  были введены
       для "программного модуля"  представленным экземпляром объекта "программный
       модуль" и продолжают существовать в нем.  Заплаты обновляют программные
       средства. Значение этого  атрибута только читается и автоматически
       обновляется, когда заплата выполняется в программных средствах. Синтаксис
       этого атрибута представляет собой последовательность  идентификаторов
       заплат, где идентификатор заплаты представляет собой выбор экземпляра
       объекта  (если заплата представлена управляемым объектом "программный
       модуль") или  графическую строку (если заплата не представлена  управляемым
       объектом "программный модуль").

       Следует заметить, что может оказаться необходимым обеспечивать
       факультативный атрибут "введенные заплаты" внутри, если он не предусмотрен в
       классе управляемых объектов. Атрибут "введенные заплаты" должен
       обеспечиваться, если обеспечивается услуга "инсталляция" или "возврат"
       (даже в том случае, когда атрибут "введенные заплаты" не находится в классе
       управляемых объектов).

       ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает латание программных средств
       */

       AppliedPatchesSeqType appliedPatchesGet
          (in MONameType name)
          raises (itut_x780::ApplicationError,
              NOappliedPatchPackage);
```

```
/**
Атрибут Check Sum идентифицирует контрольную сумму информации программных
средств,  представленной экземпляром объекта "программный модуль". ИМЕЕТ
МЕСТО, ЕСЛИ экземпляр обеспечивает проверку правильности контрольной суммы
*/

CheckSumType checkSumGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
   NOcheckSumPackage);

/**
Атрибут dateOfCreation указывает время создания управляемого объекта.
Синтаксисом этого атрибута является тип ОбобщенноеВремя. ИМЕЕТ МЕСТО, ЕСЛИ
экземпляр обеспечивает информацию файла
*/

DateOfCreationType dateOfCreationGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOfileInformationPackage);

/**
Атрибут identityOfCreator идентифицирует логический объект, который создает
управляемый объект. Это может быть пустая строка, если идентификатор
создателя неизвестен. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает информацию
файла
*/

IdentityOfCreatorType identityOfCreatorGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOfileInformationPackage);

/**
Атрибут dateOfLastModification указывает время последней или самой последней
модификации (например, латание, возврат, инсталляция, доставка) информации,
представленной экземпляром объекта "программный модуль". Действительными
значениями этого атрибута являются ОбобщенноеВремя или НУЛЬ, если информация
не была модифицирована. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает информацию
файла
*/

DateOfLastModificationType dateOfLastModificationGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
    NOfileInformationPackage);

/**
Атрибут identityOfLastModifier идентифицирует последний или самый последний
модификатор информации, представленной экземпляром объекта "программный
модуль". Это может быть пустая строка, если идентификатор создателя
неизвестен. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает информацию файла
*/

IdentityOfLastModifierType identityOfLastModifierGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOfileInformationPackage);
```

```
/**
Атрибут dateDelivered указывает время, в которое информация, представленная
экземпляром объекта "программный модуль", была доставлена управляемой
системе. Действительными значениями этого атрибута являются ОбобщенноеВремя
или НУЛЬ, если информация не была модифицирована. ИМЕЕТ МЕСТО, ЕСЛИ
экземпляр обеспечивает информацию файла
*/

DateDeliveredType dateDeliveredGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOfileInformationPackage);

/**
Атрибут dateInstalled указывает время, в которе информация, представленная
экземпляром объекта "программный модуль", была инсталлирована.
Действительными значениями этого атрибута являются ОбобщенноеВремя или НУЛЬ,
если информация не была модифицирована. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр
обеспечивает информацию файла
*/

DateInstalledType dateInstalledGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOfileInformationPackage);

/**
Атрибут fileLocation определяет полный(ые) адрес(а) (логический или
физический) объекта "программный модуль". Формат адреса зависит от
реализации, соответствуя соглашениям по адресации файлов для конкретной
рассматриваемой управляемой системы. Пустой набор этого атрибута указывает,
что информация, к которой относится управляемый объект "программный модуль",
не была инсталлирована в управляемой системе. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр
обеспечивает представление файла
*/

FileLocationSetType fileLocationGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOfilePackage);

/**
Атрибут fileSize указывает размер управляемого объекта "программный модуль".
ИМЕЕТ МЕСТО, ЕСЛИ  экземпляр обеспечивает представление файла
*/

FileSizeType fileSizeGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOfilePackage);

/**
Атрибут fileType указывает тип программного модуля. Возможными типами
программного модуля служат неструктурированный двоичный файл (например,
исполняемый файл), неструктурированный текстовый файл (например,
неисполняемый файл), блоковый специальный файл и др. ИМЕЕТ  МЕСТО, ЕСЛИ
экземпляр обеспечивает представление файла
*/

FileTypeType fileTypeGet
   (in MONameType name,
   raises (itut_x780::ApplicationError,
       NOfilePackage);
```

```
/**
Атрибут futureAutoBackupTriggerThreshold определяет пороговое значение, при
котором будет запускаться автоматическое резервирование информации,
представленной экземпляром объекта. Это пороговое значение определяется как
количество выполненных модификаций информации. Как только информация будет
смодифицирована установленное количество раз, может осуществляться
автоматическое резервирование. Адресат резервирования определяется в
атрибуте futureAutomacticBackupDestination. Такие резервирования выполняются
дополнительно к другим плановым периодическим резервированиям. При
завершении  автоматического резервирования объект должен выдать уведомление
"отчет о резервировании". ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
автоматическое резервирование
*/

FutureAutoBackupTriggerThresholdType
    futureAutoBackupTriggerThresholdGet ()
   raises (itut_x780::ApplicationError,
       NOInformationAutoBackupPackage);

void futureAutoBackupTriggerThresholdSet
   (in FutureAutoBackupTriggerThresholdType
       futureAutoBackupTriggerThreshold)
   raises (itut_x780::ApplicationError,
       NOInformationAutoBackupPackage);

/**
Атрибут futureAutoBackupDestination определяет адресата, для которого будет
резервироваться информация, представленная в этом экземпляре объекта.
Критерий резервирования определен в атрибуте futureAutoBackupTriggerThreshold
экземпляра объекта. Адресатом может быть другой экземпляр объекта из того же
класса объектов, представленного в той же локальной управляемой системе либо
в удаленной открытой системе (при использовании конкретного протокола
передачи файлов, например FTAM). ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
автоматическое резервирование
*/

FutureAutoBackupDestinationType futureAutoBackupDestinationGet ()
   raises (itut_x780::ApplicationError,
       NOInformationAutoBackupPackage);

void futureAutoBackupDestinationSet
   (in FutureAutoBackupDestinationType futureAutoBackupDestination)
   raises (itut_x780::ApplicationError,
       NOInformationAutoBackupPackage);

/**
Атрибут futureAutoRestoreSource определяет источник информации, подлежащей
восстановлению в информацию, представленную экземпляром управляемого
объекта. Источником может быть либо локальный управляемый объект, либо
удаленная система. Критерий запуска автоматического восстановления
информации специфичен для системы. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
автоматическое восстановление
*/

FutureAutoRestoreSourceType futureAutoRestoreSourceGet ()
   raises (itut_x780::ApplicationError,
       NOInformationAutoRestorePackage);
```

```
void futureAutoRestoreSourceSet
   (in FutureAutoRestoreSourceType futureAutoRestoreSource)
   raises (itut_x780::ApplicationError,
    NOInformationAutoRestorePackage

/**
Атрибут futureAutoRestoreAllowed определяет, разрешено ли автоматическое
восстановление информации, представленной этим экземпляром управляемого
объекта. Синтаксис этого атрибута булевского типа, где значение ИСТИННО
означает "разрешено", а ЛОЖНО – "не разрешено". Критерий запуска
автоматического восстановления информации специфичен для системы. ИМЕЕТ
МЕСТО, ЕСЛИ экземпляр обеспечивает автоматическое восстановление
*/

FutureAutoRestoreAllowedType futureAutoRestoreAllowedGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
    NOinformationAutoRestorePackage);

void futureAutoRestoreAllowedSet
   (in MONameType name,
   in FutureAutoRestoreAllowedType futureAutoRestoreAllowed)
   raises (itut_x780::ApplicationError,
    NOinformationAutoRestorePackage);

/**
Атрибут LastBackupTime указывает время последнего резервирования информации,
представленной  экземпляром управляемого объекта. Действительными значениями
этого атрибута являются ОбобщенноеВремя или НУЛЬ (если не было проведено
резервирования информации). ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
операцию "резервирование"
*/

LastBackupTimeType LastBackupTimeGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
    NOinformationBackupPackage);

/**
Атрибут LastBackupDestination идентифицирует адресата, при его наличии, для
которого резервируется информация, представленная управляемым объектом.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию "резервирование"
*/

LastBackupDestinationType LastBackupDestinationGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
    NOinformationBackupPackage);

/**
Услуга резервирования используется управляющей системой для запроса
выполнения резервирования информации, представленной заданным экземпляром
объекта (т. е. управлемым объектом, представляющим резервируемые программные
средства). После успешной проверки правильности аргумента будет немедленно
начата операция "резервирование". После завершения резервирования заданного
объекта выдается уведомление "отчет о резервировании".

@param name                 Имя экземпляра управляемого объекта "программный
                            модуль"
@param backupDestination    Этот параметр указывает адресата, для которого
                            будет резервироваться информация. Возможными
                            адресатами могут быть:
                            - Локальный управляемый объект – в этом случае
                            операция "резервирование" будет выполняться внутри
                            управляемой системы. Информация будет
```

```
                                резервироваться для обеспечиваемого экземпляра
                                управляемого объекта.
                                - Выбор в автономном режиме – в этом случае
                                информация резервирования будет передаваться
                                удаленной системе в автономном режиме некоторыми
                                локальными средствами.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию "резервирование"
*/

void backup
   (in MONameType name,
   in BackupDestinationType backupDestionation)
   raises (itut_x780::ApplicationError,
       NOinformationBackupPackage,
       BackupSoftwareProcessingFailure,
       ConcurrentOperationRequestFailure);

/**
Атрибут LastRestoreTime указывает время последнего восстановления информации,
представленной  экземпляром управляемого объекта. Действительными значениями
этого атрибута являются ОбобщенноеВремя или НУЛЬ (если не было выполнено
восстановление информации). ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает
операцию "восстановление" или автоматическое восстановление
*/

LastRestoreTimeType LastRestoreTimeGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOinformationRestorePackage,
       NOinformationAutoRestorePackage);

/**
Атрибут LastRestoreSource идентифицирует источник, при его наличии,
информация из которого, представленная управляемым объектом,
восстановливается. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию
"восстановление" или автоматическое восстановление
*/

LastRestoreSourceType LastRestoreSourceGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOinformationRestorePackage,
       NOinformationAutoRestorePackage);

/**
Услуга восстановления используется управляющей системой для запроса
выполнения восстановления информации, представленной заданным экземпляром
объекта. После успешной проверки правильности аргумента будет немедленно
начата операция "восстановление". После завершения резервирования заданного
объекта выдается уведомление "отчет о восстановлении".

@param name                 Имя экземпляра управляемого объекта "программный
                            модуль"
@param restoreSource        Этот параметр указывает источник, для которого
                            должна быть  восстановлена информация.
                            Возможными источниками могут быть:
                            - Локальный управляемый объект того же класса, что
                            и класс, к которому применима данная операция. В
                            этом случае операция "восстановление" будет
                            выполняться внутри управляемой системы.
                            - Выбор в автономном режиме – В этом случае
                            информация восстановления будет передаваться
                            удаленной системе в автономном режиме путем
                            использования локально выбранного протокола
                            передачи файлов.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию восстановления
*/
```

```
void restore
   (in MONameType name,
   in RestoreSourceType restoreSource)
   raises (itut_x780::ApplicationError,
       NOinformationRestorePackage,
       RestoreSoftwareProcessingFailure,
       ConcurrentOperationRequestFailure);

/**
Услуга инсталляции используется управляющей системой для выдачи управляемой
системе инструкции по инсталляции доставленного  экземпляра объекта
"программный модуль". В применимых случаях услуга инсталляции может обновить
значение атрибута "введенные заплаты".

@param name                 Имя  экземпляра управляемого объекта
                            "программный модуль"
@param targetSoftware       Этот параметр указывает источник программных
                            средств, подлежащих инсталляции. Этот источник
                            должен быть уникальным с точки зрения атрибута
                            "введенные заплаты". Источником может быть один
                            или несколько из перечисленного ниже:
                            - Id  заплаты – специфичный для системы идентификатор.
                            - Указатель заплаты - класс управляемых объектов
                            "программный модуль" (или "исполняемая
                            программа").
@return                     Услуга инсталляции будет автоматически выдавать
                            значение атрибута "введенные заплаты" того
                            экземпляра объекта "программный модуль", на
                            который ориентирована услуга.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию "инсталляция"
*/

InstallReplyType install
   (in MONameType name,
   in TargetSoftwareSetType targetSoftware)
   raises (itut_x780::ApplicationError,
       NOinstallPackage,
       InstallSoftwareProcessingFailure,
       OperationStateMismatch,
       ConcurrentOperationRequestFailure);

/**
Поле noteField содержит всю информацию и комментарии, относящиеся к объекту,
включая все специфичные для инсталляции инструкции, стартовые параметры и
значения, информацию, необходимую для активизации возможностей управляемого
объекта, и т. п. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает его
*/

NoteFieldType noteFieldGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOnoteFieldPackage);

void noteFieldSet
   (in MONameType name,
   in NoteFieldType noteField)
   raises (itut_x780::ApplicationError,
       NOnoteFieldPackage);

/**
Услуга "возврат" используется  управляющей системой (например, ОС) для
выдачи управляемой системе инструкции отменить введенную заплату или набор
заплат программных средств,  представленных управляемым объектом
"программный модуль". Если услуга возврата успешно отменяет все заплаты,
которые были до этого инсталлированы (т. е. атрибут "введенные заплаты"
пустой), то внутреннее состояние программного модуля изменится с
"инсталлировано" на "доставлено".
```

```
@param name                 Имя экземпляра управляемого объекта "программный
                            модуль"
@param revertInfo           Параметр "информация возврата" должен указывать
                            одну или несколько ранее использованных заплат,
                            для которых была осуществлена инсталляция.
                            Идентификатор каждой использованной заплаты
                            представляет собой выбор специфичного для
                            системы идентификатора или экземпляр объекта
                            "программный модуль", в зависимости от значений,
                            первоначально установленных в предыдущей
                            операции "инсталляция" в данном экземпляре
                            объекта "программный модуль".
@return                     Услуга "возврат" будет автоматически выдавать
                            значение атрибута "введенные заплаты" того
                            экземпляра объекта "программный модуль", на
                            который ориентирована эта услуга.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает ее
*/

RevertReplyType revert
   (in MONameType name,
   in RevertInfoSetType revertInfo)
   raises (itut_x780::ApplicationError,
       NOrevertPackage,
       OperationStateMismatch,
       RevertSoftwareProcessingFailure,
       ConcurrentOperationRequestFailure);

/**
Состояние "использование" описано в Рекомендации МСЭ-Т X.731. ИМЕЕТ МЕСТО,
ЕСЛИ экземпляр обеспечивает его
*/

UsageStateType usageStateGet
   (in MONameType name)
   raises (itut_x780::ApplicationError,
       NOusageStatePackage);

/**
Услуга "проверка правильности" используется управляющей системой для запроса
выполнения проверки правильности информации, представленной  экземпляром
объекта  "программный модуль".

@param name                 Имя экземпляра управляемого объекта "программный
                            модуль"
@param validateInfo         Этот параметр указывает, какой вид проверки
                            правильности должен быть выполнен. Возможными
                            видами могут быть:
                            - Регистрируемая проверка правильности – в этом
                            случае информация проверки правильности
                            определяется через другой заданный управляемый
                            объект.
                            - Проверка правильности по умолчанию – в этом
                            случае для этого конкретного экземпляра объекта
                            должна использоваться проверка правильности по
                            умолчанию.
ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает его
*/

void validate
   (in MONameType name,
   in ValidateInfoType validateInfo)
   raises (itut_x780::ApplicationError,
       NO validationPackage,
       OperationStateMismatch,
       ValidateSoftwareProcessingFailure,
       ConcurrentOperationRequestFailure);
```

```
      /**
      Параметр Alarm Effect On Service по Рекомендации МСЭ-Т X.744 включен в
      состав параметров processingErrorAlarm.

      Теперь обязательна выдача уведомления Processing Error в классе управляемых
      объектов "программные средства".
      */

      MANDATORY_NOTIFICATION(
         itut_x780::Notifications, processingErrorAlarm)

      /**
      Уведомление "отчет о резервировании" выдается для информирования о
      результатах резервирования информации, представленной данным объектом. Оно
      инициируется операцией "резервирование".  Адресат резервирования может быть
      локальным (т. е. резервирование для другого  объекта "программный модуль" в
      локальной управляемой системе) или независимым от удаленной системы путем
      использования конкретного протокола передачи файлов (например, FTAM).
      Результат резервирования, т. е. успешность или безуспешность, будет сообщен
      в этом уведомлении. ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает операцию
      "резервирование" или автоматическое резервирование
      */
      CONDITIONAL_NOTIFICATION(
         itut_x744d1::Notifications, backupReport, informationBackupPackage)
      CONDITIONAL_NOTIFICATION(
         itut_x744d1::Notifications, backupReport,
         informationAutoBackupPackage)
      /**
      Уведомление "отчет о восстановлении" выдается для информирования о
      восстановлении в исходное состояние управляемого объекта после предыдущего
      резервирования. Восстановление может быть начато автоматически (в
      соответствии с атрибутами Future Auto Restore Source Type и Future Auto
      Restore Allowed Type и со специфичными  для системы критериями) через запрос
      управления (посредством операции "восстановление") либо по инициативе
      управляемой системы. Источник восстановления может быть локальным (т. е.
      восстановление от другого  объекта "программный модуль" внутри локально
      управляемой системы) либо независимым от удаленной системы путем
      использования конкретного протокола передачи файлов (например, FTAM).
      Результат восстановления может быть передан  в этом уведомлении. ИМЕЕТ
      МЕСТО, ЕСЛИ экземпляр обеспечивает операцию "восстановление"  или
      автоматическое восстановление
      */
      CONDITIONAL_NOTIFICATION(
         itut_x744d1::Notifications, restoreReport,
         informationRestorePackage)

      CONDITIONAL_NOTIFICATION(
         itut_x744d1::Notifications, restoreReport,
         informationAutoRestorePackage)
      /**
      Уведомление "отчет о проверке правильности" выдается для информирования о
      проверке правильности управляемого объекта. Результат операции "проверка
      правильности" может быть сообщен в этом уведомлении. ИМЕЕТ МЕСТО, ЕСЛИ
      экземпляр обеспечивает операцию "проверка правильности"
      */

      CONDITIONAL_NOTIFICATION(
         itut_x744d1::Notifications, validateReport, проверка правильностиPackage)
   }; // interface  SoftwareUnit _F
/**
```

```
*/
     /**
     Создание и удаление  программного модуля
     */

       interface  SoftwareUnit Factory : itut_x780::ManagedObjectFactory
       {
          itut_x780::ManagedObject create
           (in NameBindingType nameBinding,
           in MONameType superior,
           in string reqID,   // автонаименование в случае пустой строки
           out MONameType name,
           in StringSetType packageNameList,
           in AdministrativeStateType adminstrativeState,
                // conditional
                // itut_m3120::administrativeOperationalStatePackage
                // GET-REPLACE
           in AlarmSeverityAssignmentProfileNameType profile,
                // conditional
                // itut_m3120::alarmSeverityAssignmentPointerPackage
                // GET-REPLACE
           in Istring userLabel,
                // conditional
                // itut_m3120::userLabelPackage
                // GET-REPLACE
           in Istring vendorName,
                // conditional
                // itut_m3120::vendorNamePackage
                // GET-REPLACE
           in Istring version,
                // conditional
                // itut_m3120::versionPackage
                // GET-REPLACE
           in ArcProbableCauseSetType arcProbableCauseList,
                // conditional
                // itut_m3120::arcPackage
                // GET-REPLACE, ADD-REMOVE
           in ArcIntervalProfileNameType arcIntervalProfilePointer,
                // conditional
                // itut_m3120::arcPackage
                // GET-REPLACE
           in ArcTimeType arcManagementRequestedInterval,
                // conditional
                // itut_m3120::arcPackage
                // GET-REPLACE
           in AvailabilityStatusSetType availabilityStatus,
                // GET
                // SET-BY-CREATE
           in FutureAutoBackupTriggerThresholdType
                                futureAutoBackupTriggerThreshold,
                // GET-REPLACE
                // informationAutoBackupPackage
           in FutureAutoBackupDestinationType futureAutoBackupDestination,
                // GET-REPLACE
                // informationAutoBackupPackage
           in FutureAutoRestoreSourceType futureAutoRestoreSource,
                // GET-REPLACE
                // informationAutoRestorePackage
           in FutureAutoRestoreAllowedType futureAutoRestoreAllowed,
                // GET-REPLACE
                // informationAutoRestorePackage
```

```
      in NoteFieldType noteField
         // GET-REPLACE
         // noteFieldPackage
      )
      raises (itut_x780::ApplicationError,
         itut_x780::CreateError);
   }; // interface  SoftwareUnit Factory

/**
```

9.9 Исполняемая программа

```
*/
   /**
   Этот тип значения используется для получения всех атрибутов
   */
   valuetype ExecutableSoftwareValueType : truncatable  SoftwareUnitValueType
   {
   }; // valuetype ExecutableSoftwareValueType
   /**
   Класс объектов executableSoftware представляет собой такой класс управляемых
   объектов, который обеспечивает управляемую информацию, относящуюся к
   исполняемой программе в управляемой системе. Фактически исполняемая программа
   (которая может содержать кодовые сегменты с сегментами данных или без них и
   др.) может быть в нестандартном машинно-читаемом формате, который в общем
   случае является нечитабельным для управляющей системы и для остального
   внешнего мира. Операция, называемая executeProgram (условно), может
   использоваться для выполнения программы, представленной экземпляром объекта
   executableSoftware. Для указания наличия какой-либо активной исполняемой
   программы используется атрибут usageState.
   */
/**
```

9.10 Интерфейс исполняемой программы

```
*/
   /**
   Управляемый объект "исполняемая программа"
   */
   interface ExecutableSoftware :  SoftwareUnit
   {
     /**
     Атрибут UsageState из управляемого объекта "программный модуль" теперь
     должен быть представлен с управляемым объектом "исполняемая программа". Это
     означает, что атрибут Packages для исполняемой программы всегда должен
     содержать строку "itut_x744d1::usageStatePackage"
     */
     /**
     Услуга "исполняемая программа" используется управляющей системой для
     инициализации выполнения программы, представленной в объекте "исполняемая
     программа". Для выполнения операции "исполняемая программа" сама исполняемая
     программа должна находиться во внутреннем состоянии "инсталлировано",
     административное состояние должно иметь значение "разблокировано",
     операционное состояние должно иметь значение "активизировано", а состояние
     использования должно иметь значение либо "активное", либо "холостое".
     @param additionalInfo      Определяет параметры, используемые при
                                выполнении программы.
     @return                    Успешный запрос будет подтвержден информацией,
                                включая параметры "идентификатор
```

```
                                    процесса", "владелец процесса", "начальное
                                    стартовое время" и "дополнительные
                                    обеспечиваемые параметры.
      ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает его
      */

          ExecuteProgramReplyType executeProgram
              (in AdditionalInformationSetType additionalInfo)
              raises (itut_x780::ApplicationError,
                  NOexecuteProgramPackage,
                  OperationStateMismatch,
                  ExecuteProgramSoftwareProcessingFailure);

          }; // interface ExecutableSoftware

/**
```

9.11 Интерфейс исполняемой программы _F

```
*/
     /**
     Управляемый объект "фасад исполняемого модуля" - см. Рекомендацию МСЭ-Т
  X.780.1
     */

     interface ExecutableSoftware_F :  SoftwareUnit _F
     {
       /**
       Атрибут UsageState из управляемого объекта "программный модуль" теперь
       должен быть представлен с управляемым объектом "исполняемая программа". Это
       означает, что атрибут Packages для исполняемой программы всегда должен
       содержать строку "itut_x744d1::usageStatePackage"
       */

       /**
       Услуга "исполняемая программа" используется управляющей системой для
       инициализации выполнения программы, представленной в объекте "исполняемая
       программа". Для выполнения операции "исполняемая программа" сама исполняемая
       программа должна находиться во внутреннем состоянии "инсталлировано",
       административное состояние должно иметь значение "разблокировано",
       операционное состояние должно иметь значение "активизировано", а состояние
       использования должно иметь значение либо "активное", либо "холостое".

       @param name                 Имя экземпляра управляемого объекта "исполняемая
                                   программа"
       @param additionalInfo       Определяет параметры, используемые при
                                   выполнении программы.
       @return                     Успешный запрос будет подтвержден информацией,
                                   включая параметры "идентификатор процесса",
                                   владелец процесса", "начальное стартовое время"
                                   и дополнительные обеспечиваемые параметры.
       ИМЕЕТ МЕСТО, ЕСЛИ экземпляр обеспечивает его
          */

          ExecuteProgramReplyType executeProgram
              (in MONameType name,
              in AdditionalInformationSetType additionalInfo)
              raises (itut_x780::ApplicationError,
                  NOexecuteProgramPackage,
                  OperationStateMismatch,
                  ExecuteProgramSoftwareProcessingFailure);

     }; // interface ExecutableSoftware_F

/**
```

9.12 Интерфейс производственной исполняемой программы

```
*/
     /**
     Создание и удаление  исполняемого модуля
     */

     interface ExecutableSoftwareFactory : itut_x780::ManagedObjectFactory
     {
       itut_x780::ManagedObject create
          (in NameBindingType nameBinding,
          in MONameType superior,
          in string reqID,    // автонаименование в случае пустой строки
          out MONameType name,
          in StringSetType packageNameList,
          in AdministrativeStateType adminstrativeState,
              // conditional
              // itut_m3120::administrativeOperationalStatePackage
              // GET-REPLACE
          in AlarmSeverityAssignmentProfileNameType profile,
              // conditional
              // itut_m3120::alarmSeverityAssignmentPointerPackage
              // GET-REPLACE
          in Istring userLabel,
              // conditional
              // itut_m3120::userLabelPackage
              // GET-REPLACE
          in Istring vendorName,
              // conditional
              // itut_m3120::vendorNamePackage
              // GET-REPLACE
          in Istring version,
              // conditional
              // itut_m3120::versionPackage
              // GET-REPLACE
          in ArcProbableCauseSetType arcProbableCauseList,
              // conditional
              // itut_m3120::arcPackage
              // GET-REPLACE, ADD-REMOVE
          in ArcIntervalProfileNameType arcIntervalProfilePointer,
              // conditional
              // itut_m3120::arcPackage
              // GET-REPLACE
          in ArcTimeType arcManagementRequestedInterval,
              // conditional
              // itut_m3120::arcPackage
              // GET-REPLACE
          in AvailabilityStatusSetType availabilityStatus,
              // GET
              // SET-BY-CREATE
          in FutureAutoBackupTriggerThresholdType
              futureAutoBackupTriggerThreshold,
              // GET-REPLACE
              // informationAutoBackupPackage
          in FutureAutoBackupDestinationType futureAutoBackupDestination,
              // GET-REPLACE
              // informationAutoBackupPackage
          in FutureAutoRestoreSourceType futureAutoRestoreSource,
              // GET-REPLACE
              // informationAutoRestorePackage
          in FutureAutoRestoreAllowedType futureAutoRestoreAllowed,
              // GET-REPLACE
              // informationAutoRestorePackage
```

```
        in NoteFieldType noteField
            // GET-REPLACE
            // noteFieldPackage
        )
        raises (itut_x780::ApplicationError,
            itut_x780::CreateError);

    }; // interface ExecutableSoftwareFactory

/**
```

9.13 Дистрибьютор программных средств

```
*/
    /**
    Этот тип значения используется для получения всех атрибутов
    */

    valuetype SoftwareDistributorValueType :
        truncatable itut_x780::ManagedObjectValueType
    {
      public AdministrativeStateType administrativeState;
        // GET-REPLACE
      public OperationalStateType operationalState;
        // GET

    }; // valuetype SoftwareDistributorValueType

    /**
    Управляемый объект "дистрибьютор программных средств" - это управляемый
    объект, который распространяет программные средства в заданную управляемую
    систему при получении им операции "доставка" из управляющей системы. При
    завершении распространения этот управляемый объект сообщает результат
    распространения. При изменении значения административного состояния или
    операционного состояния должно быть выдано уведомление "изменение состояния",
    определенное в Рекомендации МСЭ-Т X.780
    */

/**
```

9.14 Интерфейс дистрибьютора программных средств

```
*/
    /**
    Управляемый объект "дистрибьютор программных средств"
    */

    interface SoftwareDistributor : itut_x780::ManagedObject
    {
      /**
      Административное состояние описано в Рекомендации МСЭ-Т X.731
      */

      AdministrativeStateType administrativeStateGet ()
        raises (itut_x780::ApplicationError);

      void administrativeStateSet
        (in AdministrativeStateType administrativeState)
        raises (itut_x780::ApplicationError);

      /**
      Операционное состояние описано в Рекомендации МСЭ-Т X.731
      */
```

```
    OperationalStateType operationalStateGet ()
        raises (itut_x780::ApplicationError);

    /**
Услуга "доставка" используется управляющей системой для запроса
распространения программных средств или набора программных средств.
Информация операции "доставка" идентифицирует программные средства,
подлежащие доставке. Результат выполнения операции "доставка" состоит в том,
что копия заданных элементов программных средств доставляется заданной
системе во внутреннем состоянии "доставлено".

Пакетирование программных средств и выбор механизма передачи являются
локальным вопросом и не входят в предмет рассмотрения настоящей Рекомендации
МСЭ-Т. Например, эта информация может быть заранее сконфигурирована  или
определена в операции "доставка" наряду с другой необходимой информацией.

Результат успешного выполнения состоит в том, что  программные средства,
подлежащие распространению, копируются в заданную систему; это может
привести к созданию объектов "программный модуль" и/или "исполняемая
программа". После выполнения команды выдается уведомление "результат
доставки".

@param deliverId            Этот факультативный параметр указывает
                            уникальный идентификатор этой операции
                            "доставка".
@param targetSoftware       Указывает источник программных средств,
                            подлежащих доставке.
@param targetSystem         Указывает факультативный заданный адресат
                            программных средств, подлежащих доставке. Если
                            это незаданный адресат, система использует
                            локальные средства для определения заданного
                            адресата.
@param transferInfo         Специфичный для приложения механизм передачи.
@param additionalInfo       Дополнительная специфичная для приложения
                            информация.
    */

void deliver
    (in DeliverIdTypeOpt deliverId,
    in TargetSoftwareSetType targetSoftware,
    in DestinationType targetSystem,
    in TransferInfoType transferInfo,
    in AdditionalInformationSetType additionalInfo)
    raises (itut_x780::ApplicationError,
         OperationStateMismatch,
         DeliverSoftwareProcessingFailure);

/**
При завершении операции "доставка" управляемый объект выдает уведомление
"результат доставки". Он содержит окончательные разультаты операции и может
указывать либо успешность, либо безуспешность выполнения.
*/

MANDATORY_NOTIFICATION(
    itut_x744d1::Notifications, deliverResultNotification)
MANDATORY_NOTIFICATION(
    itut_x780::Notifications, objectCreation)
MANDATORY_NOTIFICATION(
    itut_x780::Notifications, objectDeletion)
MANDATORY_NOTIFICATION(
    itut_x780::Notifications, stateChange)

}; // interface SoftwareDistributor

/**
```

9.15 Интерфейс дистрибьютора программных средств _F

```
*/
     /**
     Управляемый объект "фасад распределителя программных средств – см.
Рекомендацию МСЭ-Т X.780.1
     */

     interface SoftwareDistributor_F : itut_x780::ManagedObject_F
     {
       /**
       административное состояние описано в Рекомендации МСЭ-Т X.731
       */

       AdministrativeStateType administrativeStateGet
          (in MONameType name)
          raises (itut_x780::ApplicationError);

       void administrativeStateSet
          (in MONameType name,
          in AdministrativeStateType administrativeState)
          raises (itut_x780::ApplicationError);

       /**
       операционное состояние описано в Рекомендации МСЭ-Т X.731
       */

       OperationalStateType operationalStateGet
          (in MONameType name)
          raises (itut_x780::ApplicationError);

       /**
       Услуга "доставка" используется управляющей системой для запроса
       распространения программных средств или набора программных средств.
       Информация операции "доставка" идентифицирует программные средства,
       подлежащие доставке. Результат выполнения операции "доставка" состоит в том,
       что копия заданных элементов программных средств доставляется заданной
       системе во внутреннем состоянии "доставлено".

       Пакетирование программных средств и выбор механизма передачи являются
       локальным вопросом и не входят в предмет рассмотрения настоящей Рекомендации
       МСЭ-Т. Например, эта информация может быть заранее сконфигурирована  или
       определена в операции "доставка" наряду с другой необходимой информацией.

       Результат успешного выполнения состоит в том, что  программные средства,
       подлежащие распространению, копируются в заданную систему; это может
       привести к созданию объектов "программный модуль" и/или "исполняемая
       программа". После выполнения команды выдается уведомление "результат
       доставки".

       @param name                Имя экземпляра управляемого объекта
                                  "дистрибьютор программных средств"
       @param deliverId           Этот факультативный параметр указывает
                                  уникальный идентификатор данной операции
                                  "доставка".
       @param targetSoftware      Указывает источник программных средств,
                                  подлежащих доставке.
       @param targetSystem        Указывает факультативный заданный адресат
                                  программных средств, подлежащих доставке. Если
                                  это незаданный адресат, система использует
                                  локальные средства для определения заданного
                                  адресата.
       @param transferInfo        Специфичный для приложения механизм передачи.
       @param additionalInfo      Дополнительная специфичная для приложения
                                  информация.
```

```
     */

       void deliver
          (in MONameType name,
          in DeliverIdTypeOpt deliverId,
          in TargetSoftwareSetType targetSoftware,
          in DestinationType targetSystem,
          in TransferInfoType transferInfo,
          in AdditionalInformationSetType additionalInfo)
          raises (itut_x780::ApplicationError,
           OperationStateMismatch,
           DeliverSoftwareProcessingFailure);

       /**
       При завершении операции "доставка" управляемый объект выдает уведомление
       "результат доставки". Он содержит окончательные разультаты операции и может
       указывать либо успешность, либо безуспешность выполнения.
       */

       MANDATORY_NOTIFICATION(
          itut_x744d1::Уведомления, deliverResultNotification)
       MANDATORY_NOTIFICATION(
          itut_x780::Уведомления, objectCreation)
       MANDATORY_NOTIFICATION(
          itut_x780::Уведомления, objectDeletion)
       MANDATORY_NOTIFICATION(
          itut_x780::Уведомления, состояниеChange)

     }; // interface SoftwareDistributor_F

/**
```

9.16 Интерфейс производственного дистрибьютора программных средств

```
*/
     /**
     Создание и удаление дистрибьютора программных средств
     */

     interface SoftwareDistributorFactory : itut_x780::ManagedObjectFactory
     {
       itut_x780::ManagedObject create
          (in NameBindingType nameBinding,
          in MONameType superior,
          in string reqID,    // автонаименование при пустой строке
          out MONameType name,
          in StringSetType packageNameList,
          in AdministrativeStateType administrativeState
               // GET-REPLACE
          )
          raises (itut_x780::ApplicationError,
               itut_x780::CreateError);

     }; // interface SoftwareDistributorFactory

/**
```

9.17 Уведомления

```
*/

     interface Notifications
     {
       /**
       Уведомление "отчет о резервировании" выдается для информировании о
       резервировании информации, представленной данным объектом.

       @param eventTime            Текущее время управляемой системы.
       @param source               Объект, выдающий уведомление.
       @param request              Указывает, кем выдан запрос, либо автоматически
                                   управляющей, либо управляемой системой
       @param backupResult         Указывает одно из следующих в зависимости от
                                   успешности или безуспешности резервирования и от
                                   значения параметров операции резервирования:
                                   - Безуспешный выбор – означает ошибку приложения.
                                   - Выбор в автономном режиме – местоположение
                                   резервирования в автономном режиме.
                                   - Выбор локального объекта – экземпляр объекта
                                   резервирования.
       */

       void backupReport (
          in ExternalTimeType eventTime,
          in MONameType source,
          in RequestType request,
          in BackupResultType backupResult
       );

       /**
       При завершении операции "доставка" управляемый объект выдает уведомление
       "результат доставки".

       @param eventTime            Текущее время управляемой системы.
       @param source               Объект, выдающий уведомление.
       @param deliverId            Уникальный идентификатор доставки, поставляемый с
                                   операцией "доставка" (если она обеспечивается).
       @param deliver              Результат операции "доставка". Она относится к
                                   типу UIDType, поэтому результаты могут быть
                                   локализованными. Результаты доставки,
                                   обеспечиваемые в настоящей Рекомендации,
                                   приведены в модуле DeliverResultConst.
       @param additionalInfo       Специфичная для приложения информация.
       */

       void deliverResultNotification (
          in ExternalTimeType eventTime,
          in MONameType source,
          in DeliverIdTypeOpt deliverId,
          in DeliverResultType deliver,
          in AdditionalInformationSetType additionalInfo
       );

       /**
       Уведомление "отчет о восстановлении" выдается для информирования о
       восстановлении управляемого объекта после предыдущего резервирования.

       @param eventTime            Текущее время управляемой системы.
       @param source               Объект, выдающий уведомление.
       @param request              Указывает, кем выдан запрос, либо автоматически
                                   управляющей, либо управляемой системой
```

```
@param restoreResult        Указывает одно из следующих в зависимости от
                            успешности или безуспешности резервирования и от
                            значения параметров операции резервирования:
                            - Безуспешный выбор – означает ошибку приложения.
                            - Выбор в автономном режиме – местоположение
                            резервирования в автономном режиме.
                            - Выбор локального объекта – экземпляр объекта
                            резервирования.
*/

void restoreReport (
   in ExternalTimeType eventTime,
   in MONameType source,
   in RequestType request,
   in ВосстановлениеResultType restoreResult
);

/**
Уведомление "отчет о проверке правильности" выдается для информирования о
проверке правильности управляемого объекта.

@param eventTime            Текущее время управляемой системы.
@param source               Объект, выдающий уведомление.
@param validateInfo         Указывает аргумент операции "проверка правильности"
@param validateResult       Указывает одно из следующих в зависимости от
                            успешности или безуспешности выполнения операции
                            "проверка правильности":
                            - Прохождение проверки правильности (с указанием
                            результата)
                            - Безуспешность проверки правильности (с
                            указанием результата)
                            - Прохождение проверки правильности
                            - Безуспешность проверки правильности (с указанием
                            ошибки)
*/

void validateReport (
   in ExternalTimeType eventTime,
   in MONameType source,
   in ValidateInfoType validateInfo,
   in ValidateResultType validateResult
   );
};

/**
Константы уведомления
*/

const string backupReportTypeName =
   "itut_x744d1:: Notifications::backupReport";
const string deliverResultNotificationTypeName =
   "itut_x744d1:: Notifications::deliverResultNotification";
const string restoreReportTypeName =
   "itut_x744d1:: Notifications::restoreReport";
const string validateReportTypeName =
   "itut_x744d1:: Notifications::validateReport";
const string additionalInfoName = "additionalInfo";
const string backupResultName = "backupResult";
const string deliverIdName = "deliverId";
const string deliverName = "deliver";
const string eventTimeName = "eventTime";
const string sourceName = "source";
```

```
      const string restoreResultName = "restoreResult";
      const string requestName = "request";
      const string validateInfoName = "validateInfo";
      const string validateResultName = "validateResult";

/**
```

9.18 Связка имен

```
*/
     module NameBinding
     {
       /**
       Эта связка имен используется для  присвоения имени объекту "дистрибьютор
       программных  средств"  для объекта ManagedElement.
       */

       module SoftwareDistributor_ManagedElement
       {
          const string superiorClass = "itut_m3120::ManagedElement";
          const boolean superiorSubclassesAllowed = TRUE;
          const string subordinateClass =
           "itut_x744d1::SoftwareDistributor";
          const boolean subordinateSubclassesAllowed = TRUE;
          const boolean managerCreatesAllowed = TRUE;
          const DeletePolicyType deletePolicy =
              itut_x780::deleteContainedObjects;
          const string kind = "SoftwareDistributor";

       }; // module SoftwareDistributor_ManagedElement

       /**
       Эта связка имен используется для  присвоения имени объекту "дистрибьютор
       программных средств"  для объекта Subsystem.
       */

       module SoftwareDistributor_Subsystem
       {
          const string superiorClass = "itut_x780::Subsystem";
          const boolean superiorSubclassesAllowed = TRUE;
          const string subordinateClass =
              "itut_x744d1::SoftwareDistributor";
          const boolean subordinateSubclassesAllowed = TRUE;
          const boolean managerCreatesAllowed = TRUE;
          const DeletePolicyType deletePolicy =
              itut_x780::deleteContainedObjects;
          const string kind = "SoftwareDistributor";

       }; // module SoftwareDistributor_Subsystem

       /**
       Эта связка имен используется для  присвоения имени объекту "дистрибьютор
       программных средств"  для объекта System.
       */

       module SoftwareDistributor_System
       {
          const string superiorClass = "itut_x780::System";
          const boolean superiorSubclassesAllowed = TRUE;
```

```
   const string subordinateClass =
       "itut_x744d1::SoftwareDistributor";
   const boolean subordinateSubclassesAllowed = TRUE;
   const boolean managerCreatesAllowed = TRUE;
   const DeletePolicyType deletePolicy =
       itut_x780::deleteContainedObjects;
   const string kind = "SoftwareDistributor";

}; // module SoftwareDistributor_System

/**
Эта связка имен используется для  присвоения имени объекту "программный
модуль" для объекта Equipment.
*/

module SoftwareUnit_Equipment
{
   const string superiorClass = "itut_m3120::Equipment";
   const boolean superiorSubclassesAllowed = TRUE;
   const string subordinateClass =
       "itut_x744d1::SoftwareUnit";
   const boolean subordinateSubclassesAllowed = TRUE;
   const boolean managerCreatesAllowed = TRUE;
   const DeletePolicyType deletePolicy =
       itut_x780::deleteContainedObjects;
   const string kind = "SoftwareUnit";

}; // module SoftwareUnit_Equipment

/**
Эта связка имен используется для  присвоения имени объекту "программный
модуль" для объекта ManagedElement.
*/

module SoftwareUnit_ManagedElement
{
   const string superiorClass = "itut_m3120::ManagedElement";
   const boolean superiorSubclassesAllowed = TRUE;
   const string subordinateClass =
       "itut_x744d1::SoftwareUnit";
   const boolean subordinateSubclassesAllowed = TRUE;
   const boolean managerCreatesAllowed = TRUE;
   const DeletePolicyType deletePolicy =
       itut_x780::deleteContainedObjects;
   const string kind = "SoftwareUnit";

}; // module SoftwareUnit_ManagedElement

/**
Эта связка имен используется для  присвоения имени объекту "программный
модуль" для объекта Subsystem.
*/

module SoftwareUnit_Subsystem
{
   const string superiorClass = "itut_x780::Subsystem";
   const boolean superiorSubclassesAllowed = TRUE;
   const string subordinateClass =
    "itut_x744d1::SoftwareUnit";
   const boolean subordinateSubclassesAllowed = TRUE;
   const boolean managerCreatesAllowed = TRUE;
   const DeletePolicyType deletePolicy =
    itut_x780::deleteContainedObjects;
   const string kind = "SoftwareUnit";

}; // module SoftwareUnit_Subsystem
```

```
/**
Эта связка имен используется для  присвоения имени объекту "программный
модуль" для объекта System.
*/

module SoftwareUnit_System
{
   const string superiorClass = "itut_x780::System";
   const boolean superiorSubclassesAllowed = TRUE;
   const string subordinateClass =
       "itut_x744d1::SoftwareUnit";
   const boolean subordinateSubclassesAllowed = TRUE;
   const boolean managerCreatesAllowed = TRUE;
   const DeletePolicyType deletePolicy =
       itut_x780::deleteContainedObjects;
   const string kind = "SoftwareUnit";

}; // module SoftwareUnit_System

/**
Эта связка имен используется для  присвоения имени объекту "программный
модуль" для объекта Equipment.
*/

module ExecutableSoftware_Equipment
{
   const string superiorClass = "itut_m3120::Equipment";
   const boolean superiorSubclassesAllowed = TRUE;
   const string subordinateClass =
       "itut_x744d1::ExecutableSoftware";
   const boolean subordinateSubclassesAllowed = TRUE;
   const boolean managerCreatesAllowed = TRUE;
   const DeletePolicyType deletePolicy =
       itut_x780::deleteContainedObjects;
   const string kind = "ExecutableSoftware";

}; // module ExecutableSoftware_Equipment

/**
Эта связка имен используется для  присвоения имени объекту "исполняемая
программа" для объекта ManagedElement.
*/

module ExecutableSoftware_ManagedElement
{
   const string superiorClass = "itut_m3120::ManagedElement";
   const boolean superiorSubclassesAllowed = TRUE;
   const string subordinateClass =
       "itut_x744d1::ExecutableSoftware";
   const boolean subordinateSubclassesAllowed = TRUE;
   const boolean managerCreatesAllowed = TRUE;
   const DeletePolicyType deletePolicy =
       itut_x780::deleteContainedObjects;
   const string kind = "ExecutableSoftware";

}; // module ExecutableSoftware_ManagedElement
```

```
    /**
    Эта связка имен используется для  присвоения имени объекту "исполняемая
    программа" для объекта Subsystem.
    */

    module ExecutableSoftware_Subsystem
    {
       const string superiorClass = "itut_x780::Subsystem";
       const boolean superiorSubclassesAllowed = TRUE;
       const string subordinateClass =
           "itut_x744d1::ExecutableSoftware";
       const boolean subordinateSubclassesAllowed = TRUE;
       const boolean managerCreatesAllowed = TRUE;
       const DeletePolicyType deletePolicy =
           itut_x780::deleteContainedObjects;
       const string kind = "ExecutableSoftware";

    }; // module ExecutableSoftware_Subsystem

    /**
    Эта связка имен используется для  присвоения имени объекту "исполняемая
    программа" для объекта System.
    */

    module ExecutableSoftware_System
    {
       const string superiorClass = "itut_x780::System";
       const boolean superiorSubclassesAllowed = TRUE;
       const string subordinateClass =
           "itut_x744d1::ExecutableSoftware";
       const boolean subordinateSubclassesAllowed = TRUE;
       const boolean managerCreatesAllowed = TRUE;
       const DeletePolicyType deletePolicy =
           itut_x780::deleteContainedObjects;
       const string kind = "ExecutableSoftware";

    }; // module ExecutableSoftware_System

  }; // module NameBinding

/**
```

9.19 Модуль "постоянный тип файла"

```
*/

  module FileTypeConst
  {
    const string moduleName = "itut_x744d1::FileTypeConst";

    const short unstructuredText = 0; // FTAM-1
    const short unstructuredBinary = 1; // FTAM-3
    const short blockSpecial = 2;

  }; // end of module FileTypeConst

/**
```

9.20 Модуль "постоянный тип доставки"

```
*/

     module DeliverResultConst
     {
       const string moduleName = "itut_x744d1::DeliverResultConst";

       const short pass = 0;
       const short communicationError = 1;
       const short equipmentError = 2;
       const short qosError = 3;
       const short accessDenied = 4;
       const short notFound = 5;
       const short insufficientSpace = 6;
       const short alreadyDelivered = 7;
       const short inProgress = 8;
       const short unknown = 9;

     }; // end of module DeliverResultConst

}; // module itut_x744d1

#endif //_itut_x744_1_idl_
```

СЕРИИ РЕКОМЕНДАЦИЙ МСЭ-Т

Серия A	Организация работы МСЭ-Т
Серия B	Средства выражения: определения, символы, классификация
Серия C	Общая статистика электросвязи
Серия D	Общие принципы тарификации
Серия E	Общая эксплуатация сети, телефонная служба, функционирование служб и человеческие факторы
Серия F	Нетелефонные службы электросвязи
Серия G	Системы и средства передачи, цифровые системы и сети
Серия H	Аудиовизуальные и мультимедийные системы
Серия I	Цифровая сеть с интеграцией служб
Серия J	Кабельные сети и передача сигналов телевизионных и звуковых программ и других мультимедийных сигналов
Серия K	Защита от помех
Серия L	Конструкция, прокладка и защита кабелей и других элементов линейно-кабельных сооружений
Серия M	TMN и техническое обслуживание сетей: международные системы передачи, телефонные, телеграфные, факсимильные и арендованные каналы
Серия N	Техническое обслуживание: международные каналы передачи звуковых и телевизионных программ
Серия O	Требования к измерительной аппаратуре
Серия P	Качество телефонной передачи, телефонные установки, сети местных линий
Серия Q	Коммутация и сигнализация
Серия R	Телеграфная передача
Серия S	Оконечное оборудование для телеграфных служб
Серия T	Оконечное оборудование для телематических служб
Серия U	Телеграфная коммутация
Серия V	Передача данных по телефонной сети
Серия X	**Сети передачи данных и взаимосвязь открытых систем**
Серия Y	Глобальная информационная инфраструктура и аспекты межсетевого протокола (IP)
Серия Z	Языки и общие аспекты программного обеспечения для систем электросвязи

Отпечатано в Швейцарии
Женева, 2004 г.