РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 365 972** (13) **C2**

(51) МПК
G06F 1/00 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2004127215/09, 10.09.2004**

(24) Дата начала отсчета срока действия патента:
10.09.2004

(30) Конвенционный приоритет:
24.10.2003 US 10/693,718

(43) Дата публикации заявки: **20.02.2006**

(45) Опубликовано: **27.08.2009** Бюл. № **24**

(56) Список документов, цитированных в отчете о поиске: **US 2003/0028685 A1, 06.02.2003. US 2003/0177282 A1, 18.09.2003. RU 2127019 C1, 27.02.1999. US 2003/0084209 A1, 01.05.2003. US 6446253 B1, 03.09.2002.**

Адрес для переписки:
129090, Москва, ул. Б.Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. Ю.Д.Кузнецову, рег.№ 595

(72) Автор(ы):
ГУЗАК Крис Дж. (US),
КАРАТАЛ Керем Б. (US),
МИЛЛЕР Марк М. (US),
ШЕЛДОН Майкл Г. (US),
МАККИ Тимоти П. (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ КОРПОРЕЙШН (US)

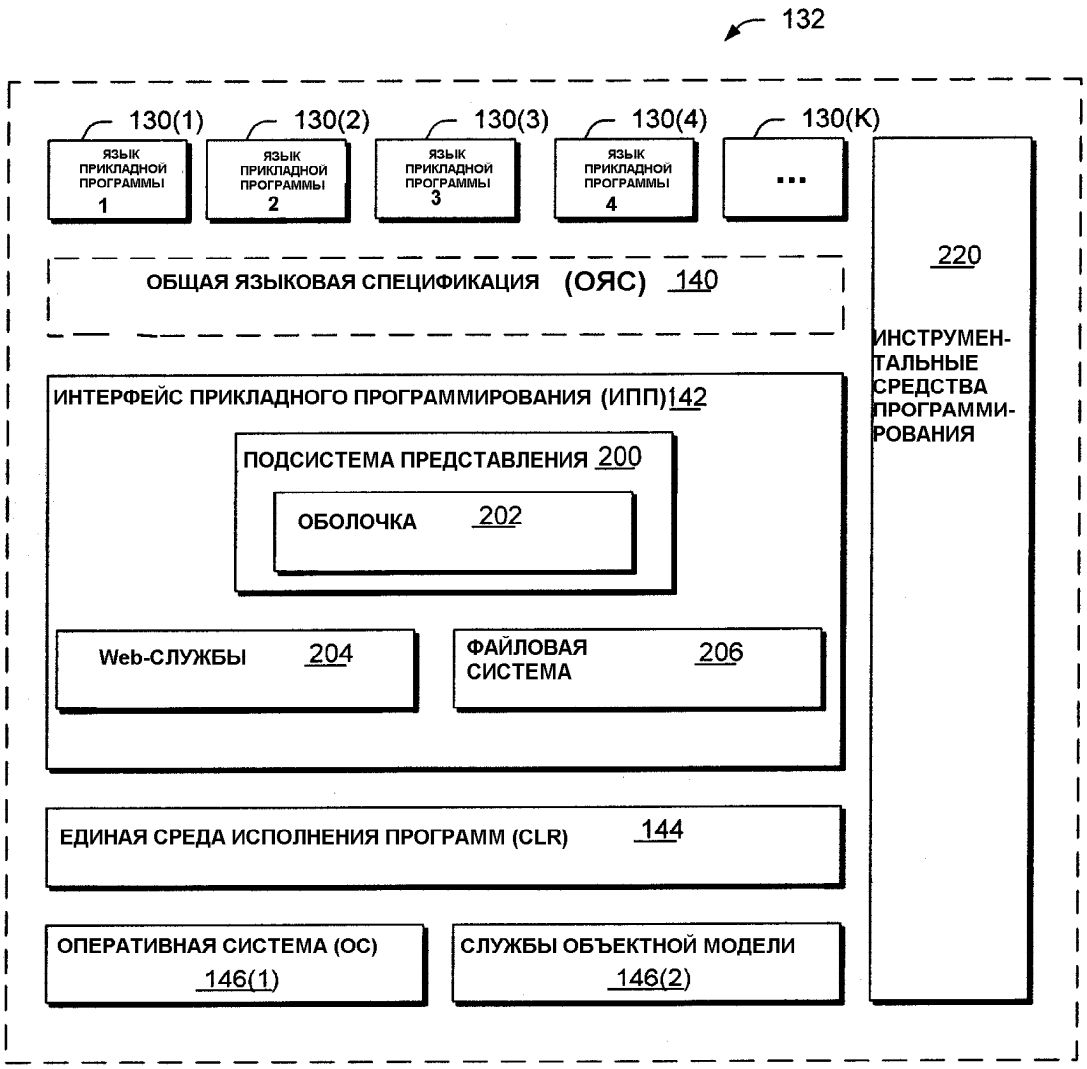
(54) ИНТЕРФЕЙС ПРОГРАММИРОВАНИЯ ДЛЯ КОМПЬЮТЕРНОЙ ПЛАТФОРМЫ

(57) Реферат:

Изобретение относится к интерфейсу прикладного программирования для сетевой платформы, на которой разработчики могут создавать веб-приложения и веб-службы. Техническим результатом изобретения является повышение эффективности рабочих характеристик интерфейса программирования. Технический результат достигается благодаря тому, что интерфейс программирования включает в себя различные функциональные возможности, а именно первую группу услуг, относящихся к повторно используемым элементам управления пользовательского интерфейса, вторую группу услуг, относящихся к диалогам пользовательского интерфейса и мастерам пользовательского интерфейса, третью группу услуг, относящуюся к расширению функциональных возможностей пользовательского интерфейса, и четвертую группу услуг, относящуюся к расширению функциональных возможностей рабочего стола пользовательского интерфейса. 2 н. и 9 з.п. ф-лы, 16 ил.

RU 2365972 C2

RU 2365972 C2



ФИГ.2

RU 2365972 C2

RU 2365972 C2

RUSSIAN FEDERATION

(19) **RU** (11) **2 365 972** (13) **C2**

(51) Int. Cl.
G06F 1/00 (2006.01)

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2004127215/09, 10.09.2004**

(24) Effective date for property rights:
10.09.2004

(30) Priority:
24.10.2003 US 10/693,718

(43) Application published: **20.02.2006**

(45) Date of publication: **27.08.2009 Bull. 24**

Mail address:
129090, Moskva, ul. B.Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. Ju.D.Kuznetsovu, reg.№ 595

(72) Inventor(s):
GUZAK Kris Dzh. (US),
KARATAL Kerem B. (US),
MILLER Mark M. (US),
ShELDON Majkl G. (US),
MAKKI Timoti P. (US)

(73) Proprietor(s):
MAJKROSOFT KORPOREJShN (US)

(54) PROGRAMMING INTERFACE FOR COMPUTING PLATFORM

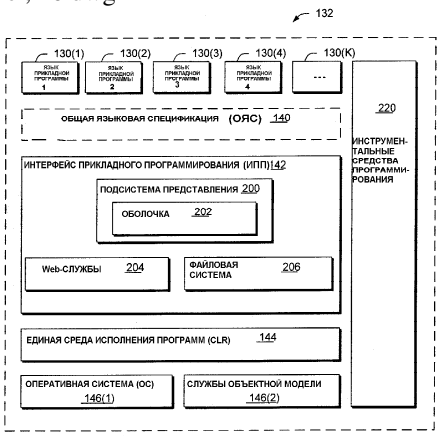
(57) Abstract:

FIELD: information technology.

SUBSTANCE: invention relates to an application programming interface for a networking platform, on which programmers can create web applications and web services. The technical outcome is achieved due to that, the programming interface includes different functionalities, and more specifically, a first group of services, related to repeatedly used user interface control elements, a second group of services, related to user interface dialogs and user interface wizards, a third group of services, related to expansion of user interface functionalities and a fourth group of services, related to expansion of functionalities of user interface desktop.

EFFECT: increased effectiveness of operating characteristics of programming interface.

11 cl, 16 dwg



ФИГ.2

RU 2365972 C2

RU 2365972 C2

Область техники

Настоящее изобретение относится к программному обеспечению и к разработке такого программного обеспечения. В частности, настоящее изобретение относится к интерфейсу программирования, который облегчает использование программной платформы прикладными программами и аппаратными средствами компьютера.

Краткое описание прилагаемых компакт-дисков

К данному описанию изобретения прилагается комплект из трех компакт-дисков, на которых хранится набор инструментальных средств для разработки программного обеспечения (НРПО) для операционной системы Microsoft® Windows® под кодовым названием «Longhorn». НРПО содержит документацию для операционной системы Microsoft® Windows® под кодовым названием «Longhorn». К данному описанию изобретения также прилагается второй экземпляр каждого из данных трех компакт-дисков.

Первый компакт-диск комплекта из трех компакт-дисков (КД) (КД 1 из 3) включает в себя папку файлов с именем «lhsdk», которая была создана 22 октября 2003 г.; размер ее составляет 586 Мбайт, она содержит 9692 подпапок и содержит 44 292 подфайла. Второй компакт-диск комплекта из трех компакт-дисков (КД 2 из 3) включает в себя папку файлов с именем «ns», которая была создана 22 октября 2003 г.; размер ее составляет 605 Мбайт, она содержит 12628 подпапок и она содержит 44934 подфайла. Третий компакт-диск комплекта из трех компакт-дисков (КД 3 из 3) включает в себя папку файлов с именем «ns», которая была создана 22 октября 2003 г.; размер ее составляет 575 Мбайт, она содержит 9881 подпапку и она содержит 43630 подфайлов. Файлы на каждом из данных трех компакт-дисков могут исполняться на вычислительном устройстве, работающем в операционной системе Windows® (например, персональный компьютер компании IBM или эквивалентный ему), которое исполняет операционную систему с торговой маркой Windows® (например, Windows® NT, Windows® 98, Windows® 2000, Windows® XP и т.д.). Файлы на каждом компакт-диске данного комплекта из трех компакт-дисков, таким образом, включаются этим в качестве ссылки.

Каждый компакт-диск комплекта из трех компакт-дисков сам по себе представляет собой компакт-диск с однократной записью, и он соответствует стандарту ISO 9660. Содержимое каждого компакт-диска комплекта из трех компакт-дисков находится в соответствии с американским стандартным кодом для обмена информацией (АСКОИ, ASCII).

Предшествующий уровень техники

Раньше программное обеспечение компьютеров можно было классифицировать как программное обеспечение «операционных систем», или «прикладное» программное обеспечение. В общих чертах, прикладная программа представляет собой программное обеспечение, предназначенное для выполнения конкретной задачи для пользователя компьютера, такой как решение математического уравнения или поддержка обработки текста. Операционная система представляет собой программное обеспечение, которое координирует и управляет аппаратными средствами компьютера. Целью операционной системы является обеспечение доступности ресурсов компьютера для программиста прикладных программ, в то же самое время скрывая сложность, необходимую для фактического управления аппаратными средствами.

Операционная система делает ресурсы доступными посредством функций, которые вместе известны как интерфейс прикладного программирования, или ИПП (API).

Термин ИПП также используется с ссылкой на единственную одну из данных функций. Функции обычно группируются на основе того, какой ресурс или службу они предоставляют программисту прикладных программ. Прикладное программное обеспечение запрашивает ресурсы посредством вызова отдельных функций ИПП. Функции ИПП также служат в качестве средства, посредством которого сообщения и информация, предоставляемые операционной системой, передаются обратно прикладному программному обеспечению.

В дополнение к изменениям в аппаратных средствах другим фактором, стимулирующим эволюцию программного обеспечения операционных систем, было желание упростить и ускорить разработку прикладного программного обеспечения. Разработка прикладного программного обеспечения может представлять собой приводящую в уныние задачу, иногда требующую годы времени работы разработчика для создания сложной программы с миллионами строк кода. Для популярной операционной системы, такой как различные версии операционной системы Microsoft Windows®, разработчики прикладного программного обеспечения пишут тысячи различных прикладных программ каждый год, которые используют операционную систему. Требуется согласованная и удобная база операционной системы для поддержки такого большого количества разработчиков разнообразных прикладных программ.

Часто разработку прикладного программного обеспечения можно упростить выполнением операционной системы более сложной. Т.е. если функция может использоваться несколькими различными прикладными программами, то может быть лучшим решением написать ее один раз для включения в операционную систему, чем требуя того, чтобы множество разработчиков программного обеспечения писали ее множество раз для включения во множество различных прикладных программ. Таким образом, если операционная система поддерживает широкий диапазон общих функциональных возможностей, требуемых рядом прикладных программ, то можно достичь существенной экономии расходов и времени на разработку прикладного программного обеспечения.

Независимо от того, где проводится граница между операционной системой и прикладным программным обеспечением, ясно, что для полезной операционной системы ИПП между операционной системой и аппаратными средствами компьютера и прикладным программным обеспечением также важен, как и эффективная внутренняя работа самой операционной системы.

В течение последних нескольких лет всеобщее принятие Интернета и сетевой технологии в целом изменило общую картину для разработчиков программного обеспечения компьютеров. Традиционно, усилия разработчиков программного обеспечения были обращены на разработку прикладных программ для автономных настольных компьютеров, или компьютеров на базе локальной сети (ЛС), которые были соединены с ограниченным количеством других компьютеров при помощи локальной сети (ЛС). Такие прикладные программы обычно упоминались как «упакованные в термосадочную пленку» продукты, так как программное обеспечение предлагалось и продавалось в упаковке из термоусадочной пленки. Прикладные программы использовали хорошо определенный ИПП для доступа к лежащей в основе операционной системе компьютера.

Так как Интернет эволюционировал и приобрел широко распространенное признание, индустрия начала признавать мощность хостирующих прикладных программ на различных сайтах во Всемирной паутине (или просто Сети). В сетевом

мире клиенты с любого узла могут представить запросы на серверные прикладные программы, хостированные в различных местоположениях сети, и получить обратно ответы за доли секунды. Эти прикладные программы Сети (веб-приложения), однако, обычно разрабатывались с использованием той же платформы операционной системы, что первоначально была разработана для автономных вычислительных машин или компьютеров в локальной сети. К сожалению, в некоторых случаях данные прикладные программы не переходят адекватно в распределенный режим вычислений. Лежащая в основе платформа просто не была построена с целью поддержки неограниченного количества соединенных между собой компьютеров.

Для того чтобы обеспечить переход к распределенной вычислительной среде, вводимой Интернетом, корпорация Microsoft разработала сетевую программную платформу, известную как «.NET» Framework (читается как «Dot Net»). Microsoft®.NET представляет собой программное обеспечение для осуществления связи людей, информации, систем и устройств. Платформа позволяет разработчикам создавать веб-службы, которые исполняются в Интернете. Данный динамический переход сопровождался набором функций ИПП для .NET™ Framework корпорации Microsoft.

Так как использование .NET™ Framework становится все более распространенным, были определены пути повышения эффективности и/или рабочих характеристик платформы. Изобретатели разработали уникальный набор функций интерфейса программирования, делающий возможным такую повышенную эффективность и/или рабочие характеристики.

Сущность изобретения

В данной заявке описывается интерфейс программирования для компьютерной платформы.

В соответствии с некоторыми аспектами интерфейс программирования может включать в себя одну или несколько из следующих групп служб: первая группа служб, относящаяся к повторно используемым элементам управления пользовательского интерфейса, вторая группа служб, относящаяся к диалогам пользовательского интерфейса и мастерам пользовательского интерфейса, третья группа служб, относящаяся к расширению функциональных возможностей пользовательского интерфейса, и четвертая группа служб, относящаяся к расширению функциональных возможностей “рабочего стола” пользовательского интерфейса.

Краткое описание чертежей

Одинаковые позиции используются на чертежах для ссылки на аналогичные признаки.

На фиг.1 представлена сетевая архитектура, в которой клиенты получают доступ к веб-службам через Интернет, используя обычные протоколы.

На фиг.2 представлена блок-схема архитектуры программного обеспечения для сетевой платформы, которая включает в себя интерфейс прикладного программирования (ИПП).

На фиг.3 представлена блок-схема уникальных пространств имен, поддерживаемых ИПП, а также классы функций различных функций ИПП.

На фиг.4 представлена блок-схема примерного компьютера, который может исполнять всю или часть архитектуры программного обеспечения.

На фиг.5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 и 16 представлены различные примерные осуществления интерфейса программирования.

Подробное описание

Настоящее описание относится к интерфейсу программирования, такому как

интерфейс прикладного программирования (ИПП), для сетевой платформы, на которой разработчики могут создавать веб-приложения и веб-службы. Более конкретно, описывается примерный ИПП для операционных систем, которые используют сетевую платформу, такую как .NET™ Framework, созданную корпорацией Microsoft. .NET™ Framework представляет собой программную платформу для веб-служб и веб-прикладных программ (веб-приложений, выполняемых в распределенной вычислительной среде. Она представляет следующее поколение обработки данных в Интернет, используя открытые стандарты передачи данных для передачи данных между слабосвязанными веб-службами, которые работают совместно для выполнения конкретной задачи.

В описанных осуществлениях сетевая платформа использует расширяемый язык разметки (РЯР, XML), открытый стандарт для описания данных. РЯР управляется консорциумом «Всемирной паутины» (КВП, W3C). РЯР используется для определения элементов данных на веб-странице и документах типа «бизнес-бизнес». РЯР использует такую же структуру тегов, что и язык разметки гипертекста (ЯРГТ, HTML);

однако, тогда как ЯРГТ определяет, как элементы отображаются, РЯР определяет, что данные элементы содержат. ЯРГТ использует предварительно определенные теги, а РЯР делает возможным определение тегов разработчиком страницы. Таким образом, фактически, можно определить любые элементы данных, позволяя веб-страницам функционировать как записи баз данных. Посредством использования РЯР и других открытых протоколов, таких как простой протокол доступа к объектам (ППДО, SOAP), сетевая платформа делает возможным интеграцию широкого спектра услуг, которые могут быть адаптированы к потребностям пользователя. Хотя варианты выполнения, описанные в настоящей заявке, описываются в отношении РЯР и других открытых стандартов, они не требуются для работы заявляемого изобретения. Другие в равной степени пригодные технологии достаточны для выполнения изобретений, описанных в настоящей заявке.

Используемый в настоящей заявке признак “интерфейс прикладного программирования” или ИПП (API)) включает в себя традиционные интерфейсы, которые используют вызовы метода или функции, а также удаленные вызовы (например, прокси-объект, фиктивное отношение) и вызовы ППДО/РЯР.

Примерная сетевая среда

На фиг.1 представлена сетевая среда 100, в которой может быть реализована сетевая платформа, такая как .NET™ Framework. Сетевая среда 100 включает в себя представительные веб-службы 102(1), … 102(N), которые предоставляют службы, к которым можно получить доступ по сети 104 (например, Интернет). Веб-службы, обозначаемые в целом позицией 102, представляют собой программируемые прикладные компоненты, которые являются повторно используемыми и взаимодействуют программным путем по сети 104, обычно при помощи веб-протоколов промышленного стандарта, таких как РЯР, ППДО, протокол передачи информации в Интернет приложений (ПБП, WAP), протокол передачи гипертекста (ППГТ, HTTP) и простой протокол электронной почты (ППЭП, SMTP), хотя также могут использоваться другие средства взаимодействия с веб-службами по сети, такие как удаленный вызов процедуры (УВП, RPC) или технология типа брокера объектных запросов. Веб-служба может быть самоописываемой и часто определяется на языке форматов и упорядочения сообщений.

Веб-службы 102 являются доступными непосредственно для других служб (как представлено линией 106 передачи данных) или прикладной программы, такой как

веб-прикладная программа 110 (как представленная линиями 112 и 114 передачи данных). Каждая веб-служба 102 изображается как включающая в себя один или несколько серверов, которые исполняют программное обеспечение для обработки запросов на конкретные службы. Такие службы часто поддерживают базы данных, в которых хранится информация для предоставления в ответ запросчикам. Веб-службы могут конфигурироваться для выполнения любой одной из многочисленных других служб. Примеры веб-служб включают в себя верификацию регистрационного имени, уведомление, хранение базы данных, котировки акций, каталоги местоположения, отображение, музыка, электронный бумажник, календарь/планировщик, списки телефонов, новости и информация, игры, продажа и покупка билетов и т.д. Веб-службы могут объединяться друг с другом и с другими прикладными программами для построения интеллектуальных интерактивных практических разработок.

Сетевая среда 100 также включает в себя представительные (примерные) клиентские устройства 120 (1), 120 (2), 120 (3), 120 (4) … 120 (M), которые используют веб-службы 102 (как представлено линией 122 передачи данных), и/или веб-прикладную программу 110 (как представлено линиями 124, 126 и 128 передачи данных). Клиенты могут передавать данные друг другу, используя также стандартные протоколы, как представлено примерной линией 130 передачи данных РЯР между клиентами 120 (3) и 120 (4).

Клиентские устройства, обозначенные в целом позицией 120, могут быть выполнены многими различными путями. Примеры возможных осуществлений клиента включают в себя, без ограничения, портативные компьютеры, стационарные компьютеры, планшетные персональные компьютеры, телевизоры/телевизионные приставки, беспроводные устройства связи, персональные цифровые помощники, игровые консоли, принтеры, фотокопировальные устройства и другие интеллектуальные устройства.

Веб-приложение 110 представляет собой прикладную программу, предназначенную для выполнения на сетевой платформе, и может использовать веб-службы 102 при обработке и обслуживании запросов от клиентов 120. Веб-приложение 110 состоит из одной или нескольких прикладных программ 130, которые выполняются поверх инфраструктуры 132 программирования, которые исполняются на одном или нескольких серверах 134 или других компьютерных системах. Отметьте, что часть веб-приложение 110 может фактически постоянно находиться на одном или нескольких клиентах 120. Альтернативно, веб-приложение 110 может координироваться с другим программным обеспечением на клиентах 120 для фактического выполнения своих задач.

Инфраструктура 132 программирования представляет собой структуру, которая поддерживает прикладные программы и службы, разработанные разработчиками прикладных программ. Она позволяет производить многоязыковую разработку и “бесшовную” (гладкую) интеграцию посредством поддержки многочисленных языков. Она поддерживает открытые протоколы, такие как SOAP, и инкапсулирует лежащую в основе операционную систему и службы объектной модели. Инфраструктура обеспечивает надежную и безопасную среду исполнения программ для многочисленных языков программирования и предлагает безпасные, интегрированные библиотеки классов.

Инфраструктура 132 представляет собой многоуровневую архитектуру, которая включает в себя уровень 142 интерфейса прикладного программирования (ИПП, API),

уровень 144 единой среды выполнения программ на различных языках (CLR) и уровень 146 операционной системы/служб. Данная многоуровневая архитектура делает возможным производить обновление и модификацию различных уровней без воздействия на другие части инфраструктуры. Общая языковая спецификация (ОЯС, CLS) 140 позволяет разработчикам различных языков писать код, который может обращаться к лежащим в основе функциональным возможностям библиотеки. Спецификация 140 функционирует в качестве контракта между разработчиками языка и разработчиками библиотеки, который может использоваться для того, чтобы содействовать функциональной совместимости языков. Придерживаясь ОЯС, библиотеки, написанные на одном языке, могут быть непосредственно доступны кодовым модулям, написанным на других языках, достигая “бесшовной” интеграции между модулями программ, написанными на одном языке, и модулями программ, написанными на другом языке. Одно примерное подробное осуществление ОЯС описывается в стандарте Европейской Ассоциации производителей компьютеров (ЕАПК), созданном участниками технического комитета TC39/TG3 ЕАПК. Авторы предлагают посетить веб-сайт ЕАПК по адресу www.ecma.ch.

Уровень 142 API представляет группы функций, которые прикладные программы 130 могут вызвать для доступа к ресурсам и службам, предоставляемым уровнем 146. Раскрывая функции API для сетевой платформы, разработчики прикладных программ могут создавать веб-прикладные программы для распределенных вычислительных систем, которые обеспечивают полное использование сетевых ресурсов и других веб-служб без необходимости понимания обеспечения межсетевого обмена, как данные сетевые ресурсы фактически работают или делаются доступными. Кроме того, веб-прикладные программы могут быть написаны на любом количестве языков программирования и транслироваться в промежуточный язык, поддерживаемый единой средой выполнения программ 144, и включаться как часть общей языковой спецификации 140. Таким образом, уровень 142 API может предоставить методы для широкого и разнотипного многообразия прикладных программ.

Кроме того, инфраструктура 132 может конфигурироваться для поддержки вызовов API, размещаемых удаленными прикладными программами, исполняющимися удаленно от серверов 134, которые хостируют эту инфраструктуру. Примерные прикладные программы 148 (1) и 148 (2), постоянно находящиеся на клиентах 120 (3) и 120 (M) соответственно, могут использовать функции API, выполняя вызовы прямо, или косвенно, к уровню 142 API по сети 104.

Инфраструктура также может выполняться на клиентах. Клиент 120 (3) представляет ситуацию, когда инфраструктура 150 выполняется на клиенте. Данная инфраструктура может быть идентична инфраструктуре 132 на основе сервера или модифицироваться под потребности клиента. Альтернативно, инфраструктура на основе клиента может концентрироваться в том случае, когда клиент представляет собой устройство с ограниченной или специализированной функцией, такое как сотовый телефон, персональный цифровой помощник, карманный компьютер или другое устройство передачи данных/вычислительное устройство.

Инфраструктура программирования разработчиков

На фиг.2 более подробно представлена инфраструктура 132 программирования. Уровень 140 общей языковой спецификации (CLS) поддерживает прикладные программы, написанные на многочисленных языках 130 (1), 130 (2), 130 (3), 130 (4) … 130 (K). Такие языки прикладных программ включают в себя Visual Basic, C++, C#,

Кобол, Jscript, Perl, Eiffel, Python и т.д. Общая языковая спецификация 140 определяет поднабор особенностей или правил об особенностях, которые, если ими руководствоваться, позволяют устанавливать связь между различными языками. Например, некоторые языки не поддерживают данный тип (например, тип «int*»), который иным образом может поддерживаться единой средой 144 выполнения программ. В этом случае общая языковая спецификация 140 не включает в себя данный тип. С другой стороны, типы, которые поддерживаются всеми или большинством языков (например, тип «int[]»), включаются в общую языковую спецификацию 140, так что разработчики библиотек свободно используют его, и гарантируется, что языки смогут его обработать. Данная возможность связи приводит к “бесшовной” интеграции между модулями программ, написанными на одном языке, и модулями программ, написанными на другом языке. Так как различные языки особенно хорошо подходят к конкретным задачам, “бесшовная” интеграция между языками позволяет разработчику выбрать определенный язык для конкретного модуля программы с возможностью использовать данный модуль программы с модулями, написанными на других языках. Единая среда 144 выполнения программ позволяет получить “бесшовную” многоязыковую разработку, с межъязыковым наследованием, и обеспечивает надежную и безопасную среду исполнения программ для многочисленных языков программирования. Для дополнительной информации по общей языковой спецификации 140 и единой среде 144 выполнения программ внимание обращается на совместно рассматриваемые заявки, названные «Method and System for Compiling Multiple Languages», поданную 6/21/2000 (номер 09/598105), и «Unified Data Type System and Method», поданную 7/10/2000 (номер 09/613289), которые включаются в качестве ссылки.

Инфраструктура 132 инкапсулирует операционную систему 146(1) (например, операционные системы под торговой маркой Windows®) и службы 146(2) объектной модели (например, компонентная объектная модель (КОМ) или распределенная КОМ). Операционная система 146(1) обеспечивает обычные функции, такие как управление файлами, уведомление, обработка событий, пользовательские интерфейсы (например, управление окнами, меню, диалоги и т.д.), безопасность, аутентификация, верификация, процессы и потоки, управление памятью и т.д. Службы 146(2) объектной модели обеспечивают сопряжение с другими объектами для выполнения различных задач. Вызовы, сделанные на уровень 142 ИПП (API), передаются на уровень 144 общеязыковой исполняющей среды для локального исполнения операционной системой 146(1) и/или службами 146(2) объектной модели.

ИПП 142 группирует функции ИПП в многочисленные пространства имен. Пространства имен, по существу, определяют коллекцию классов, интерфейсов, делегатов, перечислений и структур, которые вместе называются «типами», которые обеспечивают специфический набор относящихся функциональных возможностей. Класс представляет данные с распределением в управляемой куче, которые имеют семантику присваивания ссылки. Делегат представляет собой объектно-ориентированный указатель на функцию. Перечисление представляет собой особый вид размерного типа, который представляет именованные константы. Структура представляет данные со статическим распределением, которые имеют семантику присваивания значения. Интерфейс определяет контракт, который другие типы могут выполнять.

Посредством использования пространств имен разработчик может организовать набор типов в иерархическое пространство имен. Разработчик может создать

многочисленные группы из набора типов, причем каждая группа содержит по меньшей мере один тип, который раскрывает (имеет) логически связанные функциональные возможности. В примерном осуществлении ИПП 142 организуется для включения трех корневых пространств имен. Необходимо отметить, что, хотя только три корневые пространства имен изображены на фиг.2, дополнительные корневые пространства имен также могут быть включены в ИПП 142. Тремя корневыми пространствами имен, изображенными в ИПП 142, являются: первое пространство 200 имен для подсистемы представления (которое включает в себя пространство 202 имен для оболочки пользовательского интерфейса), второе пространство 204 имен для веб-служб и третье пространство 206 имен для файловой системы. Каждой группе затем может быть присвоено имя. Например, типам в пространстве 200 имен подсистемы представления может быть присвоено имя «Windows», и типам в пространстве 206 имен файловой системы могут быть присвоены имена «Storage». Именованные группы могут быть организованы под единственным «глобальным корневым» пространством имен для ИПП системного уровня, таким как общее пространство имен System. Посредством выбора и присоединения идентификатора верхнего уровня ссылку на типы в каждой группе легко можно выполнить посредством иерархического имени, которое включает в себя выбранный идентификатор верхнего уровня, который присоединен к имени группы, содержащей тип. Например, на типы в пространстве 206 имен файловой системы можно ссылаться с использованием иерархического имени «System.Storage». Таким образом, индивидуальные пространства 200, 204 и 206 имен становятся главными ответвлениями пространства имен System и могут иметь обозначение, в которых к индивидуальным пространствам имен присоединяются указатель, такой как префикс «System.».

Пространство 200 имен подсистемы представления относится к программированию и разработке контента. Оно предоставляет типы, которые позволяют создавать прикладные программы, документы, медиа-представления и другой контент. Например, пространство 200 имен подсистемы представления обеспечивает модель программирования, которая позволяет разработчикам получать услуги из операционной системы 146 (1) и/или службы 146 (2) объектной модели.

Пространство 202 имен оболочки имеет отношение к функциональным возможностям пользовательского интерфейса. Оно предоставляет типы, которые позволяют разработчикам внедрять функциональные возможности пользовательского интерфейса в их прикладные программы и дополнительно позволяет разработчикам расширять функциональные возможности пользовательского интерфейса.

Пространство 204 имен веб-служб относится к инфраструктуре, позволяющей создавать разнообразные прикладные программы, например, прикладные программы такие же простые, как прикладные программы диалогового взаимодействия пользователей, которые работают между двумя одноранговыми узлами в интрасети, и/или такие же сложные, как масштабируемая веб-служба для миллионов пользователей. Описанная инфраструктура выгодно является сильно изменяемой в том, что необходимо использовать только те части, которые соответствуют сложности конкретного решеия. Инфраструктура обеспечивает основу для создания основанных на сообщениях прикладных программ различного масштаба и сложности. Инфраструктура или структура обеспечивает ИПП для базового обмена сообщениями, безопасного обмена сообщениями, надежного обмена сообщениями и

обмена сообщениями транзакций. В описанном ниже варианте выполнения ассоциированные ИПП были разложены на иерархию пространств имен таким образом, который был тщательно создан для согласования полезности, используемости, расширяемости и управляемости версиями.

Пространство 206 имен файловой системы относится к хранению. Оно предоставляет типы, которые делают возможным выполнять хранение и извлечение информации.

В дополнении в инфраструктуре 132 инструментальные средства 210 программирования предусмотрены для того, чтобы содействовать разработчику при построении веб-служб и/или веб-приложений. Одним из примеров инструментальных средств 210 программирования является Visual Studio™, многоязыковый набор инструментальных средств программирования, предлагаемый корпорацией Microsoft.

Корневые пространства имен ИПП

На фиг.3 подробно изображено пространство 200 имен подсистемы представления. В одном варианте выполнения пространства имен идентифицируются в соответствии с соглашением об иерархических именах, в котором строки имен соединяются точками. Например, пространство 200 имен подсистемы представления идентифицируется корневым именем «System.Windows». Внутри пространства имен «System.Windows» существует другое пространство имен для оболочки, определенное как «System.Windows.Explorer», которое дополнительно идентифицирует другое пространство имен для элементов управления, известных как «System.Windows.Explorer.Controls». С учетом данного соглашения об именах нижеследующее предоставляет общий обзор пространства 200 имен подсистемы представления, хотя с одинаковым эффектом могут использоваться другие соглашения об именах.

Пространство 202 имен Shell («System.Windows.Explorer») включает в себя классы и ИПП (API), которые поддерживают функциональные возможности пользовательского интерфейса, позволяющие конечным пользователям выполнять исследование и навигацию к набору конечных точек, таких как элементы в системах хранения, файлы в местоположениях протокола передачи файлов (ППФ, FTP) или местоположениях распределенного авторинга и контроля версий (РАКВ, DAV), апплеты панели управления в панелях управления, прикладные программы и т.д. Данная навигация может включать в себя открытие одной или нескольких папок, а также открытие папок внутри папок. Пространство 202 имен Shell позволяет разработчикам внедрять такие функциональные возможности в свои прикладные программы и дополнительно позволяет разработчикам расширять эти функциональные возможности исследования и навигации. В версиях операционной системы Windows® данный пользовательский интерфейс обычно упоминается как «Explorer» (Проводник).

В некоторых вариантах выполнения система хранения, у которой функциональные возможности пользовательского интерфейса пространства 202 имен оболочки позволяют пользователям выполнять исследование и навигацию, представляет собой активную платформу хранения для организации, поиска и совместного использования всех видов информации. Данная платформа определяет модель данных с широкими возможностями, надстраивает верхнюю часть реляционного механизма хранения, поддерживает гибкую модель программирования и обеспечивает набор служб передачи данных для контроля, управления и манипулирования данными. Данные могут быть основаны на файлах или быть нефайловыми данными, и данные обычно

упоминаются как «элемент». Файловая система расширяет функциональные возможности, обычно предусматриваемые файловыми системами, так как она также имеет дело с элементами, которые представляют собой нефайловые данные, такие как персональные контакты, календари событий и сообщения электронной почты. Одним примером такой файловой системы является файловая система «WinFS».

Универсальное хранилище данных системы хранения, у которого функциональные возможности пользовательского интерфейса пространства 202 имен оболочки позволяют пользователям выполнять исследование и навигацию, реализует модель данных, которая поддерживает организацию, поиск, совместное использование, синхронизацию и безопасность данных, которые постоянно находятся в хранилище. Основная единица информации хранения в данном хранилище данных упоминается как “элемент”. Модель данных обеспечивает механизм для объявления (описания) элементов и расширений элементов, для установления зависимостей между элементами и для организации элементов в папки и категории.

Элемент представляет собой единицу хранимой информации, которая, в отличие от простого файла, представляет собой объект, имеющий базовый набор свойств, которые, как правило, поддерживаются по всем объектам, открытым системой хранения пользователю или прикладной программе. Элементы также имеют свойства и зависимости, которые, как правило, поддерживаются по всем типам элементов, включая признаки, которые позволяют вводить новые свойства и зависимости. Эти данные свойств и зависимостей также могут упоминаться как метаданные, ассоциированные с элементом. Как подробно описано ниже, метаданные могут храниться в соответствии со схемой декорирования (оформления) элемента. Данная схема декорирования элемента может указывать соответствующий способ, которым следует представлять элемент пользователю.

Элементы представляют собой объекты для общих операций, таких как копирование, удаление, перемещение, открытие, печать, резервирование, восстановление, дублирование и т.д. Элементы представляют собой единицы, которые могут храниться и извлекаться, и все формы хранимой информации, манипулируемой платформой хранения, существуют в виде элементов, свойств элементов или зависимостей (соотношений) между элементами, каждая из которых подробно описывается в данной заявке ниже. Элементы предназначены для представления реальных и легко понимаемых единиц данных, таких как Contacts (контакты), People (люди), Services (службы), Locations (местоположения), Documents (документы) (всех различных видов) и т.д.

Элементы представляют собой отдельные (автономные) объекты; таким образом, если элемент удаляется, то также удаляются все свойства элемента. Аналогично, при извлечении элемента, тем, что принимают, является элемент и все его свойства, содержащиеся в метаданных элемента. Некоторые варианты выполнения могут делать возможным выполнение запроса поднабора свойств при извлечении определенного элемента; однако, для многих таких вариантов выполнения по умолчанию предоставляется элемент со всеми его непосредственными и наследованными свойствами при извлечении. Кроме того, свойства элементов также могут быть расширены добавлением новых свойств к существующим свойствам данного типа элемента. Эти «расширения» представляют собой, соответственно, истинные свойства элемента, и подтипы данного типа элемента могут автоматически включать в себя свойства расширения. Расширения также могут упоминаться как метаданные, ассоциированные с файлом.

Группы элементов могут быть организованы в специальные элементы, называемые Папки элементов (которые не следует путать с папками файлов). В отличие от большинства файловых систем, однако, элемент может принадлежать к более чем одной Папке элементов, так что, когда к элементу производится доступ в одной Папке элементов и он исправляется, то к данному исправленному элементу затем можно произвести доступ непосредственно из другой Папки элементов. По существу, хотя доступ к элементу может происходить из разных Папок элементов, тем, к чему производится доступ, является, фактически, один и тот же элемент. Однако Папка элементов необязательно имеет все свои элементы членов или может просто совместно владеть элементами во взаимодействии с другими Папками элементов, так что удаление Папки элементов необязательно приводит к удалению элемента.

Группы элементов также могут быть организованы в Категории. Категории концептуально отличаются от Папок элементов тем, что, хотя Папки элементов могут содержать элементы, которые не являются взаимосвязанными (т.е. без общей описанной характеристики), каждый элемент в Категории имеет общий тип, свойство или значение («общность»), который описывается для данной Категории, и именно данная общность образует базу для ее взаимосвязи с другими элементами и среди других элементов в Категории. Кроме того, хотя членство элементов в конкретной Папке не является обязательным, на основании любого конкретного аспекта данного элемента, для некоторых вариантов осуществления все элементы, имеющие общность, безусловно относящуюся к Категории, могут автоматически становиться членом Категории на системном уровне интерфейса аппаратных средств/программного обеспечения. Концептуально Категории также могут рассматриваться в качестве виртуальных Папок элементов, членство которых основывается на результатах конкретного запроса (такого как в контексте базы данных), и элементы, которые удовлетворяют условиям данного запроса (определенного общностью Категории), составляют, таким образом, членство в Категории.

В противоположность файлам, папкам и каталогам элементы, Папки элементов и Категории настоящего изобретения не являются типично «физическими» по характеру, так как они не имеют концептуальных эквивалентов физическим контейнерам (накопителям), и поэтому элементы могут существовать в более чем одном таком местоположении. Возможность для элементов существовать в более чем одном местоположении Папки элементов, а также организовываться в Категории, обеспечивает улучшенную и усовершенствованную степень манипулирования данными и возможности структуры хранения на уровне интерфейса аппаратных средств/программного обеспечения свыше той, которая доступна в технике в настоящее время.

Элементы также могут содержать реляционную информацию, которая позволяет определять зависимости (отношения) между двумя или более элементами. Зависимости (отношения) являются двоичными зависимостями, где один элемент обозначается как исходный, а другой элемент как целевой. Исходный элемент и целевой элемент связаны такой зависимостью.

Элементы в универсальном хранилище данных представляются пользователю браузером оболочки, который обеспечивает пользовательский интерфейс (используя классы и ИПП пространства 202 имен Shell), позволяющим пользователю просматривать и взаимодействовать с интерфейсом аппаратных средств/программного обеспечения. Каждый элемент в универсальном хранилище данных хранится в соответствии с универсальной схемой данных. Данная схема

включает в себя механизм для описания элементов, названный присваивание типа. Каждое присваивание типа имеет базовое представление в оболочке; посредством хранения элемента в соответствии с присваиванием типа оболочка может отображать элемент в соответствии по меньшей мере с базовым представлением (видом) отображения или представлением (видом) отображения по умолчанию.

Присваивание типа представляет собой свойство, ассоциированное с элементом; при размещении данных в универсальном хранилище данных одно или несколько свойств, ассоциированных с данными, должно объявляться для определения, каким типом элемента оно является. Данные свойства могут быть включены в качестве метаданных, ассоциированных с данными. Оболочка имеет набор присваиваний типа по умолчанию, которые представляют наиболее базовые и минимальные свойства, которые должны объявляться для элемента.

Кроме объявлений свойств метаданные, ассоциированные с элементом, могут включать в себя данные, указывающие на то, как оболочка должна декорировать представление элемента. Декорирования, в данном случае, могут рассматриваться как «рекомендации» относительно того, как представлять элемент пользователю. Эти метаданные могут храниться в соответствии со схемой декорирования элемента. Схема декорирования элемента определяет вид декорирования элемента, который оболочка может использовать для представления элемента. Например, данные декорирования элемента могут описывать наиболее важные объявленные свойства для элемента. Эти «высокозначимые» свойства могут быть наиболее желательными для представления в оболочке.

Для представления элемента данные декорирования элемента могут указывать набор полей вида, подходящих для представления элемента. Поля вида представляют собой проекции объявленных свойств, и общие поля видов могут включать в себя «название», «автора», «дату создания» или «последнюю редакцию». Оболочка включает в себя набор стандартных полей видов, и независимые поставщики программного обеспечения (НППО, ISV) могут определять поля видов, которые подходят для представления их данных. Во время разработки новых типов элементов НППО могут или отобразить свойства элемента, которые они определяют, на поля видов оболочки, или они могут предоставить свои собственные поля видов.

Например, конкретные данные элемента могут содержать данные о песне. Набор объявленных свойств может включать в себя свойства, такие как название песни, певца, дату записи, альбом, продолжительность песни и другие объявления, соответствующие такому элементу песни. Данные декорирования элемента могут указывать, что поля видов «Название», «Певец» и «Альбом» должны отображаться пользователю во время представления элемента в оболочке.

Декорирования элемента могут быть любым аспектом отображения, поддерживаемым оболочкой. Некоторыми другими общими декорированиями элемента являются, например, форматирование данных, порядок сортировки по умолчанию и размер пиктограмм по умолчанию. Кроме того, данные декорирования элемента могут описывать общие элементы управления для использования при отображении данного элемента. Например, поле Оценки может использовать элементы управления оценок, которые представляют оценки в виде последовательности звездочек. Данные декорирования элемента могут описывать задачи или команды, подходящие для использования с элементом. Термины «задача» и «команда» описывают некоторое действие, которое должно быть предпринято в отношении элемента, и такие термины могут использоваться попеременно. Например,

«Редактирование» или «Предварительный просмотр» могут быть соответствующими задачами/командами, связанными с элементом. Оболочка может быть дополнительно сконфигурирована для запуска прикладных программ для поддержки данных задач при выборе пользователем выполнения действия в отношении элемента.

Вид декорирования элемента является достаточным для полного представления данного элемента или однородного набора элементов, состоящего из элементов, имеющих аналогичные виды декорирования элемента. Для отображения элементов с различными схемами декорирования элемента оболочка обеспечивает схемы видов оболочки, которые представляют элементы в соответствии с видами декорирования оболочки. Схема видов оболочки позволяет оболочке или НППО объявить соответствующие виды для данной совокупности неоднородных данных.

Элементы, выбранные для представления внутри вида декорирования оболочки, могут включать в себя общие характеристики. Большое разнообразие общих характеристик может быть допустимым для вида декорирования оболочки. Например, схема видов оболочки может определять вид «Изображение», используемый для отображения общих и соотвествующих полей и метаданных для всех известных типов изображения (например, .GIF, .JPEG, .BMP, .TIFF и т.д.). Схема видов оболочки подменяет конфликтующие атрибуты отображения для данного вида декорирования элемента и представляет каждый элемент изображения в соответствии со схемой видов оболочки. В качестве другого примера, оболочка может обеспечивать вид оболочки «Документ», который является оптимизированным по соответствующим столбцам и метаданным для элементов, создаваемых обычными рабочими прикладными программами, такими как электронная обработка текста, электронные таблицы или базы данных, даже если декорирования элемента для каждого из этих элементов могут сильно отличаться друг от друга. Такой вид имеет значение посредством обеспечения общих свойств среди каждого из этих документов. Если установлены более поздние типы документов, то вид оболочки сможет представить эти новые элементы в соответствии с совместимым видом оболочки, даже если новый тип не рассматривался в то время, когда ранее создавался вид.

Схемы видов оболочки могут обеспечивать многочисленные атрибуты отображения, и НППО могут пожелать предоставить такие виды оболочки. Атрибуты отображения могут включать в себя без ограничения: размер области предварительного просмотра, метаданные для отображения внутри области предварительного просмотра, подлежащие использованию специализированные элементы управления, а также задачи и команды, подходящие для представленных элементов.

Оболочка может представлять элементы в соответствии с видом отображения проводника. «Проводник» может упоминаться в качестве прикладной программы хранения и может предусматриваться оболочкой или НППО. Проводник, например, может предоставить пользователю возможность просматривать, запрашивать, проводить навигацию, запускать задачи или организовывать выбранные элементы в хранилище данных. Термин «проводник» не должен означать местоположение, где постоянно находятся отображаемые элементы, и термины, такие как «центр активности», «программа просмотра» и «библиотека», могут использоваться взаимозаменяемо с «проводником» для описания прикладной программы хранения.

Примерная иерархия схемы проводника включает в себя нижний уровень, который представляет собой схему видов элемента. Схема видов элемента обеспечивает базовое отображение, необходимое для представления элемента, и схема видов

проводника может полагаться или обращаться к своим элементам отображения, когда это потребуется.

Схема видов проводника также включает в себя схему видов оболочки и декорирования проводника. Декорирования проводника декорируют проводник в целом и обеспечивают элементы отображения, такие как отличительные цвета и маркирующие элементы. Эти декорирования проводника сохраняются среди различных видов, которые обеспечивает проводник. Многочисленные атрибуты отображения могут подходить для декорирования проводника. Например, запросы данных или задачи/команды, ассоциированные с элементами проводника, могут назначаться для отображения проводником. Отображаемые задачи часто связаны с прикладной программой, способной выполнять задачу.

Схема видов проводника может дополнительно включать в себя схему видов оболочки или многочисленные схемы видов оболочки. Схема видов оболочки может конфигурироваться для предоставления вида оболочки для поднабора элементов проводника. Например, проводник может конфигурироваться для отображения пользователю элементов песни. Первая схема видов оболочки может включаться для обеспечения отображения альбомов, и вторая схема видов оболочки может включаться для обеспечения отображения дорожек песни. Таким образом, оба типа элементов будут иметь соответствующие виды в проводнике. Как описано выше, использование вида оболочки относится к представлению набора элементов, которые, дополнительно, могут совместно использовать общие характеристики.

Проводник также может зависеть от видов (полагаться на виды) оболочки, включенных в оболочку. Если элементы, выбранные для представления внутри проводника, не поддерживаются никаким видом оболочки, включенным проводником, то оболочка может предоставить соответствующий вид оболочки для использования в проводнике. Аналогично и как описано выше, проводник также может возвращаться к виду отображения элемента или виду отображения по умолчанию, предусмотренному оболочкой. Данная функциональная возможность гарантирует то, что любой элемент, который может быть отображен оболочкой, также может отображаться в проводнике. Проводник может конфигурироваться так, чтобы полагаться на эти предусмотренные оболочкой схемы отображения или может зависеть от них, например, для предоставления отображения для непредвиденных данных.

Пространство 202 имен Shell поддерживает функциональные возможности пользовательского интерфейса, которые позволяют производить манипулирование и взаимодействие элементов совместимым образом независимо от местоположения, где хранятся лежащие в основе данные для элемента (например, содержимое файла, данные изображения, данные о контактной информации и т.д.). С точки зрения разработчика прикладных программ, функциональные возможности, поддерживаемые пространством 202 имен Shell, приводят к лучшему опыту разработки по меньшей мере частично, так как такие элементы могут рассматриваться совместимым образом несмотря на то, что лежащие в основе данные хранятся в различных местоположениях. Разработчик приобретает последовательный опыт при использовании пространства 202 имен Shell, несмотря на потенциальное присутствие таких различных местоположений хранения данных.

Пространство 202 имен Shell определяет дополнительные пространства имен, включая пространство 302 имен Controls, пространство 304 имен Dialogs, пространство 306 имен Addin и пространство 308 имен Desktop. Каждое из этих

дополнительных пространств 302, 304, 306 и 308 имен в пространстве имен Shell включает в себя одну или несколько служб или компонентов, которые обеспечивают различные функциональные возможности, такие как элементы управления, диалоги и/или обработчики, как подробно описано ниже.

Пространство 202 имен Shell определяет объектную модель, которая может использоваться каждым дополнительным пространством 302, 304, 306 и 308 имен. Данная объектная модель включает в себя следующие объекты:

- Объект ExplorerItem (также упоминаемый как объект Item), используемый для описания элементов, которые могут быть представлены пользователю посредством Shell, такие как папки файлов, стеки, индивидуальные файлы, индивидуальные элементы WinFS (например, альбомы, песни, изображения и т.д.) и т.п.

- Объект Libraries, используемый для описания запроса в отношении ExplorerItem.

- Объект ViewFields (также упоминаемый как объект Properties), используемый для проецирования данных из элементов (например, представляемых объектами EplorerItem) для отображения пользователю. Проекция (отображение), определяемая между свойствами в системе хранения и объектом ViewFields, может выполнять дополнительные вычисления или обработку, основываясь на ExplorerItems. Например, если папка, представляющая устройство хранения, отображается (на экране), и разработчик желает включить в данную папку некоторое количество свободного пространства на данном устройстве хранения, то тогда проекция вычисляет величину свободного пространства, основываясь на общей емкости устройства хранения и размерах файлов (представленных как ExplorerItem), хранимых на устройстве хранения.

- Объект StorageFavorites, используемый для описания связи с динамическим списком, который представляет собой постоянную форму объекта Library. Объект StorageFavorites представляет собой комбинацию запроса и вида для представления результатов запроса пользователю.

Данная объектная модель подробно описывается в заявке на патент США №_____, реестр поверенного № MFCP.109834 (MS# 306923.01), поданной 10/23/03, названной «System and method for presenting items to a user with a contextual presentation», которая этим включается посредством ссылки.

Пространство 302 имен Controls («System.Windows.Explorer.Controls») определяет коллекцию повторно используемых элементов управления пользовательского интерфейса, которые могут использоваться прикладными программами. Так как многочисленные различные прикладные программы могут использовать данные элементы управления, то данные элементы управления могут рассматриваться как повторно используемые. Данная коллекция повторно используемых элементов управления может упростить разработку прикладной программы, так как разработчики прикладной программы могут полагаться на данные элементы управления, а не быть вынужденными разрабатывать свои собственные версии элементов управления. Кроме того, посредством использования данной коллекции повторно используемых элементов управления может быть достигнут совместимый «визуальный стиль» для многочисленных прикладных программ.

Элементы управления, определяемые пространством 302 имен Controls, позволяют производить манипулирование и/или отображение оболочек пользовательского интерфейса, включая манипулирование и/или отображение элементов в пользовательском интерфейсе. Как описано выше, эти элементы, которые могут

отображаться внутри оболочек пользовательского интерфейса, представляют любой из многочисленных видов данных и не являются просто традиционными «файлами», хранимыми в файловой системе. Кроме того, элементы управления, определяемые пространством 302 имен Controls, позволяют разработчикам прикладных программ использовать эти же элементы управления при работе с данными из разнообразных местоположений, приводя к совместимому и последовательному опыту разработки.

Пространство 302 имен Controls определяет следующие повторно используемые элементы управления:

- Элемент управления ExplorerView используется для инкапсуляции опыта пользователя в хранении. Опыт пользователя в хранении определяется для работы с элементами с зависимостями (отношениями), запросами и т.п. Данный опыт пользователя в хранении позволяет конечным пользователям, например, выдавать запросы, основываясь на значениях свойств, помещать в стек элементы WinFS с общими значениями свойств, позволяет перетащить и оставить из нестековых элементов в стеки (например, посредством установки значения свойства в предикате свойства равным тому, что существует в стеке) и т.д. Элемент управления ExplorerView также может использоваться для описания папки для отображения, имеющей такой же вид, который можно найти в предыдущих операционных системах (также упоминаемых как унаследованные системы), такие как предыдущие версии операционной системы Microsoft® Windows® (например, версии Windows® XP или Windows® 98 операционной системы Windows®).

- Элемент управления ExplorerItemView, используемый для отображения ExplorerItems. Компоновка (размещение) и конструктивное исполнение ExplorerItems могут быть определены разработчиком прикладной программы. Операционная система также может определять опыт пользователя, который накладывает некоторые ограничения на компоновку и конструктивное исполнение, которые могут определяться разработчиком прикладной программы, так что компоновка и конструктивное исполнение, определенные разработчиком прикладной программы, совместимы с опытом пользователя, определенным операционной системой. Таким образом, элемент управления ExplorerItemView позволяет разработчику определить, как данные из ExplorerItems должны отображаться пользователю (например, в столбцах, строках или в некоторой другой конфигурации), а также какие данные должны отображаться пользователю.

- Элемент управления BasketControl, используемый для того, чтобы можно было добавлять элементы к боковому полю (боковые поля более подробно описываются ниже). Элемент может быть добавлен к боковому полю посредством «перетаскивания и оставления», используя устройство управления курсором для выбора элемента и перетаскивания его на боковое поле, где он может быть оставлен (например, посредством отпускания кнопки управления курсором). Элемент управления BasketControl подробно описывается в заявке на патент США №_____, реестр поверенного № 003797.00695 (MS# 304631.1), поданной 10/13/03, названной «Extensible Creation And Editing Of Integrated Collections», которая этим включается посредством ссылки.

- Элемент управления ListMaker, используемый для того, чтобы можно было добавлять элементы в список или группу. Списки или группы элементов могут создаваться посредством «перетаскивания и оставления» с использованием устройства управления курсором для выбора элемента и перетаскивания его на соответствующий список или группу, где он может быть оставлен (например, посредством отпускания

кнопки управления курсором). Например, элементами могут быть песни и списком может быть плейлист (список воспроизведения), элементами могут быть изображения и списком может быть фотоальбом и т. д.

- Компонент PreviewImageControl, используемый для того, чтобы можно было представлять пользователю изображения предварительного просмотра элементов. Когда элемент выбран, то изображение предварительного просмотра может быть представлено пользователю, причем изображение предварительного просмотра обычно больше «миниатюры» для изображения, но меньше полной области отображения пользовательского интерфейса.

Пространство 304 имен Dialogs («System.Windows.Explorer.Dialogs») определяет коллекцию диалогов и “мастеров” (программ-разработчиков) пользовательского интерфейса, которые могут использоваться прикладной программой. Аналогично элементам управления в пространстве 302 имен Controls диалоги и “мастера” в пространстве 304 имен Dialogs могут повторно использоваться, так что многочисленные различные прикладные программы могут воспользоваться данными диалогами и “мастерами”. Данная коллекция повторно используемых диалогов и “мастеров” способствует сохранению совместимого «визуального стиля» в многочисленных прикладных программах.

Диалоги и “мастера”, определяемые пространством 304 имен Dialogs, позволяют выполнять действия внутри оболочки пользовательского интерфейса. Как описано выше, данные оболочки могут быть основаны на элементах, которые могут представлять любые из данных многочисленных видов, а не просто традиционные «файлы», хранимые в файловой системе. Кроме того, диалоги и “мастера”, определяемые пространством 304 имен Dialogs, позволяют разработчикам прикладных программ использовать одни и те же диалоги и “мастера” при работе с данными из многочисленных местоположений, приводя к совместимому и последовательному опыту разработки.

Пространство 304 имен Dialogs определяет следующие диалоги и “мастера”:

- диалоги Open/Save, используемые для того, чтобы можно было открывать и сохранять файлы и папки. Данные диалоги также являются расширяемыми, позволяя разработчикам прикладных программ расширять функциональные возможности диалогов. Например, данные диалоги могут быть расширены для использования различных дополнительных метаданных, касающихся файлов и/или папок, которые разработчик прикладных программ желает использовать.

- “Мастера” CD/DVD Burn, используемые для выполнения записи на оптические диски, такие как компакт-диски и/или цифровые многофункциональные диски (DVD). Разработчики аппаратных средств могут предоставить свои собственные мастера, включающие свои собственные характерные для аппаратных средств этапы, которые позволяют производить запись на компакт-диски и/или DVD.

- Мастер Send Photos, используемый для того, чтобы помогать пользователям в пересылке изображений по электронной почте. “Мастер” позволяет пользователю именить размер изображения до конкретного размера. “Мастер” может изменить размер изображения до конкретного размера, считающегося пригодным для отправки по электронной почте, или, альтернативно, может позволить пользователю произвести выбор из двух или более различных размеров.

Пространство 306 имен Addin («System.Windows.Explorer.Addin») определяет коллекцию базовых классов и интерфейсов для расширения функциональных возможностей пользовательского интерфейса. Пространство 306 имен Addin позволяет

разработчикам прикладных программ добавлять управляемый код, который расширяет функциональные возможности пользовательского интерфейса. Данное расширение позволяет, например, разработчикам прикладных программ настраивать части пользовательского интерфейса так, как они считают подходящим.

Пространство 306 имен Addin предусматривает расширение оболочек пользовательского интерфейса, которые используются для отображения элементов, которые представляют любые данные разнообразных видов, а не просто традиционные «файлы», хранимые в файловой системе. Кроме того, пространство 306 имен Addin позволяет разработчикам прикладных программ расширять оболочки пользовательского интерфейса последовательным образом, обеспечивая более управляемый и улучшенный опыт разработки.

Управляемый код для различных функциональных возможностей поддерживается пространством 306 имен Addin, например:

- Обработчики контекстного меню. К контекстному меню можно получить доступ, например, щелчком по правой кнопке мыши на отображаемом элементе. Обработчики контекстного меню позволяют разработчикам прикладных программ определять свои собственные позиции, добавляемые к этим контекстным меню.

- Выделители «миниатюр». Выделители «миниатюр» позволяют разработчикам прикладных программ разрабатывать свои собственные «миниатюры» (например, для файлов, папок и/или других отображаемых элементов) и идентифицировать эти «миниатюры» в операционной системе для отображения пользователю.

- Обработчики для составления/разбиения проекций из WinFS в Explorer. Данные обработчики позволяют разработчикам прикладных программ выполнять различные вычисления при отображении пользователю информации, касающейся файлов и/или папок (например, позволяют вычислять величину свободного пространства на устройстве хранения).

Пространство 308 имен Desktop («System.Windows.Explorer.Desktop») определяет коллекцию функциональных возможностей, которые позволяют разработчикам прикладных программ расширять функциональные возможности рабочего стола, отображаемого пользователям. Пространство имен Desktop может быть частью пространства 202 имен Shell (например, «System.Windows.Explorer.Desktop») или, альтернативно, может быть частью пространства 200 имен подсистемы представления, но не частью пространства 202 имен Shell (например, «System.Windows.Desktop»). Пространство имен Desktop включает в себя элементы, входящие в боковое поле и уведомления. Включение элементов бокового поля и уведомления в пространство имен Desktop предоставляет пользователю централизованное управление для бокового поля и уведомлений. Например, правила, касающиеся уведомлений от всех прикладных программ, могут устанавливаться пользователем один раз, не требуя от пользователя проведения навигации по определениям (описаниям) правил доступа для каждой прикладной программы, которая использует уведомления.

В общем случае боковое поле обеспечивает динамический доступ к передаче данных и информационной осведомленности в интегрированном интерактивном периферийном отображении осведомленности, внутри которого определенные контакты передачи данных и информационные элементы динамически отслеживаются или принимаются и предоставляются пользователю на текущей основе. В некоторых вариантах выполнения данная возможность предусматривается при помощи по меньшей мере одной настраиваемой динамической «миниатюры», отображаемой в

одном или нескольких столбцах в постоянной полоске отображения вдоль одной кромки обычного устройства отображения. Далее, в дополнительных вариантах выполнения «миниатюры» отображаются на любой одной части или частях отображения, включая все отображение. Вариант выполнения, в котором охватывается все отображение, особенно полезно там, где боковое поле будет использоваться на устройстве, имеющем относительно небольшую область отображения, таком как, например, ручное или карманное вычислительное устройство, сотовый телефон или любое другое электронное устройство, имеющее ограниченную область отображения.

Каждая из настраиваемых динамических «миниатюр» представляет, например, конкретные контакты передачи данных (такие как конкретные лица, предприятия, организации или другие субъекты) или конкретные элементы информации, в которых пользователь может быть заинтересован. Такие элементы информации включают в себя случаи, например, когда модифицируются совместно используемые файлы или папки, когда изменяется информация в совместно используемой базе данных или рабочем пространстве, состояние электронной почты, календари, веб-страницы Интернет, погодные условия, встречи, планы, статистическая информация, биржевые сводки, информация о движении или любая другая доступная из Интернет или сети информация, которая может представлять интерес для пользователя.

Вышеупомянутые динамические «миниатюры», в основном, содержат комбинацию «тайла» (элемент мозаичного отображения) (также упоминаемого как «этикетка»), описывающего контакт или представляющую интерес информацию, и специализированной «программы просмотра» для отображения любого контакта передачи данных или информации, которые изображаются тайлом. Каждый тайл и программа просмотра могут, например, являться элементом в системе хранения, к которому можно получить доступ при помощи функциональных возможностей пользовательского интерфейса, предусматриваемых пространством 202 имен оболочки. Одна или несколько «служб» используются для автоматического взаимодействия, отслеживания или приема текущего состояния информации и/или состояния контактов передачи данных, описываемых каждым тайлом. Текущее состояние информации и состояние контактов передачи данных затем динамически предоставляются посредством хостинга каждого тайла в «контейнере», постоянно находящимся в интерактивном периферийном интерфейсе осведомленности для графического и/или текстового отображения элементов. Периферийный интерфейс осведомленности отображает информацию и/или контакты передачи данных так, чтобы уменьшить любое потенциальное отвлечение внимания и препятствие для пользователя.

В общем, тайл представляется структурой данных, такой как файл данных XML. Каждый тайл включает в себя инструкции в отношении того, какая информация или контакт передачи данных должен быть представлен тайлом, а также указатели на конкретные службы, которые представляют любое количество обычных средств для доступа и/или взаимодействия с информацией или контактами передачи данных. Эти службы автоматически или вручную выбираются из предварительно определенной или определяемой пользователем библиотеки служб. В частности, различные службы представляют совместно используемый код или функции, которые обеспечивают функциональные возможности для доступа, приема, извлечения и/или другого взаимодействия с любой обычной информацией, источником информации или контактом передачи данных. Эти службы используются совместно в том смысле, что

они используются или отдельно (автономно), или в комбинации, и могут использоваться одновременно одним или несколькими тайлами. Следовательно, необходимо отметить, что в некоторых вариантах выполнения многочисленные службы (услуги) используются в комбинации для обеспечения комплексного взаимодействия с любой обычной информацией, источником информации или контактом передачи данных.

Одним примером службы (услуги) является функциональная возможность, необходимая для контроля за папкой электронной почты посредством соединения с обычным сервером интерфейса прикладного программирования электронной почты (ИППЭП, MAPI). Другим примером услуги является функциональная возможность посылки или приема сообщений электронной почты. Соответствующие службы обеспечивают функциональную возможность для связи с контактами или передачи информации при помощи любого количества обычных методов, таких как, например, моментальная передача сообщений или схемы передачи данных между равноправными узлами. Другим примером услуги является функциональная возможность преобразования текстового файла с одного языка на другой. Другим примером услуги является функциональная возможность, необходимая для контроля базы данных. Другие примеры услуг включают в себя функциональную возможность приема или извлечения данных из веб-сайта или удаленного сервера. Ясно, что любой способ взаимодействия с любой информацией, источником информации или контактом передачи данных может выполняться в качестве совместно используемой службы (услуги) для использования одним или несколькими тайлами.

Далее, как отмечено выше, инструкции каждого тайла включают в себя указатель на одну из числа специализированных программ просмотра, имеющих возможность отображения любого вида информации или контакта передачи данных, который представляется тайлом. Другими словами, каждый тайл представляет комбинацию информации или контакта, которые пользователь желает отслеживать, вместе с определением того, как пользователь желает просматривать эту конкретную информацию или контакт, а также возможность использования любого количества услуг для доступа и взаимодействия с информацией или контактом. Такой доступ или взаимодействие могут выполняться локально или по локальным интрасетям, экстрасетям, проводным или беспроводным сетям, Интернет и т.д. при помощи любого обычного протокола передачи данных.

Программы просмотра графически отображают тайл в виде «миниатюры» с изменяемым размером или окна с размерами пиктограммы (иконки), имеющего информацию или данные контакта, извлекаемые при помощи одной или нескольких услуг. В частности, программа просмотра выполнена с возможностью динамического отображения тайла, имеющего текстовую, звуковую или графическую информацию, включая неподвижные изображения или изображения при прямой передаче, или любую комбинацию екстовой, звуковой или графической информации. Например, один тип программы просмотра выполнен с возможностью отображения контактной информации, т. е. «персонального тайла», другой тип выполнен с возможностью отображения характерной для электронной почты информации, такой как, например, количество принятых сообщений или количество сообщений от конкретного источника, другая программа просмотра предназначена для взаимодействия с базой данных для предоставления сводки конкретной информации из базы данных в «миниатюре». Другие примеры типов программ просмотра включают в себя программы просмотра, имеющие возможность отображать неподвижные

изображения, видеоизображения, сводку состояния передачи данных, результаты запроса базы данных и т.д. Ясно, что может быть разработан любой тип программы просмотра для ассоциации с любым соответствующим типом информации для обеспечения того, чтобы можно было отобразить любую возможную информацию.

Боковое поле подробно описывается в заявке на патент США №10/063296, названной «A system and process for providing dynamic communication access and information awareness», которая включается в настоящее описание посредством ссылки.

Функциональные возможности бокового поля пространства 308 имен позволяет разработчикам прикладных программ добавлять тайлы к боковому полю.

Уведомления (например, звуковые или визуальные уведомления) составляют часть системы контекста пользователя, которая, в соответствии с некоторыми вариантами выполнения, включает в себя три элемента, которые сравниваются для принятия решения в отношении того, как обрабатывать уведомление. Первым элементом является контекст пользователя (который может обеспечиваться операционной системой и произвольными программами, которые расширили его). Вторым элементом являются правила и предпочтения пользователя. Третьим элементом является само уведомление (которое содержит элементы, такие как данные и свойства, которые могут соответствовать правилам пользователя).

Система контекста пользователя приводится в действие операционной системой и другими программами, объявляющими контексты пользователя, после которых система ведет контекст и правила пользователя. Уведомления устанавливаются другими программами, вызывающими в систему. Контекст, правила пользователя и элементы уведомления сравниваются, и затем принимается решение в отношении того, что необходимо делать с уведомлением. Примеры различных вариантов того, что может быть сделано с уведомлением, включают в себя запрет (например, если уведомлению не разрешается создавать или производить шум, и это уведомление никогда не должно быть показано пользователю), задержку (например, уведомление удерживается до тех пор, пока изменения контекста пользователя или правила пользователя не будут предписывать, что впоследствии его следует доставить), доставку (например, разрешена доставка уведомления в соответствии с контекстом и правилами пользователя) и выбор маршрута (например, правила пользователя указывают, что уведомление должно быть передано другой системе независимо от того, разрешена ли также доставка уведомления в настоящей системе).

В общем, пользователь может находиться в состоянии, считающемся «недоступным», в данном случае уведомление или не доставляется или удерживается до тех пор, пока пользователь не станет «доступным». Например, если пользователь выполняет полноэкранную прикладную программу, то такой пользователь может считаться недоступным. Или пользователь может быть «доступным», но в таком состоянии, что необходимо модифицировать уведомление, чтобы оно было подходящим для пользователя. Например, если пользователь слушает музыку или участвует в совещании, то пользователь может указать, что уведомления должны доставляться на экран пользователя, но звук, который они создают, должен быть или более тихим, или его совсем не должно быть.

Как отмечено выше, контекст пользователя может определять, в частности, показываются ли уведомления на экране пользователя. Если уведомление показывается, то оно может показываться, основываясь на некоторых градиентах внутри контекста пользователя. Другими словами, существуют различные уровни навязчивости формы выводимого уведомления, которые могут быть определены.

Например, нормальное уведомление свободно появляется на клиентской области и кратковременно скрывает окно. Если пользователь находится в слегка ограничительном контексте, уведомление может свободно показываться, но только менее навязчивым образом, например, ему может быть не разрешено выводиться поверх другого окна. В качестве другого примера, в одном варианте выполнения, где пользователь выполняет полноэкранную прикладную программу, установкой по умолчанию может быть такая, которая означает, что контекст слегка ограничен, и что пользователь ясно сделал заявление, что он желает, чтобы данная прикладная программа получила всю клиентскую область. При такой установке уведомлению все же может быть разрешен вывод, но ему может быть разрешено появиться только внутри бокового поля. Другими словми, данный вид пониженной навязчивости в форме вывода уведомления уменьшает воздействие уведомления, и в целом уменьшает познавательную нагрузку.

Системы контекста пользователя подробно описаны в заявке на патент США № 10/402179, поданной 03/26/03, названной «System and Method Utilizing Test Notifications», которая включается в настоящее описание посредством ссылки.

Функциональные возможности уведомлений пространства 308 имен позволяют разработчикам прикладных программ создавать свои собственные уведомления и отображать их на рабочем столе, уведомляя пользователя, когда произошло что-то, что представляет интерес для пользователя (например, было принято сообщение электронной почты, было принято мгновенное сообщение, и т.д.). Функциональные возможности уведомления пространства 308 имен дополнительно обеспечивают центральное местоположение для пользователей, чтобы определять их правила и предпочтения для уведомлений от многочисленных прикладных программ, освобождая пользователей от необходимости многочисленных установок одних и тех же правил и предпочтений (один раз для каждой из многочисленных прикладных программ).

Кроме того, пространство имен Contacts определяет коллекцию элементов управления и диалогов для контактной информации. Эта контактная информация может храниться в качестве элементов в системе хранения. «Контакт», в основном, относится к любому человеку, группе, организации, бизнесу, дому или другому типу идентифицируемого субъекта. «Контактная информация», в основном, относится к любой информации, которая соответствует контакту и которая может рассматриваться как релевантная к идентификации, контакту, доступу, соответствию или связи с контактом. Контактная информация используется прикладной программой для выполнения требуемой функции, такой как, например, посылка электронной почты, инициирование телефонного вызова, доступ к веб-сайту, инициирование игрового сеанса, выполнение финансовой транзакции и т.д. Неограничивающие примеры контактной информации включают в себя имена, псевдонимы, телефонные номера, адреса электронной почты, домашние адреса, адреса моментальной передачи сообщений (МПС) и веб-адреса. Контактная информация также может относиться к другим типам информации, такой как состояние контакта. Например, информация, указывающая, что контакт в настоящее время используется в системе или осуществляется по телефонной линии, также может рассматриваться в широком смысле как контактная информация.

Пространство имен Contacts может быть частью пространства 200 имен подсистемы представления, но не частью пространства 202 имен Shell, такого как «System.Windows.Contacts», «System.Windows.Controls» (элементы 310 управления на

фиг.3) или «System.Windows.Collaboration». Альтернативно, пространство имен Contacts может быть частью пространства 202 имен Shell. Пространство имен Contacts расширяет функциональные возможности пространства имен System.Windows, но не должно выполняться как часть Shell (т. е. не должно быть частью пространства 302 имен System.Windows.Explorer).

Пространство имен Contacts позволяет производить ввод пользователем контактной информации один раз, все же получая доступ посредством многочисленных различных прикладных программ. Пользователь, таким образом, освобождается от необходимости многократного ввода одной и той же контактной информации (один раз для каждой из многочисленных прикладных программ).

Элементы управления и диалоги в пространстве имен Contacts включают в себя:

- Диалог Contact Picker, используемый для того, чтобы позволить пользователям выбирать или отбирать контакт из их хранимых контактов. Например, позволяет пользователям выбрать контакт, по которому будет послано сообщение электронной почты, или контакт для моментальной передачи сообщений. Пользователь может просматривать список, например, всех контактов пользователя, хранимых в качестве элементов в системе хранения, и выбрать один или несколько из данных контактов. Диалог Contact Picker подробно описывается в заявке на патент США № 10/324 746, поданной 12/19/02, названной «Contact Picker», которая включается этим в качестве ссылки.

- Элемент управления Contact Textbox, используемый для того, чтобы дать возможность пользователям ввести контактную информацию, такую как имя, в свободной форме (например, посредством впечатывания имени) и, когда контактная информация будет введена, элемент управления разрешает информацию для одного или нескольких контактов из хранимых контактов пользователя. Разрешенная информация может отображаться, например, в ниспадающем меню или может автоматически вводиться в строку ввода с клавиатуры, где пользователь вводил имя в свободной форме.

- Элемент управления Contact, используемый для отображения контактной информации пользователям. Редактирование или ввод в свободной форме контактной информации не поддерживается данным элементом управления.

Примерные члены пространства имен

Данный раздел включает в себя многочисленные таблицы, описывающие примеры членов, которые могут быть раскрыты примерными пространствами имен (например, пространства имен в пространстве 200 имен подсистемы представления на фиг.3). Эти раскрытые члены могут включать в себя, например, классы, интерфейсы, перечисления и делегаты. Необходимо понять, что члены, описанные в данных примерах, являются только примерами, и, альтернативно, другие члены могут раскрываться пространствами имен.

Необходимо понять, что в некоторых описаниях пространств имен ниже оставлены пустыми описания некоторых классов, интерфейсов, перечислений и делегатов. Более полное описание данных классов, интерфейсов, перечислений и делегатов можно найти в содержании компакт-дисков, на которых хранится вышеупомянутый НРПО.

System.Windows.Explorer.Controls

В нижеследующих таблицах перечисляются примеры членов, раскрываемых пространством имен System.Windows.Explorer.Controls.

Классы

DefaultCommandEventArgs	Содержит данные о событиях для события DefaultCommandEvent.

ExplorerView	Позволяет пользователю просматривать информацию о содержимом папки.
FolderSelectionChangedEventArgs	Содержит данные о событиях для события FolderSelectionChangedEvent.
IncludeItemEventArgs	Содержит данные о событиях для события IncludeItemEvent.
ItemVerbModifyEventArgs	Инициализирует новый экземпляр класса ItemVerbModifyEventArgs.
NavigationCompleteEventArgs	Содержит данные о событиях для события NavigationCompleteEvent.
NavigationFailedEventArgs	Содержит данные о событиях для события NavigationFailedEvent.
NavigationPendingEventArgs	Содержит данные о событиях для события NavigationPendingEvent.

Перечисления

FolderViewFlags	Указывает свойства вида для ExplorerView.
FolderViewMode	Указывает режим вида для ExplorerView.

Делегаты

DefaultCommandEventHandler	Представляет метод, который обрабатывает DefaultCommandEvent.
FolderSelectionChangedEvent Handler	Представляет метод, который обрабатывает FolderSelectionChangedEvent.
IncludeItemEventHandler	Представляет метод, который обрабатывает IncludeItemEvent.
ItemVerbModifyEventHandler	Представляет метод, который обрабатывает ItemVerbModifyEvent.
NavigationCompleteEventHandler	Представляет метод, который обрабатывает NavigationCompleteEvent.
NavigationFailedEventHandler	Представляет метод, который обрабатывает NavigationFailedEvent.
NavigationPendingEventHandler	Представляет метод, который обрабатывает NavigationPendingEvent.

System.Windows.Explorer.Dialogs

Нижеследующие таблицы перечисляют примеры членов, раскрываемых пространством имен System.Windows.Explorer.Dialogs.

Классы

ColorDialog	
CommonDialog	
DialogControlItemSelectedEvent Args	
FileDialog	Абстрактный класс, который используется в качестве родительского класса для OpenFileDialog и SaveFileDialogBase.
FileDialogCheckButton	Представляет элемент управления отмечаемой экранной кнопки, который может быть размещен на FileDialog.
FileDialogComboBox	
FileDialogContainerControlBase	Абстрактный класс, который используется в качестве родительского класса для FileDialogComboBox, FileDialogOpenDropDown, FileDialogRadioButtonGroup и FileDialogToolbarMenu.
FileDialogControlBase	Абстрактный класс, который используется в качестве родительского класса для FileDialogCheckButton, FileDialogEditBox, FileDialogPushButton и FileDialogContainerControlBase.
FileDialogControlItem	
FileDialogEditBox	Представляет элемент управления текстового окна, который может быть размещен на FileDialog.
FileDialogOpenDropDown	
FileDialogPushButton	Представляет элемент управления кнопки, который может быть размещен на FileDialog.

FileDialogRadioButtonGroup	
FileDialogToolbarMenu	
FileOkEventArgs	
FileType	
OpenFileDialog	Позволяет пользователю выбрать один или несколько файлов для открытия. Данный класс не может наследоваться.
SaveAsFileDialog	Позволяет пользователю выбрать местоположение, в котором следует сохранить файл, и указать имя файла. Данный класс не может наследоваться.
SaveFileDialog	Позволяет пользователю выбрать местоположение, в котором следует сохранить файл. Данный класс не может наследоваться.
SaveFileDialogBase	Абстрактный класс, который используется в качестве родительского класса для SaveAsFileDialog и SaveFileDialog.

Перечисления

FileDialogLayout

TileAttributes

Делегаты

DialogControlItemSelectedEventHandler

FileOkEventHandler

System.Windows.Desktop

Нижеследующие таблицы перечисляют примеры членов, раскрываемых пространством имен System.Windows.Desktop. Данное пространство имен содержит элементы, включаемые в боковое поле и уведомления.

Классы

AnalogClockPanel	
Appbar	
AreaButton	
BaseComTile	
BaseSidebarClockSettings	
BaseTile	Абстрактный класс, используемый в качестве родительского класса тайла специального бокового поля
BasketControl	
CalendarElement	
CalendarImages	
ChildrenWontFitArgs	
ClockHacks	
ClockPanel	
DataSourceEventArgs	Представляет данные о событиях, переданные от источника данных
DigitalDateTimeElement	
DragButton	
DragControlWindow	
DraggableButton	
DragWindow	
DropArgs	
DropEventArgs	
ExtraSpaceArgs	
FillAlphaPresenter	
FillImageResourcePresenter	
FillPanel	
FillPresenter	
Flyout	
FlyoutLinkClickEventArgs	Представляет данные о событиях, переданные к RMAActionEventHandler в ответ на событие, которое произошло из необязательной связи, находящейся в нижней части раскрывающегося вида тайла бокового поля.
FlyoutPresenter	
FlyoutStuff	
FocusableButton	
FocusWithinWorkaroundHelper	

FolderContentsChangedEventArgs	Представляет данные о событиях для FolderContentsChangedEventHandler в ответ на изменение в содержимом папки.
GlobalSetting	
HackBorder	
HackImage	
ImageButton	
ImageResource	
ImageResourcePresenter	
ItemControl	
ItemToolbarControl	
ListMakerControl	
MenuStuff	
NativeResource	
NativeResourceHelper	
NativeResources	
NativeResourceTypeConverter	
NormalButton	
Notification	Представляет сообщение и связанные с ним данные, которые система посылает пользователю в части пользовательского интерфейса (ПИ).
NotificationArea	
NotificationButton	Определяет кнопку, которая появляется на уведомлении.
NotificationClickedEventArgs	Содержит данные, связанные с событием щелчка по кнопке мыши в окне уведомления.
NotificationContext	Объявляет, находится ли прикладная программа в настоящее время в конкретном состоянии. Данное состояние используется как часть определения контекста пользователя.
NotificationTimerPresenter	
NotifyCompleteEventArgs	
NotifyWindow	
OptionsButton	
OuterTile	
OverflowableCollection	
OverflowableControlCollection	
OverflowPanel	
OverflowPresenter	
OverflowWrapper	
PanelInfo	
ProgressBar	
QuickLaunchTile	
RMAActionEventArgs	Представляет данные о событии, генерируемые прикладной программой со свернутым окном и широкими возможностями (ППСОШВ, RMA). Используется RMAActionEventHandler.
RMAData	
SetDataArrayEventArg	Представляет данные о событиях, генерируемые массивом данных и посылаемые на SetDataArrayEventHandler.

Sidebar
SidebarAlarmClock
SideBarClock
SidebarClockSettings
SlideShowTile
StackAlphaPresenter
StackImageResourcePresenter
StartButton
SyncHelper
SyncItemBar
SyncTile
Taskbar
TaskChevron
TaskGroup
TaskItem
TaskList
TaskOverflow
TaskOverflowableControlCollection
TaskPresenter
TestTile
TestTimeZone
TextElementFontInfo
Theme
ThemeHelper
ThemeHelperOld
ThemeImage
ThemeImageTypeConverter
ThemeResources
Tile
TileCollection
TileOverflow
TileThumb
TileThumbDotNet
Timer
TimerStuff
TimeZone
ViewStatusControl
WebHostEventArgs
WindowOrigin
WindowOriginOld
WindowStuff

Интерфейсы

ICOMDataSourceHandler	Используется для создания и осуществления связи с источником данных, выполненным в неуправляемом коде.
IOverflow	
IOverflowList	
ISidebar	Обеспечивает доступ к боковому полю, которое хостирует индивидуальные тайлы. Хост бокового поля ответственен за открытие и закрытие раскрывающихся объектов, отображение меню быстрого вызова и за ответ на события, которые затрагивают индивидуальные тайлы.
ISidebarAlarm	

Перечисления

AlarmState	
BasketFlags	
DragWindowPos	
InvokeFolderActionEnum	Перечисление для принятия решения, что делать, когда пользователь вызывает ItemControl для папки. Используется с InvokeFolderAction и ItemControlInvokeFolderAction.
MAAction	Константы, используемые со свойством Action для определения действия события.
SelectionModeEnum	

Делегаты

ChildrenWontFitHandler	
DocumentCompleteEventHandler	
DropEventHandler	
DropHandler	
DummyDelegateToSetupAppDomainProperly	
ExtraSpaceHandler	
FlyoutClosingEventHandler	
FlyoutLinkClickEventHandler	Представляет метод, который обрабатывает события FlyoutLinkClickEvent
FolderContentsChangedEventHandler	
NavigateErrorEventHandler	
NotificationClickedEventHandler	Представляет метод, который обрабатывает события NotificationClickedEvent
NotifyCompleteEventHandler	Представляет метод, который обрабатывает события NotifyCompleteEvent.
RMAActionEventHandler	Представляет метод, который обрабатывает события RMAActionEvent.
SetDataArrayEventHandler	Представляет метод, который обрабатывает события SetDataArrayEvent.

System.Windows.Control

Обеспечивает классы и интерфейсы, используемые компонентами оболочки.

Нижеследующие таблицы перечисляют примеры членов, раскрываемых пространством имен System.Windows.Controls.

Классы

ContactControl
ContactPickerDialog
ContactPropertyRequest
ContactPropertyRequestCollection
ContactSelection
ContactSelectionCollection
ContactTextBox
ContactTextBoxSelectionChangedEventArgs
ContactTextBoxTextChangedEventArgs
ContactTextBoxTextResolvedEventArgs
IncludeContactEventArgs
IteratedEventArgs
Iterator
OKCancelApplyEventArgs
OKCancelApplyStrip

Перечисления

ContactControlPropertyPosition
ContactPickerDialogLayout
ContactPropertyCategory
ContactPropertyType
ContactType
OKCancelApplyType

Делегаты

ContactTextBoxSelectionChangedEventHandler
ContactTextBoxTextChangedEventHandler
ContactTextBoxTextResolvedEventHandler
IncludeContactEventHandler
IteratedEventHandler

OKCancelApplyEventHandler
OpenedEventHandler

Примерная вычислительная система и среда

На фиг.4 изображен пример подходящей вычислительной среды 400, в которой может быть осуществлена инфраструктура 132 программирования (или полностью, или частично). Вычислительная среда 400 может использоваться в компьютерных и сетевых архитектурах, описанных в настоящей заявке.

Примерная вычислительная среда 400 представляет собой только один пример вычислительной среды и, как предполагается, не означает какого-либо ограничения в отношении объема использования или функциональных возможностей компьютерных и сетевых архитектур. Вычислительная среда 400 не должна интерпретироваться как имеющая какую-либо зависимость или требование, относящееся к любому одному или комбинации компонентов, изображенных в примерной вычислительной среде 400.

Инфраструктура 132 может быть реализована с многочисленными другими средами или конфигурациями вычислительных систем общего назначения или специального назначения. Примеры общеизвестных вычислительных систем, сред и/или конфигураций, которые могут быть пригодны для использования, включают в себя, но не ограничиваются ими, персональные компьютеры, серверные компьютеры, мультипроцессорные системы, микропроцессорные системы, сетевые персональные компьютеры (ПК), миникомпьютеры, мейнфреймы, распределенные вычислительные среды, которые включают в себя любую из вышеупомянутых систем или устройств, и т.д. Компактные версии или версии с поднабором инфраструктуры также могут осуществляться на клиентах с ограниченными ресурсами, такими как сотовые телефоны, персональные цифровые помощники, карманные компьютеры или другие устройства передачи данных/вычислительные устройства.

Инфраструктура 132 может описываться в общем контексте исполняемых компьютером инструкций, таких как программные модули, исполняемые одним или несколькими компьютерами или другими устройствами. Обычно программные модули включают в себя подпрограммы, программы, объекты, компоненты, структуры данных и т.д., которые выполняют конкретные задачи или воплощают определенные абстрактные типы данных. Инфраструктура 132 также может быть реализована в распределенных вычислительных средах, где задачи выполняются удаленными обрабатывающими устройствами, которые связаны через сеть передачи данных. В распределенной вычислительной среде программные модули могут располагаться как на локальных, так и на удаленных носителях данных компьютера, включая устройства хранения в памяти.

Вычислительная среда 400 включает в себя вычислительное устройство общего назначения в виде компьютера 402. Компоненты компьютера 402 могут включать в себя, но не ограничиваются ими, один или несколько процессоров или блоков 404 обработки, системную память 406 и системную шину 408, которая соединяет различные системные компоненты, включая процессор 404, с системной памятью 406.

Системная шина 408 представляет одну или более из нескольких возможных типов шинных структур, включая шину памяти или контроллер памяти, периферийную шину, ускоренный графический порт и процессорную или локальную шину, используя любую из многочисленных шинных архитектур. В качестве примера, такие архитектуры могут включать в себя шину архитектуры промышленного стандарта (ISA), шину микроканальной архитектуры (MCA), шину расширенной АПС (EISA), локальную шину Ассоциации по стандартам в области

видеоэлектроники (VESA) и шину межсоединений периферийных компонентов (PCI), также известную как шина расширения.

Компьютер 402 обычно включает в себя многочисленные считываемые компьютером носители. Такими носителями могут быть любые доступные носители, к которым может обращаться компьютер 402, и они включают в себя как энергозависимые, так и энергонезависимые носители, как съемные, так и несъемные носители.

Системная память 406 включает в себя считываемые компьютером носители в виде энергозависимой памяти, такой как оперативное запоминающее устройство (ОЗУ) 410, и/или энергонезависимой памяти, такой как постоянное запоминающее устройство (ПЗУ) 412. Базовая система 414 ввода-вывода (БСВВ, BIOS), содержащая базовые подпрограммы, которые способствуют передаче информации между элементами внутри компьютера 402, например, во время запуска, хранится в ПЗУ 412. ОЗУ 410 обычно содержит данные и/или программные модули, которые имеют немедленный доступ к ним и/или в настоящее время выполняются блоком 404 обработки.

Компьютер 402 также может включать в себя другие съемные/несъемные, энергозависимые/энергонезависимые носители данных компьютера. В качестве примера, на фиг.4 изображен накопитель 416 на жестких дисках для считывания с несъемного, энергонезависимого магнитного носителя (не показан) и записи на него, накопитель 418 на магнитных дисках для считывания со съемного, энергонезависимого магнитного диска 420 (например, дискеты) и записи на него и накопитель 422 на оптических дисках для считывания и записи на съемный, энергонезависимый оптический диск 424, такой как компакт-диск (CD-ROM), цифровой многофункциональный диск (DVD-ROM) или другой оптический носитель. Накопитель 416 на жестких дисках, накопитель 418 на магнитных дисках и накопитель 422 на оптических дисках, каждый, подсоединяется к системной шине 408 посредством одного или нескольких интерфейсов 426 носителя данных. Альтернативно, накопитель 416 на жестких дисках, накопитель 418 на магнитных дисках и накопитель 422 на оптических дисках могут быть подсоединены к системной шине 408 посредством одного или нескольких интерфейсов (не показаны).

Накопители на дисках и связанные с ними считываемые компьютером носители обеспечивают энергонезависимое хранение считываемых компьютером инструкций, структур данных, программных модулей и других данных для компьютера 402. Хотя в примере изображен жесткий диск 416, съемный магнитный диск 420 и съемный оптический диск 424, необходимо понять, что другие типы считываемых компьютером носителей, которые могут хранить данные, к которым может обращаться компьютер, такие как магнитные кассеты или другие магнитные устройства хранения, карты флэш-памяти, компакт-диск, цифровые многофункциональные диски (ЦМД) или другие оптические запоминающие устройства, оперативные запоминающие устройства (ОЗУ), постоянные запоминающие устройства (ПЗУ), электрически стираемые программируемые постоянные запоминающие устройства (ЭСППЗУ) и аналогичные, также могут использоваться для осуществления примерной вычислительной системы и среды.

Любое количество программных модулей может храниться на жестком диске 416, магнитном диске 420, оптическом диске 424, в ПЗУ 412 и/или ОЗУ 410, включая, в качестве примера, операционную систему 426, одну или несколько прикладных программ 428, другие программные модули 430 и данные 432 программ.

Операционная система 426, одна или несколько прикладных программ 428, другие программные модули 430 и данные 432 программ (или некоторая их комбинация), каждая, может включать в себя элементы инфраструктуры 132 программирования.

Пользователь может вводить команды и информацию в компьютер 402 при помощи устройств ввода, таких как клавиатура 434 и указательное устройство 436 (например, «мышь»). Другие устройства 438 ввода (конкретно не показаны) могут включать в себя микрофон, джойстик, игровой планшет, антенну спутниковой связи, последовательный порт, сканер и/или т.п. Эти и другие устройства ввода подсоединяются к блоку 404 обработки через интерфейсы 440 ввода-вывода, которые подсоединяются к системной шине 408, но могут подсоединяться посредством другого интерфейса и шинных структур, таких как параллельный порт, игровой порт или универсальную последовательную шину (USB).

Монитор 442 или другой тип устройства отображения также может подсоединяться к системной шине 408 через интерфейс, такой как видеоадаптер 444. В дополнение к монитору 442 другие периферийные устройства вывода могут включать в себя компоненты, такие как громкоговорители (не показаны) и принтер 446, которые могут подсоединяться к компьютеру 402 при помощи интерфейсов 440 ввода-вывода.

Компьютер 402 может работать в сетевой среде, используя логические соединения с одним или несколькими удаленными компьютерами, такими как удаленное вычислительное устройство 448. В качестве примера, удаленным вычислительным устройством 448 может быть персональный компьютер, портативный компьютер, сервер, маршрутизатор, сетевой компьютер, одноранговое устройство или другой общий сетевой узел, и т.д. Удаленное вычислительное устройство 448 изображено в виде портативного компьютера, который может включать в себя многие или все из элементов и признаков, описанных в настоящей заявке в отношении компьютера 402.

Логические соединения между компьютером 402 и удаленным компьютером 448 изображаются как локальная сеть (ЛС) 450 и общая глобальная сеть (ГС) 452. Такие сетевые среды являются обычным делом в офисах, компьютерных сетях масштаба предприятия, интрасетях и Интернет.

При осуществлении в сетевой среде ЛС компьютер 402 соединяется с локальной сетью 450 при помощи сетевого интерфейса или адаптера 454. При осуществлении в сетевой среде ГС компьютер 402 обычно включает в себя модем 456 или другое средство для установления связи через глобальную сеть 452. Модем 456, который может быть внутренним или внешним относительно компьютера 402, может быть соединен с системной шиной 408 при помощи интерфейсов 440 ввода-вывода или других подходящих механизмов. Необходимо понять, что изображенные сетевые соединения являются примерными и что могут использоваться другие средства для установления линии (линий) связи между компьютерами 402 и 448.

В сетевой среде, такой как изображенная с вычислительной средой 400, программные модули, описанные в отношении компьютера 402, или его частей, могут храниться на удаленном устройстве хранения данных. В качестве примера, удаленные прикладные программы 458 постоянно находятся в устройстве памяти удаленного компьютера 448. Для целей иллюстрации прикладные программы и другие исполняемые программные компоненты, такие как операционная система, изображаются в настоящей заявке в виде дискретных блоков, хотя признается, что такие программы и компоненты постоянно находятся в различные моменты времени в различных компонентах хранения вычислительного устройства 402 и исполняются процессором(ами) данных компьютера.

Осуществление инфраструктуры 132, и, в частности, ИПП 142 (API) или вызовов, сделанных в ИПП 142, может храниться или передаваться посредством некоторого считываемого компьютером носителя. Считываемыми компьютером носителями могут быть любые доступные носители, к которым может обращаться компьютер. В качестве примера, а не ограничения, считываемые компьютером носители могут содержать «носители данных компьютера» и «среды передачи данных». «Носители данных компьютера» включают в себя энергозависимые и энергонезависимые, съемные и несъемные носители, выполненные любым способом или по любой технологии для хранения информации, такой как считываемые компьютером инструкции, структуры данных, программные модули или другие данные. Носители данных компьютера включают в себя, но не ограничиваются ими, ОЗУ, ПЗУ, ЭСППЗУ, флэш-память или другую технологию памяти, компакт-диск, цифровые многофункциональные диски (ЦМД) или другое оптическое запоминающее устройство, магнитные кассеты, магнитную ленту, запоминающее устройство на магнитных дисках или другие магнитные запоминающие устройства, или любой другой носитель, который может использоваться для хранения требуемой информации и к которому может обращаться компьютер.

«Среды передачи данных» обычно охватывают считываемые компьютером инструкции, структуры данных, программные модули или другие данные в модулированном данными сигнале, таком как сигнал несущей или другой транспортный механизм. Среда передачи данных также включает в себя любую среду доставки информации. Термин «модулированный данными сигнал» означает сигнал, одна или несколько характеристик которого устанавливаются или изменяются таким образом, чтобы кодировать информацию в сигнале. В качестве примера, а не ограничения, среда передачи данных включает в себя проводную среду, такую как проводная сеть или прямое проводное соединение, и беспроводную среду, такую как акустическая, радиочастотная (РЧ), инфракрасная и другая беспроводная среда. Комбинации любых из вышеупомянутых также включаются в сферу рассмотрения считываемых компьютером носителей.

Альтернативно, части инфраструктуры могут быть реализованы аппаратными средствами или комбинацией аппаратных средств, программного обеспечения и/или программно-аппаратными средствами. Например, одна или несколько специализированных интегральных схем (специализированных ИС) или программируемые логические устройства (ПЛУ) могут быть разработаны или запрограммированы для осуществления одной или нескольких частей инфраструктуры.

Интерфейс программирования (или, проще, интерфейс) может рассматриваться как любой механизм, процесс, протокол, позволяющий одному или нескольким сегментам кода устанавливать связь или обращаться к функциональной возможности, предусматриваемой одним или несколькими другими сегментами кода. Альтернативно, интерфейс программирования может рассматриваться как один или несколько механизмов, методов, вызовов функций, модулей, объектов и т.д. компонента системы, способных соединяться с возможностью установления связи с одним или несколькими механизмами, методами, вызовами функций, модулями и т.д. другого компонента(ов). Термин «сегмент кода» в предыдущем предложении, как предполагается, включает в себя одну или несколько инструкций или строк кода и включает в себя, например, модули программ, объекты, подпрограммы, функции и т.д. независимо от применяемой терминологии, или от того, компилировались ли

отдельно сегменты кода, или предусматривались ли сегменты кода в качестве исходного, промежуточного или объектного кода, использовались ли сегменты кода в системе или процессе поддержки исполнения программ, или расположены ли они на одной и той же или на различных машинах или расределены по многочисленным машинам, или реализуются ли функциональные возможности, представляемые сегментами кода, полностью программным обеспечением, полностью аппаратными средствами или комбинацией аппаратных средств и программного обеспечения.

Абстрактно, интерфейс программирования может рассматриваться, в общем, как показано на фиг.5 или фиг.6. На фиг.5 изображен интерфейс Интерфейс1 в качестве туннеля, через который происходит передача первого и второго сегмента кода. На фиг.6 изображен интерфейс, содержащий объекты И1 и И2 интерфейса (которые могут быть или могут не быть частью первого и второго сегмента кода), которые позволяют первому и второму сегменту кода системы устанавливать связь при помощи среды М. На фиг.6 можно рассматривать объекты И1 и И2 интерфейса в качестве отдельных интерфейсов одной и той же системы, и также можно рассматривать, что объекты И1 и И2 плюс среда М составляют интерфейс. Хотя на фиг.5 и 6 показан двунаправленный поток и интерфейсы на каждой стороне потока, некоторые варианты осуществления могут иметь поток информации только в одном направлении (или без потока информации, как описано ниже) или могут иметь только объект интерфейса на одной стороне. В качестве примера, а не ограничения, термины, такие как интерфейс прикладного программирования или программ (ИПП, API), точка входа, метод, функция, подпрограмма, вызов удаленной процедуры и интерфейс компонентной объектной модели (КОМ), заключаются в определении интерфейса программирования.

Аспекты такого интерфейса программирования могут включать в себя метод, посредством которого первый сегмент кода передает информацию (где «информация» используется в ее самом широком смысле и включает в себя данные, команды, запросы и т.д.) второму сегменту кода; метод, посредством которого второй сегмент кода принимает информацию; и структуру, последовательность, синтаксис, организацию, схему, тактирование и содержимое информации. В этом отношении лежащая в основе транспортная среда сама по себе может быть маловажной для работы интерфейса, является ли среда проводной или беспроводной или их комбинацией, пока информация передается так, как определено интерфейсом. В некоторых ситуациях информация может не передаваться в одном или обоих направлениях в обычном смысле, так как пересылка информации может или происходить при помощи другого механизма (например, информация помещается в буфер, файл и т.д. отдельно от потока информации между сегментами кода), или быть несуществующей как тогда, когда один сегмент кода просто обращается к функциональной возможности, выполняемой вторым сегментом кода. Любые или все из этих аспектов могут быть важными в данной ситуации, например, в зависимости от того, являются ли сегменты кода частью системы в слабосвязанной или сильносвязанной конфигурации, и поэтому данный список должен рассматриваться иллюстративным и неограничивающим.

Данное понятие интерфейса программирования известно для специалиста в данной области техники и ясно из вышеприведенного подробного описания изобретения. Существуют, однако, другие пути осуществления интерфейса программирования, и, если только специально не исключено, они также, как предполагается, охватываются формулой изобретения, приложенной к описанию изобретения. Такие другие пути

могут быть, по-видимому, более совершенными или комплексными, чем упрощенный вид на фиг.5 и 6, но они, тем не менее, выполняют аналогичную функцию для достижения такого же общего результата. Ниже теперь кратко описываются некоторые иллюстративные альтернативные осуществления интерфейса программирования.

A. Разложение на составные части

Передача данных от одного сегмента кода другому может осуществляться косвенно посредством разбиения передачи данных на многочисленные дискретные передачи данных. Это схематически изображается на фиг.7 и 8. Как показано, некоторые интерфейсы могут быть описаны на языке делимых наборов функциональных возможностей. Таким образом, функциональные возможности интерфейса на фиг.5 и 6 могут быть разложены на составные части для достижения такого же результата, точно так, как можно математически получить 24, или 2 умножить на 2, умножить на 3, умножить на 2. Следовательно, как изображено на фиг.7, функция, обеспечиваемая интерфейсом Интерфейс1, может быть разделена для преобразования передачи данных интерфейса на множество интерфейсов Интерфейс1A, Интерфейс 1B, Интерфейс 1C и т.д., в то же самое время достигая такого же результата. Как показано на фиг.8, функция, обеспечиваемая интерфейсом И1, может быть разделена на многочисленные интерфейсы И1a, И1b, И1c и т.д., в то же самое время достигая такого же результата. Аналогично, интерфейс И2 второго сегмента кода, который принимает информацию от первого сегмента кода, может быть разложен на многочисленные интерфейсы И2a, И2b, И2c и т.д. При разложении на составные части число интерфейсов, включенных в 1-й сегмент кода, необязательно соответствует числу интерфейсов, включенных во 2-й сегмент кода. В обоих случаях на фиг.7 и 8 функциональная сущность интерфейсов Интерфейс1 и И1 остается той же, что и на фиг.5 и 6 соответственно. Разложение на составные части интерфейсов также может руководствоваться ассоциативностью, коммутативностью и другими математическими свойствами, так что разложение на составные части может быть трудным для распознавания. Например, порядок операций может быть маловажным и, следовательно, функция, выполняемая интерфейсом, может выполняться задолго до достижения интерфейса, посредством другой части кода или интерфейса, или выполняться отдельным компонентом системы. Кроме того, для специалиста в области программирования понятно, что существуют многочисленные пути выполнения вызовов различных функций, которые достигают одинакового результата.

B. Переопределение

В некоторых случаях можно игнорировать, добавить или переопределить некоторые аспекты (например, параметры) интерфейса программирования, в то же самое время, тем не менее, достигая предполагаемого результата. Это изображено на фиг.9 и 10. Например, предположим, что интерфейс Интерфейс1 на фиг.5 включает в себя вызов функции Square (ввод, точность, вывод), вызов, который включает в себя три параметра ввод, точность и вывод, и который выдается от 1-го сегмента кода 2-му сегменту кода. Если средний параметр точность не представляет собой интереса в данном сценарии, как показано на фиг.9, он может просто игнорироваться или даже заменяться не имеющим смысла (в данной ситуации) параметром. Можно также добавить параметр “дополнительный”, который не представляет интереса. В любом случае может быть достигнута функциональная возможность возведения в квадрат при условии, что возвращаются выходные данные, после того как входные данные возводятся в квадрат вторым сегментом кода. Точность может быть значащим

параметром для некоторой части в дальнейшем процессе или другой части вычислительной системы; однако, если определено, что точность не является обязательной для узкой цели вычисления возведения в квадрат, она может быть заменена или игнорирована. Например, вместо передачи действительного значения точность, не имеющее смысла значение, такое как день рождения, может быть передано без неблагоприятного влияния на результат. Аналогично, как показано на фиг.10, интерфейс И1 заменяется интерфейсом И1', переопределенным для игнорирования или добавления параметров к интерфейсу. Интерфейс И2 может быть переопределен аналогичным образом как интерфейс И2', преопределенный для игнорирования необязательных параметров или параметров, которые могут быть обработаны где-то в другом месте. Вопрос в данном случае заключается в том, что в некоторых случаях интерфейс программирования может включать в себя аспекты, такие как параметры, которые не нужны для некоторой цели, и поэтому они могут быть проигнорированы или переопределены, или обработаны где-то в другом месте для других целей.

C. Встроенное кодирование

Также может быть допустимым объединение некоторых или всех функциональных возможностей двух отдельных модулей программ, так что «интерфейс» между ними меняет форму. Например, функциональные возможности на фиг.5 и 6 могут быть преобразованы в функциональные возможности фиг.11 и 12 соответственно. На фиг.11 предыдущие 1-й и 2-й сегменты кода на фиг.5 объединяются в модуль, содержащий обоих. В данном случае сегменты кода могут все еще передавать друг другу данные, но интерфейс может адаптироваться в форму, которая более пригодна для одного модуля. Таким образом, например, формальные операторы Call и Return больше могут быть не нужны, но все еще может быть в действии аналогичная обработка или ответ(ы), согласующиеся с интерфейсом Интерфейс1. Аналогично, как показано на фиг.12, часть (или весь) интерфейса И2 на фиг.6 может быть встроена в интерфейс И1, образуя интерфейс И1". Как изображено, интерфейс И2 разделяется на И2a и И2b, и часть интерфейса И2a кодируется встроенным образом в интерфейс И1, образуя интерфейс И1". Для конкретного примера, рассмотрим, что интерфейс И1 на фиг.6 выполняет вызов функции square (ввод, вывод), которая принимается интерфейсом И2, который после обработки значения, переданного при помощи ввода (для его возведения в квадрат) вторым сегментом кода, передает обратно возведенный в квадрат результат при помощи вывода. В таком случае обработка, выполняемая вторым сегментом кода (возводящим в квадрат ввод), может быть выполнена первым сегментом кода без вызова в интерфейс.

D. Разъединение

Передача данных от одного сегмента кода другому может осуществляться косвенно посредством разбиения передачи данных на многочисленные дискретные передачи данных. Это изображается схематически на фиг.13 и 14. Как показано на фиг.13, одна или несколько частей промежуточного программного обеспечения (интерфейс(ы) разъединения, так как они разъединяют функциональные возможности и/или функции интерфейсов от исходного интерфейса) предусматриваются для преобразования передачи данных по первому интерфейсу Интерфейс1 для соответствия их другому интерфейсу, в данном случае интерфейсам Интерфейс2A, Интерфейс2B и Интерфейс2C. Это может быть сделано, например, там, где имеется установленная база прикладных программ, предназначенных для установления связи, например, с операционной системой в соответствии с протоколом Интерфейс1, но

операционная система переходит на использование другого интерфейса, в данном случае интерфейсов Интерфейс2A, Интерфейс2B и Интерфейс2C. Вопрос заключается в том, что исходный интерфейс, используемый 2-м сегментом кода, меняется, так что он больше не является совместимым с интерфейсом, используемым 1-м сегментом кода, и поэтому используется промежуточный для того, чтобы сделать совместимыми старый и новый интерфейсы. Аналогично, как показано на фиг.14, может быть введен третий сегмент кода при помощи интерфейса DI1 (ИР1) разъединения для приема передачи данных от интерфейса И1 и при помощи интерфейса DI2 (ИР2) разъединения для передачи функциональных возможностей интерфейса, например, интерфейсам И2a и И2b, доработанных для работы с DI2, но для получения такого же функционального результата. Аналогично, DI1 и DI2 могут работать вместе для перевода функциональных возможностей интерфейсов И1 и И2 на фиг.6 в новую операционную систему, в то же самое время обеспечивая такой же или аналогичный функциональный результат.

Е. Перезапись

Еще другим возможным вариантом является динамическая перезапись кода для замены функциональных возможностей интерфейса чем-либо еще, но тем, что достигает такого же общего результата. Например, может быть система, в которой сегмент кода, представленный промежуточным языком (например, промежуточным языком Microsoft, Java-байт-кодом и т.д.), обеспечивается оперативным (JIT) компилятором или интерпретатором в среде исполнения (например, той, которая обеспечивается инфраструктурой .Net, средой исполнения языка Java или другими подобными средами исполнения). Оперативный (JIT) компилятор может быть написан так, чтобы динамически преобразовывать передачу данных от 1-го сегмента кода 2-му сегменту кода, т.е. согласовывать их с различными интерфейсами, что может потребоваться 2-м сегментом кода (или исходным, или другим 2-м сегментом кода). Это изображено на фиг.15 и 16. Как показано на фиг.15, данный подход аналогичен сценарию разъединения, описанному выше. Это может быть сделано, например, там, где установленная база прикладных программ предназначена для передачи данных операционной системе в соответствии с протоколом Интерфейс 1, но затем операционная система переходит на использование другого интерфейса. Оперативный (JIT) компилятор может использоваться для согласования передачи данных «на лету» от прикладных программ с установленной базой на новый интерфейс операционной системы. Как изображено на фиг.16, данный подход динамической перезаписи интерфейса(ов) может также применяться для динамического разложения на составные части или изменения иным образом интерфейса(ов).

Также отмечается, что вышеописанные сценарии для достижения такого же или аналогичного результата в качестве интерфейса при помощи альтернативных вариантов выполнения также могут объединяться различными путями, последовательно и/или параллельно, или с другим промежуточным кодом. Таким образом, альтернативные варианты выполнения, представленные выше, не являются взаимно исключающими и могут соединяться, сопоставляться и объединяться для получения таких же или эквивалентных сценариев относительно общих сценариев, представленных на фиг.5 и 6. Также необходимо отметить, что, как и с большинством программных конструкций, существуют другие подобные пути достижения таких же или аналогичных функциональных возможностей интерфейса, которые могут не описываться в данной заявке, но, тем не менее, представляются сущностью и объемом

изобретения, т.е. отмечается, что именно по меньшей мере частично функциональные возможности, представленные интерфейсом, и выгодные результаты, достигаемые им, лежат в основе ценности интерфейса.

Заключение

Хотя изобретение было описано на языке, характерном для конструктивных признаков и/или методологических действий, необходимо понять, что изобретение, определенное в прилагаемой формуле изобретения, необязательно ограничивается конкретными описанными признаками или действиями. Скорее, конкретные признаки и действия описываются в качестве примерных форм осуществления заявляемого изобретения.

Формула изобретения

1. Интерфейс программирования, осуществленный на одном или нескольких считываемых компьютером запоминающих носителях, содержащий:

иерархическое пространство имен, включающее в себя набор типов для пользовательского интерфейса, скомпонованных в упомянутое иерархическое пространство имен, причем иерархическое пространство имен включает в себя функциональные возможности, позволяющие выполнять определенные приложением вычисления;

идентификатор верхнего уровня, стоящий перед именем каждой группы в упомянутой иерархии так, чтобы на типы в каждой группе ссылаться посредством иерархического имени, которое включает в себя упомянутый выбранный идентификатор верхнего уровня, стоящий перед именем группы, содержащей тип;

первую группу услуг, относящуюся к повторно используемым элементам управления пользовательского интерфейса, причем первая группа услуг включает в себя элемент управления, который позволяет отображать изображения предварительного просмотра элементов;

вторую группу услуг, относящуюся к диалогам пользовательского интерфейса и мастерам пользовательского интерфейса, причем вторая группа услуг включает в себя первый диалог, чтобы разрешить открывать и сохранять файлы и папки;

третью группу услуг, относящуюся к расширению функциональных возможностей пользовательского интерфейса, причем третья группа услуг включает в себя функциональные возможности, обеспечивающие возможность идентификации определенных приложением «миниатюр»; и

четвертую группу услуг, относящуюся к расширению функциональных возможностей рабочего стола пользовательского интерфейса, причем четвертая группа услуг включает в себя функциональные возможности для того, чтобы разрешить отображение бокового поля на рабочем столе, причем первая и вторая и третья и четвертая группы услуг определены соответствующими пространствами имен упомянутого интерфейса программирования, причем одним из соответствующих пространств имен является пространство имен Desktop, при этом пространство имен Desktop включает в себя элементы в боковом поле, и при этом элемент управления BasketControl позволяет добавлять один или более элементов к боковому полю; и

объектную модель, которая может быть использована каждой из групп услуг, причем объектная модель включает в себя:

объект ExplorerItem, который описывает элементы, которые могут быть представлены пользователю,

объект Library, который описывает запрос в отношении объекта ExplorerItem,

объект ViewFields, который проецирует данные из объекта EplorerItem пользователю, и

объект StorageFavorites, который описывает ссылку на динамический список, генерируемый из объекта Library.

2. Интерфейс программирования по п.1, в котором первая группа услуг включает в себя:

первый элемент управления, который инкапсулирует опыт пользователя в хранении;

второй элемент управления, который делает возможным отображение элемента определенным прикладной программой образом;

третий элемент управления, который делает возможным добавление элементов к боковому полю рабочего стола, и

четвертый элемент управления, который делает возможным добавление элементов к списку.

3. Интерфейс программирования по п.1, в котором вторая группа услуг включает в себя:

первый мастер, делающий возможным запись на оптические диски; и

второй мастер, способствующий посылке изображений по электронной почте.

4. Интерфейс программирования по п.1, в котором третья группа услуг включает в себя:

первые функциональные возможности, обеспечивающие возможность добавления к контекстному меню; и

вторые функциональные возможности, обеспечивающие возможность выполнения вычислений при отображении информации, касающейся одного или более файлов или папок.

5. Интерфейс программирования по п.1, в котором четвертая группа услуг включает в себя функциональные возможности, обеспечивающие возможность отображения определяемых прикладной программой уведомлений на рабочем столе.

6. Система, реализуемая посредством одного или более компьютеров, которые содержат один или более считываемых компьютером запоминающих носителей, содержащая:

средство для организации набора типов для пользовательского интерфейса в иерархическое пространство имен, причем иерархическое пространство имен включает в себя функциональные возможности, позволяющие выполнять определенные приложением вычисления;

средство для предоставления первого набора функций, которые запускают в работу повторно используемые элементы управления пользовательского интерфейса;

средство для предоставления второго набора функций, которые запускают в работу повторно используемые диалоги пользовательского интерфейса и повторно используемые мастера пользовательского интерфейса; и

средство для предоставления третьего набора функций, которые запускают в работу расширение функциональных возможностей рабочего стола пользовательского интерфейса,

причем средство для предоставления первого набора функций включает в себя средство для предоставления одной или более функций, которые позволяют добавлять элементы к боковому полю рабочего стола,

средство для выбора идентификатора верхнего уровня и для вставки этого идентификатора верхнего уровня перед именем каждого набора, чтобы на типы в

каждом наборе ссылались посредством иерархического имени, которое включает в себя упомянутый выбранный идентификатор верхнего уровня, стоящий перед именем набора, содержащего тип, причем первый набор функций связан с идентификатором верхнего уровня Desktop и включает в себя элементы в боковом поле, и при этом элемент управления BasketControl позволяет добавлять один или более элементов к боковому полю; и

средство для предоставления объектной модели, которая может быть использована каждым из наборов функций, причем объектная модель включает в себя:

объект ExplorerItem, который описывает элементы, которые могут быть представлены пользователю,

объект Library, который описывает запрос в отношении объекта ExplorerItem,

объект ViewFields, который проецирует пользователю данные из объекта EplorerItem, и

объект StorageFavorites, который описывает ссылку на динамический список, генерируемый из объекта Library.

7. Система по п.6, дополнительно содержащая:

средство для предоставления четвертого набора функций, которые запускают в работу расширение функциональных возможностей пользовательского интерфейса.

8. Система по п.7, в которой средство для предоставления четвертого набора функций содержит:

средство для предоставления одной или более функций, обеспечивающих возможность добавления к контекстному меню;

средство для предоставления одной или более функций, обеспечивающих возможность идентификации определяемых прикладной программой «миниатюр»; и

средство для предоставления одной или более функций, обеспечивающих возможность выполнения вычислений при отображении информации, касающейся одного или более файлов или папок.

9. Система по п.6, в которой средство для предоставления первого набора функций содержит:

средство для предоставления одной или более функций, которые инкапсулируют опыт пользователя в хранении;

средство для предоставления одной или более функций, которые обеспечивают возможность отображения элемента определенным прикладной программой образом;

средство для предоставления одной или более функций, которые обеспечивают возможность добавления элементов к списку; и

средство для предоставления одной или более функций, которые обеспечивают возможность отображения изображений предварительного просмотра элементов.

10. Система по п.6, в которой средство для предоставления второго набора функций содержит:

средство для предоставления одной или более функций, которые обеспечивают возможность открытия и сохранения файлов и папок;

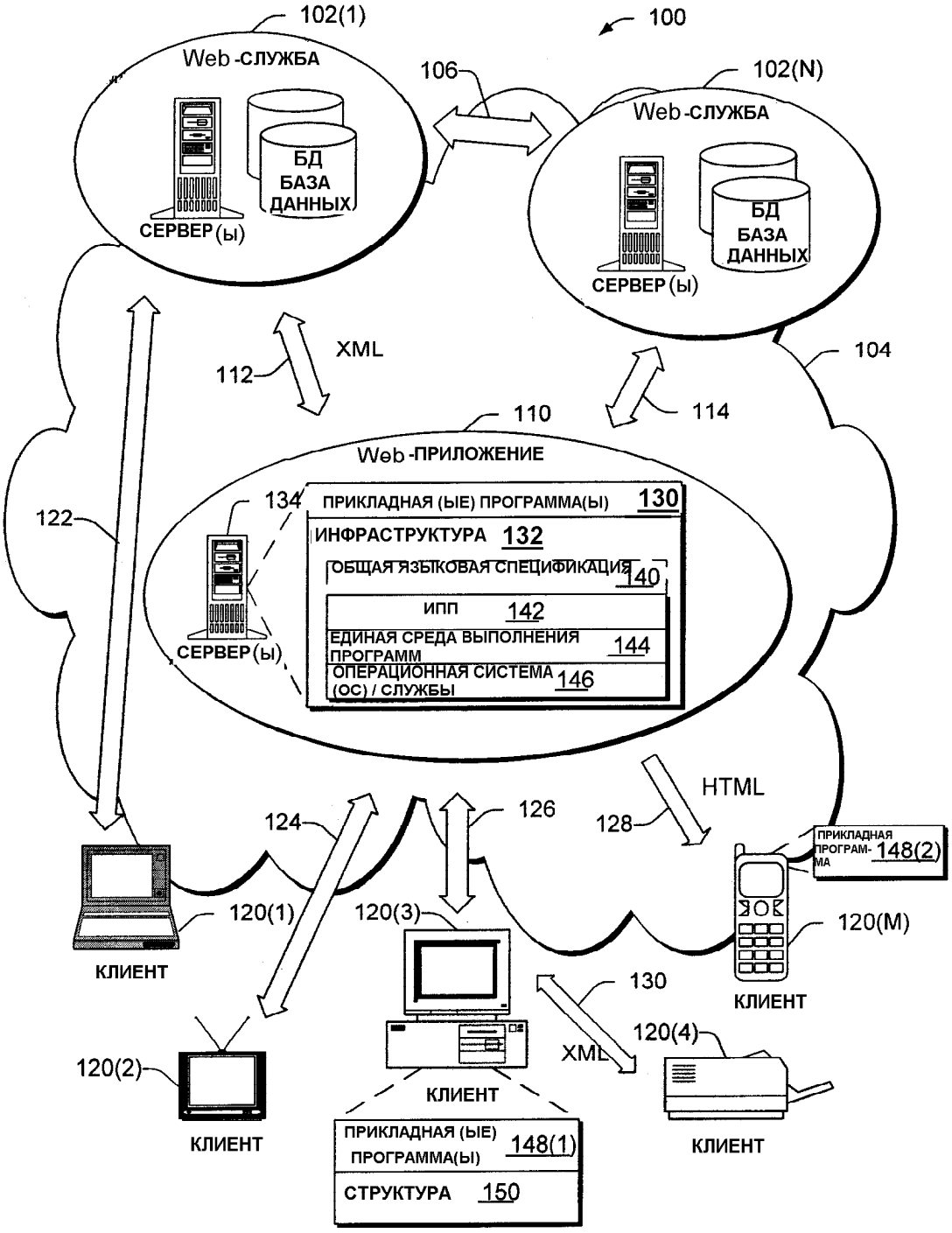
средство для предоставления одной или более функций, которые обеспечивают возможность записи на оптические диски; и

средство для предоставления одной или более функций, которые обеспечивают возможность изменения размеров изображений, подлежащих посылке по электронной почте.

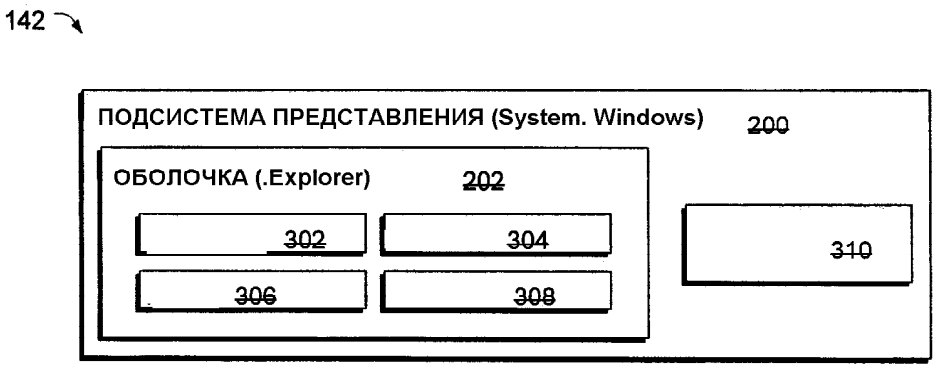
11. Система по п.6, в которой средство для предоставления третьего набора функций содержит:

средство для предоставления одной или более функций, которые обеспечивают возможность отображения бокового поля на рабочем столе; и

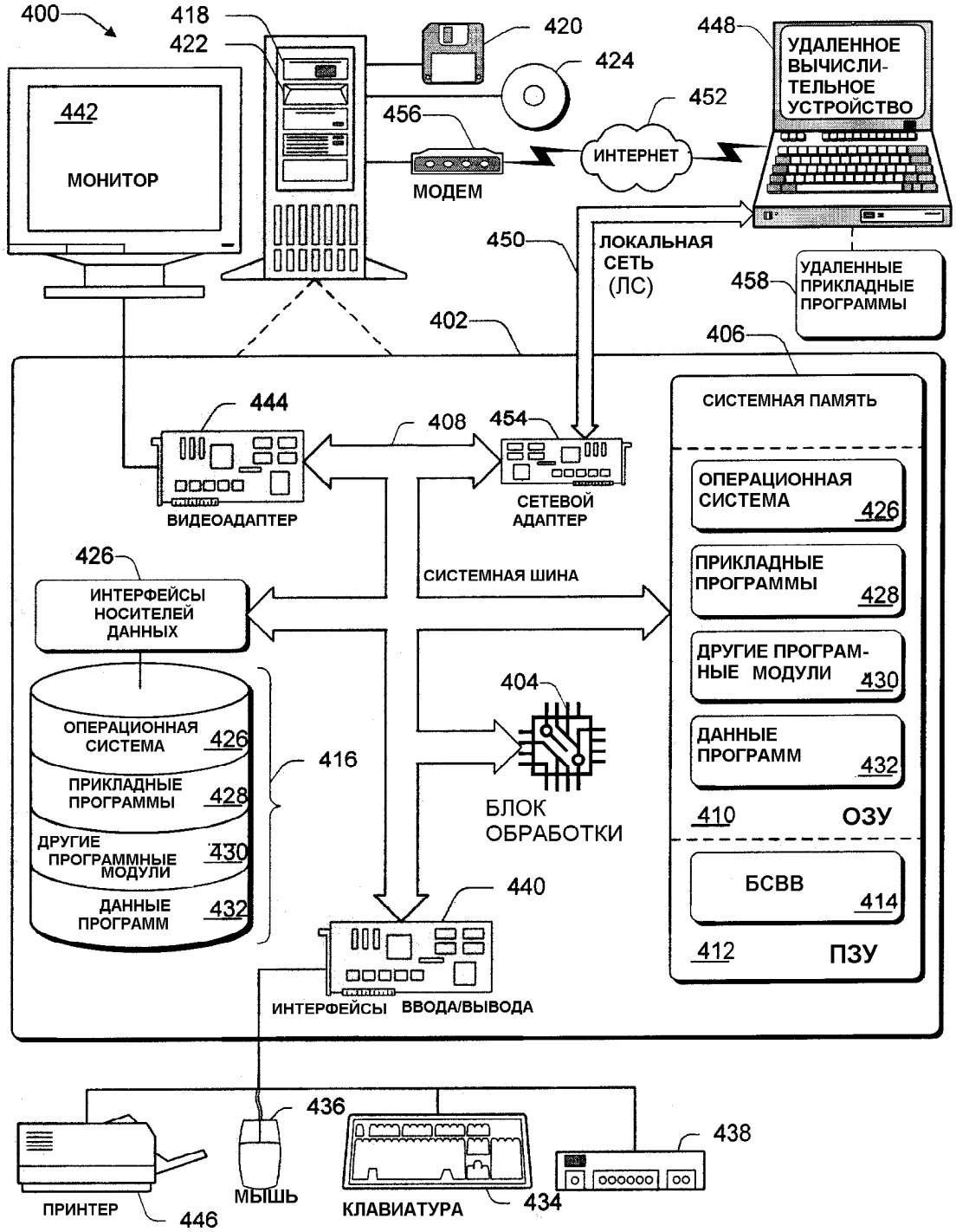
средство для предоставления одной или более функций, которые обеспечивают возможность отображения определяемых прикладной программой уведомлений на рабочем столе.


102(1)
100
Web -СЛУЖБА
106
102(N)
Web -СЛУЖБА
БД
БАЗА
ДАННЫХ
СЕРВЕР(Ы)
БД
БАЗА
ДАННЫХ
СЕРВЕР(Ы)
XML
112
104
110
114
Web -ПРИЛОЖЕНИЕ
122
134
ПРИКЛАДНАЯ (ЫЕ) ПРОГРАММА(Ы) 130
ИНФРАСТРУКТУРА 132
ОБЩАЯ ЯЗЫКОВАЯ СПЕЦИФИКАЦИЯ 140
ИПП 142
ЕДИНАЯ СРЕДА ВЫПОЛНЕНИЯ ПРОГРАММ 144
ОПЕРАЦИОННАЯ СИСТЕМА (ОС) / СЛУЖБЫ 146
СЕРВЕР(Ы)
HTML
124
126
128
ПРИКЛАДНАЯ ПРОГРАММА 148(2)
120(1)
120(3)
120(M)
КЛИЕНТ
КЛИЕНТ
130
120(4)
XML
120(2)
КЛИЕНТ
КЛИЕНТ
КЛИЕНТ
ПРИКЛАДНАЯ (ЫЕ) ПРОГРАММА(Ы) 148(1)
СТРУКТУРА 150

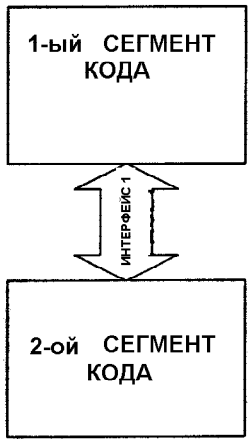
ФИГ.1


142
ПОДСИСТЕМА ПРЕДСТАВЛЕНИЯ (System. Windows) 200
ОБОЛОЧКА (.Explorer) 202
302
304
306
308
310

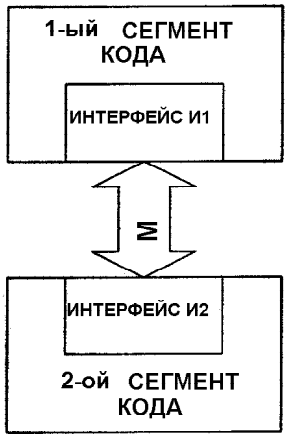
ФИГ.3


400
418
422
420
424
448
УДАЛЕННОЕ ВЫЧИСЛИ-ТЕЛЬНОЕ УСТРОЙСТВО
442
МОНИТОР
456
452
ИНТЕРНЕТ
МОДЕМ
450
ЛОКАЛЬНАЯ СЕТЬ (ЛС)
458
УДАЛЕННЫЕ ПРИКЛАДНЫЕ ПРОГРАММЫ
402
406
СИСТЕМНАЯ ПАМЯТЬ
444
408
454
ОПЕРАЦИОННАЯ СИСТЕМА 426
ВИДЕОАДАПТЕР
СЕТЕВОЙ АДАПТЕР
426
ПРИКЛАДНЫЕ ПРОГРАММЫ 428
ИНТЕРФЕЙСЫ НОСИТЕЛЕЙ ДАННЫХ
СИСТЕМНАЯ ШИНА
ДРУГИЕ ПРОГРАМ-НЫЕ МОДУЛИ 430
404
ОПЕРАЦИОННАЯ СИСТЕМА 426
ДАННЫЕ ПРОГРАММ 432
416
ПРИКЛАДНЫЕ ПРОГРАММЫ 428
БЛОК ОБРАБОТКИ
410
ОЗУ
ДРУГИЕ ПРОГРАММНЫЕ МОДУЛИ 430
БСВВ 414
440
ДАННЫЕ ПРОГРАММ 432
412
ПЗУ
ИНТЕРФЕЙСЫ ВВОДА/ВЫВОДА
436
438
ПРИНТЕР
446
МЫШЬ
КЛАВИАТУРА
434

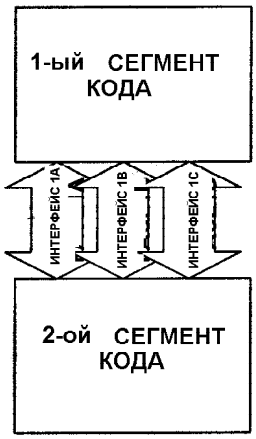
ФИГ.4



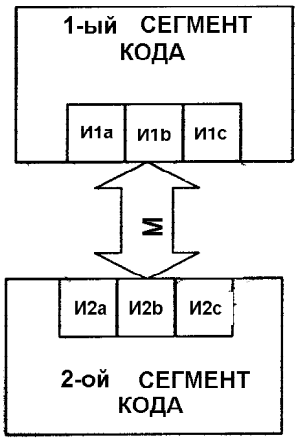
ФИГ.5



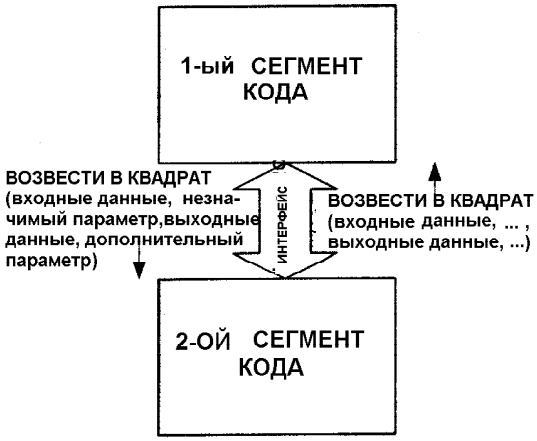
ФИГ.6



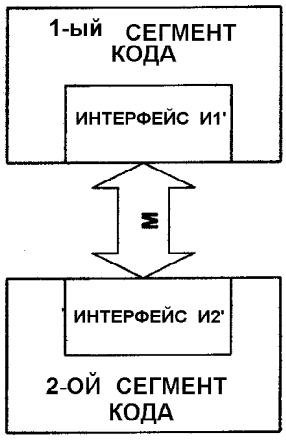
ФИГ.7



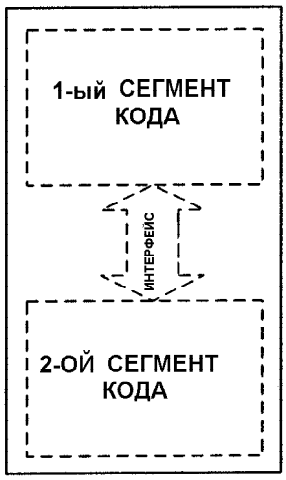
ФИГ.8



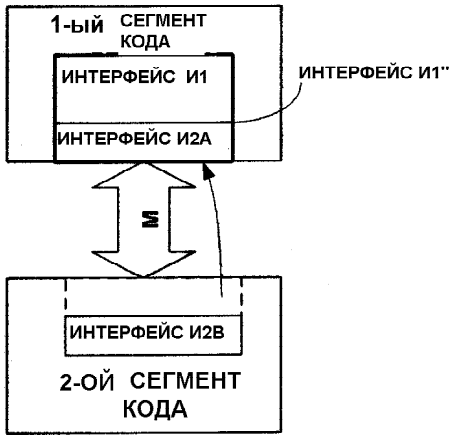
ФИГ.9



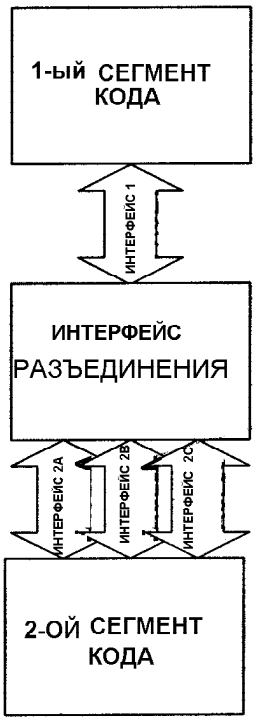
ФИГ.10



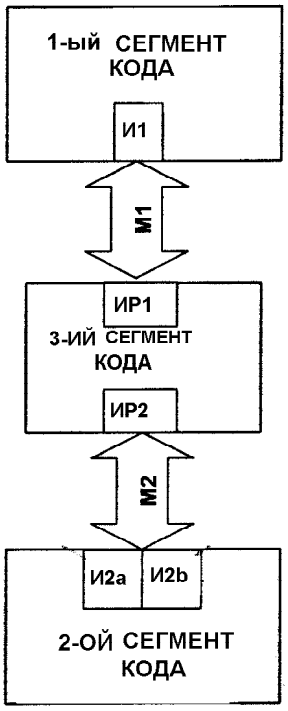
ФИГ.11



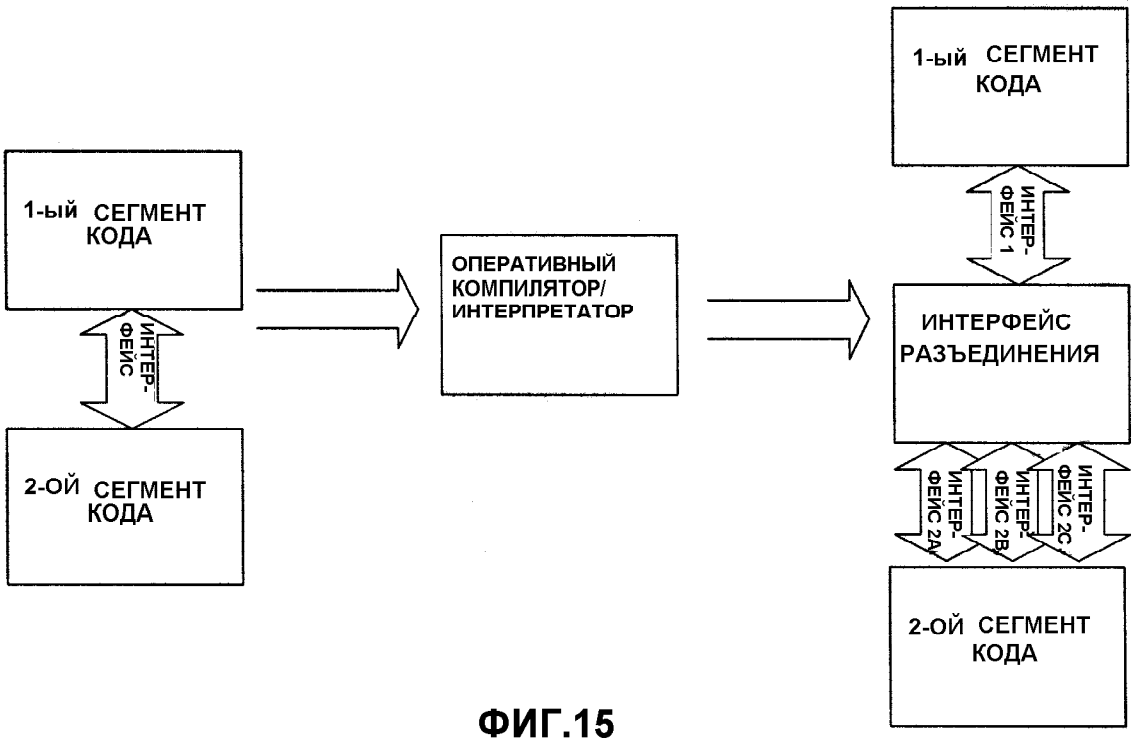
ФИГ.12



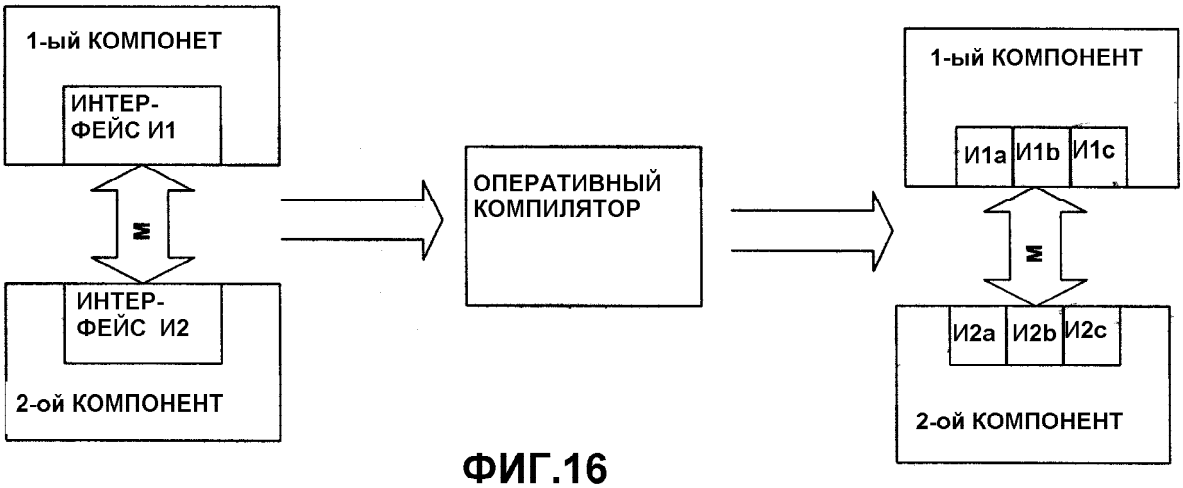
ФИГ.13


1-ый СЕГМЕНТ КОДА
И1
М1
ИР1
3-ИЙ СЕГМЕНТ КОДА
ИР2
М2
И2a
И2b
2-ОЙ СЕГМЕНТ КОДА

ФИГ.14


1-ый СЕГМЕНТ КОДА
ИНТЕР-ФЕЙС
2-ОЙ СЕГМЕНТ КОДА
ОПЕРАТИВНЫЙ КОМПИЛЯТОР/ ИНТЕРПРЕТАТОР
1-ый СЕГМЕНТ КОДА
ИНТЕР-ФЕЙС 1
ИНТЕРФЕЙС РАЗЪЕДИНЕНИЯ
ИНТЕР-ФЕЙС 2A
ИНТЕР-ФЕЙС 2B
ИНТЕР-ФЕЙС 2C
2-ОЙ СЕГМЕНТ КОДА

ФИГ.15


1-ый КОМПОНЕТ
ИНТЕР-
ФЕЙС И1
М
ИНТЕР-
ФЕЙС И2
2-ой КОМПОНЕНТ
ОПЕРАТИВНЫЙ
КОМПИЛЯТОР
1-ый КОМПОНЕНТ
И1a
И1b
И1c
М
И2a
И2b
И2c
2-ой КОМПОНЕНТ

ФИГ.16