РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 679 971** (13) **C2**

(51) МПК
G06F 9/44 (2006.01)
G06F 17/30 (2006.01)

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
G06F 17/30554 (2018.08); G06F 17/30731 (2018.08); G06F 8/20 (2018.08); G06F 8/33 (2018.08)

(21)(22) Заявка: **2016147085, 01.06.2015**

(24) Дата начала отсчета срока действия патента:
01.06.2015

Дата регистрации:
14.02.2019

Приоритет(ы):
(30) Конвенционный приоритет:
02.06.2014 US 62/006,662;
12.11.2014 US 14/539,521

(43) Дата публикации заявки: **01.06.2018** Бюл. № **16**

(45) Опубликовано: **14.02.2019** Бюл. № **5**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **01.12.2016**

(86) Заявка PCT:
US 2015/033554 (01.06.2015)

(87) Публикация заявки PCT:
WO 2015/187567 (10.12.2015)

Адрес для переписки:
129090, Москва, ул. Б.Спасская, 25, строение 3, ООО "Юридическая фирма Городисский и Партнеры"

(72) Автор(ы):
ШАКИРЗИАНОВ Антон (US),
НАРАЯНАН Сурия (US),
Ю Лян (US),
КАМИНСКИ Томаш (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ ТЕКНОЛОДЖИ ЛАЙСЕНСИНГ, ЭлЭлСи (US)

(56) Список документов, цитированных в отчете о поиске: **US 20100169871 A1, 01.07.2010. US 20050076017 A1, 07.04.2005. US 20070050343 A1, 01.03.2007. RU 2012119082 A, 20.11.2013.**

(54) ОСУЩЕСТВЛЕНИЕ ДОСТУПА К СЕМАНТИЧЕСКОМУ КОНТЕНТУ В СИСТЕМЕ РАЗРАБОТКИ

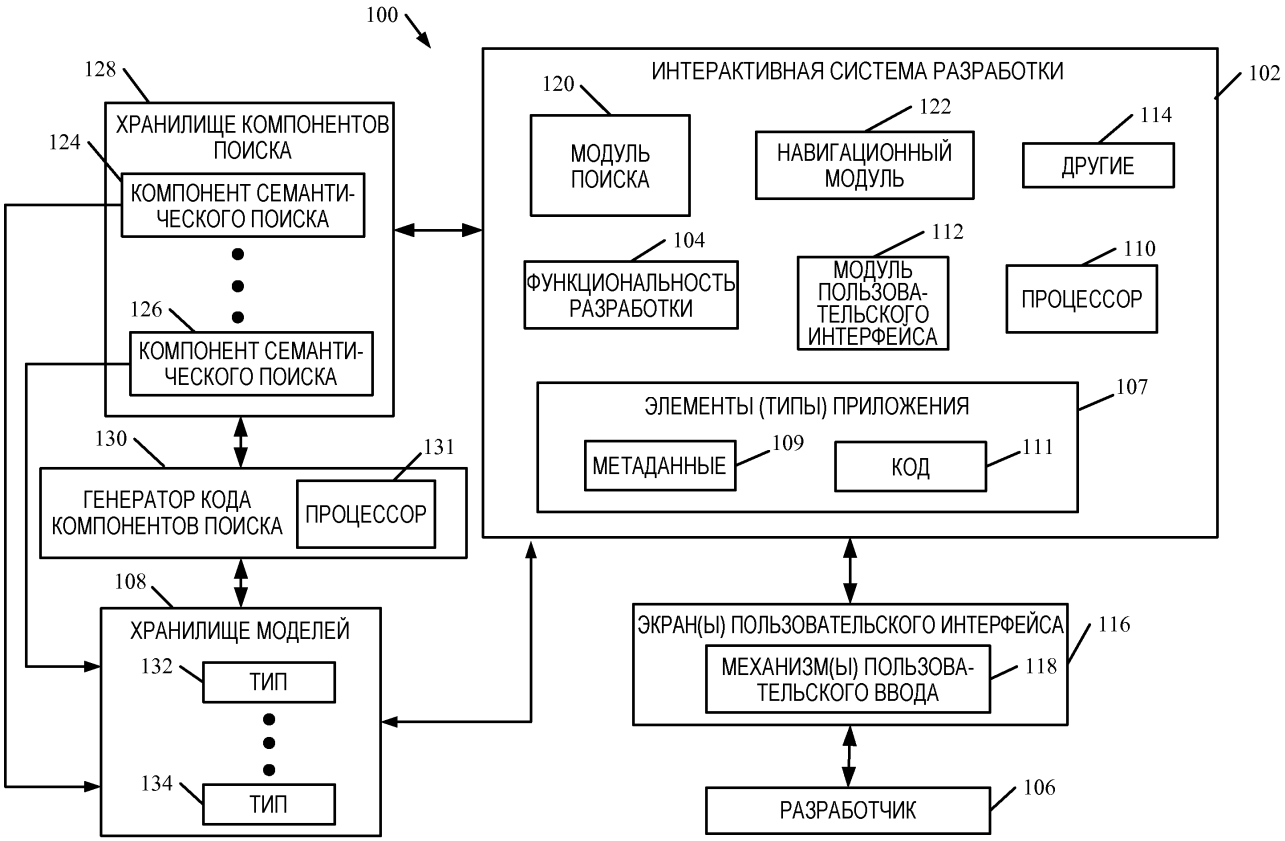
(57) Реферат:

Изобретение относится к области разработки программного обеспечения. Техническим результатом является упрощение процесса разработки. Система разработки элементов компьютерной системы и управления поиском, содержащая: модуль разработки, считывающий пользовательские вводы и преобразующий элементы компьютерной системы, причем элементы содержат типы, моделируемые в компьютерной системе, модуль пользовательского интерфейса, считывающий пользовательский поисковый ввод, принимаемый через механизм пользовательского ввода и поисковый механизм, идентифицирующий параметр поиска на основе типа поискового запроса и выдающий набор результатов поиска в отображении пользовательского интерфейса, при этом система дополнительно содержит

RU 2679971 C2

RU 2679971 C2

хранилище компонентов поиска, хранящее множество компонентов поиска, причем каждый компонент поиска соответствует заданному одному из упомянутых различных типов элементов и выполнен с возможностью осуществлять поиск в наборе свойств и методов на предмет элементов этого заданного типа. 3 н. и 11 з.п. ф-лы, 13 ил.



ФИГ. 1

RU 2679971 C2

RU 2679971 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 679 971** (13) **C2**

(51) Int. Cl.
G06F 9/44 (2006.01)
G06F 17/30 (2006.01)

(12) ABSTRACT OF INVENTION

(52) CPC
G06F 17/30554 (2018.08); G06F 17/30731 (2018.08); G06F 8/20 (2018.08); G06F 8/33 (2018.08)

(21)(22) Application: **2016147085, 01.06.2015**

(24) Effective date for property rights:
01.06.2015

Registration date:
14.02.2019

Priority:
(30) Convention priority:
02.06.2014 US 62/006,662;
12.11.2014 US 14/539,521

(43) Application published: **01.06.2018** Bull. № **16**

(45) Date of publication: **14.02.2019** Bull. № **5**

(85) Commencement of national phase: **01.12.2016**

(86) PCT application:
US 2015/033554 (01.06.2015)

(87) PCT publication:
WO 2015/187567 (10.12.2015)

Mail address:
129090, Moskva, ul. B.Spasskaya, 25, stroenie 3, OOO "Yuridicheskaya firma Gorodisskij i Partnery"

(72) Inventor(s):
SHAKIRZIANOV, Anton (US),
NARAYANAN, Suriya (US),
YU, Liang (US),
KAMINSKI, Tomasz (US)

(73) Proprietor(s):
MICROSOFT TECHNOLOGY LICENSING, LLC (US)

(54) IMPLEMENTATION OF ACCESS TO SEMANTIC CONTENT IN DEVELOPMENT SYSTEM

(57) Abstract:

FIELD: calculation; counting.

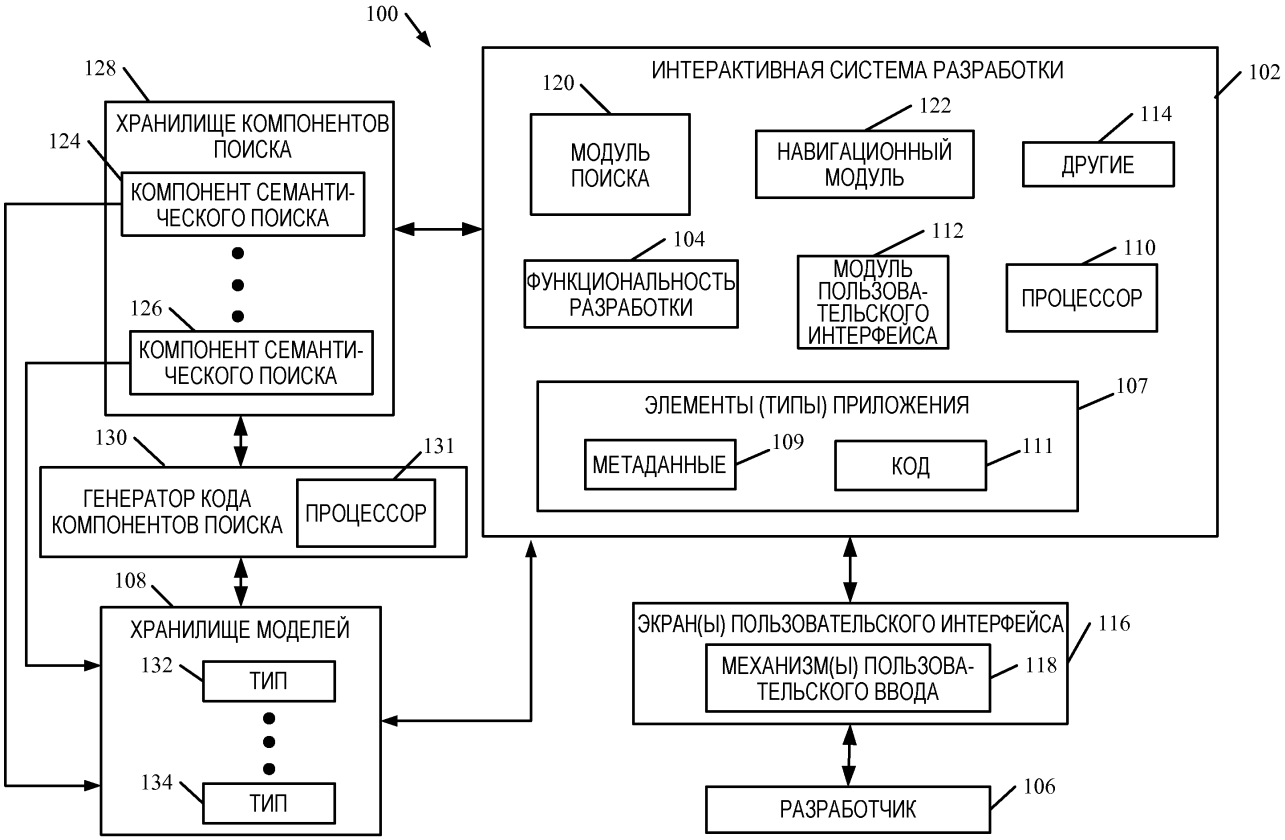
SUBSTANCE: invention relates to the field of software development. Computer system development and search management system, comprising: development module that reads user inputs and converts elements of a computer system, the elements contain types modelled in the computer system, a user interface module that reads user search input received via the user input mechanism and a search engine identifying the search parameter based on the type of the search query and outputting a set of search results in the user interface display, the system additionally contains a search component storage that stores many search components, each search component corresponds to a specified one of the various types of elements mentioned and is designed to search for items of the specified type in the set of properties and methods.

EFFECT: technical result is to simplify the development process.

14 cl, 13 dwg

RU 2679971 C2

RU 2679971 C2



ФИГ. 1

RU 2679971 C2

RU 2679971 C2

Уровень техники

[0001] Компьютерные программы разрабатываются в различных инструментах разработки. Например, многие разработчики программного обеспечения используют интерактивные (или интегрированные) среды разработки (IDE) для того, чтобы разрабатывать программное обеспечение. Разработчики используют IDE для того, чтобы разрабатывать модели типов в компьютерной системе, и для того, чтобы индивидуально настраивать эти модели.

[0002] Примерная интерактивная среда разработки включает в себя множество различных инструментов, так что разработчики могут разрабатывать и тестировать код, который должен быть разработан, и индивидуально настраивать компьютерную систему требуемым образом. В качестве примера, IDE может включать в себя редактор исходного кода, один или более инструментов автоматизации сборки и отладчик, которые обеспечивают возможность программистам разрабатывать программное обеспечение. Некоторые IDE иллюстративно включают в себя компилятор, интерпретатор или и то, и другое. Они могут включать в себя систему управления версиями и различные инструменты, чтобы упрощать конструирование графических пользовательских интерфейсов. Они также могут включать в себя обозреватель классов, обозреватель объектов и схему иерархии классов для использования с разработкой объектно-ориентированного программного обеспечения. Таким образом, разработчики могут использовать IDE для того, чтобы формировать код и метаданные, вместе с индивидуальными настройками кода и метаданных, которые могут быть использованы при разработке системы для использования в данной организации. Например, разработчик может работать с исходным кодом и файлами метаданных, которые связаны с элементами приложения. Одно приложение может требовать создания или изменения как метаданных, так и кода, который использует метаданные различными способами.

[0003] При формировании или индивидуальной настройке программного обеспечения с использованием IDE, разработчик приложений моделирует конкретные понятия (которые могут представляться как типы) в приложении и, в случае необходимости, пишет код. Крупномасштабные приложения, для которых разработчики зачастую используют IDE, могут включать в себя тысячи различных типов.

[0004] В качестве примера, некоторые компьютерные системы включают в себя системы бизнес-операций, к примеру, системы планирования ресурсов предприятия (ERP), системы управления взаимоотношениями с клиентами (CRM), бизнес-системы (LOB), в числе других. Эти типы компьютерных систем зачастую имеют много тысяч различных типов, которые моделируются и индивидуально настраиваются. В качестве примера, только некоторые такие системы бизнес-операций зачастую имеют тысячи различных форм, не говоря уже о многих других типах.

[0005] Системы бизнес-операций не являются единственными типами компьютерных систем, которые имеют большое число типов. Например, игровые системы или широкий спектр других типов систем зачастую также имеют много тысяч различных типов, которые моделируются в программной системе.

[0006] Вышеприведенное пояснение предоставлено просто в качестве общей исходной информации и не имеет намерение использования в качестве помощи при определении объема заявленного изобретения.

Сущность изобретения

[0007] В ходе разработки программного обеспечения, разработчик выполняет поиск элементов для того, чтобы упрощать процесс разработки. Поисковая архитектура

обеспечивает возможность разработчику выполнять поиск метаданных и кода, которые соответствуют определенным критериям. Поисковая архитектура использует информацию семантических элементов для того, чтобы возвращать результаты, которые являются релевантными для запроса разработчика.

[0008] Система разработки содержит, в одном примере, модуль разработки, считывающий пользовательские вводы по разработке и преобразующий элементы компьютерной системы на основе пользовательских вводов по разработке. Элементы содержат типы, моделируемые в компьютерной системе. Модуль пользовательского интерфейса формирует экран пользовательского интерфейса с помощью механизма пользовательского ввода и считывает пользовательский поисковый ввод, принимаемый через механизм пользовательского ввода, указывающий пользовательский поисковый запрос для поиска элементов компьютерной системы. Поисковый механизм идентифицирует параметр поиска на основе типа для пользовательского поискового запроса. Поисковый механизм управляется с возможностью активировать компонент поиска на основе типа на основе параметра поиска на основе типа. Компонент поиска на основе типа выполняет элементный поиск для того, чтобы возвращать набор результатов поиска на экране пользовательского интерфейса.

[0009] Данное краткое изложение сущности изобретения приведено для введения в упрощенной форме подборки концепций, которые дополнительно описаны ниже в подробном описании. Это краткое изложение сущности изобретения не имеет намерение идентифицировать ключевые или важнейшие признаки заявленного изобретения, а также не имеет намерение использоваться в качестве помощи при определении объема заявленного изобретения. Заявленное изобретение не ограничивается реализациями, которые разрешают все недостатки, отмеченные в разделе "Уровень техники".

Краткое описание чертежей

[0010] Фиг. 1 является блок-схемой одного примера архитектуры семантического поиска.

[0011] Фиг. 2 является блок-схемой последовательности операций, иллюстрирующей один пример способа для формирования компонентов семантического поиска.

[0012] Фиг. 3 является блок-схемой, иллюстрирующей функциональность семантического поиска, согласно одному примеру.

[0013] Фиг. 4 является блок-схемой последовательности операций, иллюстрирующей один пример способа для выполнения поиска с использованием компонентов семантического поиска.

[0014] Фиг. 5 иллюстрирует один пример экрана пользовательского интерфейса.

[0015] Фиг. 6 иллюстрирует один пример экрана пользовательского интерфейса.

[0016] Фиг. 7 является блок-схемой, показывающей один пример архитектуры, проиллюстрированной на фиг. 1, развернутой в архитектуре облачных вычислений.

[0017] Фиг. 8-12 показывают различные примеры мобильных устройств, которые могут использоваться с архитектурой, показанной на фиг. 1.

[0018] Фиг. 13 является блок-схемой одного примерного вычислительного окружения.

Подробное описание изобретения

[0019] Фиг. 1 является блок-схемой одного примера архитектуры 100 семантического поиска. Архитектура 100 включает в себя интерактивную систему 102 разработки (например, IDE), имеющую функциональность 104 разработки. Фиг. 1 показывает то, что разработчик 106 взаимодействует с системой 102, чтобы выполнять разработку и/или индивидуальную настройку элементов 107 приложения, которые выполняются в компьютерной системе. Например, каждый из элементов приложения также включает

в себя метаданные 109 и может включать в себя код 111. В качестве примера, разработчик 106 использует функциональность 104 для того, чтобы разрабатывать элементы 107 для приложения, к примеру, посредством создания или изменения метаданных 109 и кода 111. В одном примере, но не в качестве ограничения, элементы 107 содержат объекты в среде объектно-ориентированного программирования. Любой подходящий язык(и) программирования может быть использован в системе 102.

[0020] В проиллюстрированном примере, хранилище 108 моделей сохраняет метаданные и код, соответствующий различным специальным типам элементов приложения (например, типам), и является доступным, например, посредством системы 102 и генератора 130 кода компонентов поиска. "Тип" означает абстракцию, представляющую понятия, моделируемые в системе. Например, в системе бизнес-операций, типы элементов могут включать в себя формы, объекты, классы, таблицы, пункты меню, роли и/или разрешения системы безопасности, помимо прочего. В одном примере, табличные объекты содержат метаданные и код для хранения данных приложения в базе данных. В другом примере, объекты формы содержат метаданные и код, чтобы описывать информационный контент, который должен отображаться на различных устройствах для того, чтобы пользователи приложения использовали информацию и взаимодействовали с приложением.

[0021] В одном примере, при использовании функциональности 104 разработки для того, чтобы разрабатывать элементы 107 приложения, разработчику 106 представляется интегрированное или IDE-представление для кодирования элементов 107 приложения. Один упрощенный пример показан в нижеприведенной таблице 1 для иллюстрации.

Табл. 1

```
public class Table1 extends common
{
        /// <summary>
        ///
        /// </summary>
        private void Method1()
        {
        }

        /// <summary>
        ///
        /// </summary>
        public void insert()
        {
                super();
        }

}
```

[0022] Таким образом, код и метаданные, авторские разрабатываемые разработчиком 106, чтобы разрабатывать элементы 107 приложения, представляются в первом формате, например, в представлении редактора кода, которое предоставляет удобный для пользователя интерфейс для кодирования элементов 107 приложения. Тем не менее, хотя разработчик 106 просматривает и создает авторскую разработку кода и метаданных в первом формате, интерактивная система 102 разработки поддерживает и управляет представлением исходного кода разработанных элементов приложения во втором формате, который отличается от первого формата. В одном примере, преобразованное в последовательную форму представление, содержащее код и метаданные, поддерживается посредством системы 102 для каждого элемента. Второй формат является машиночитаемым и поддается выполнению посредством системы 102. В одном примере, но не в качестве ограничения, хранилище 108 моделей содержит файловую систему, которая сохраняет представления исходного кода в качестве XML-файлов. XML метаданных и кода содержат преобразованные в последовательную форму структуры элементов, каждая из которых имеет собственный тип. Таблица 2 ниже показывает примерный XML-файл, который соответствует интегрированному представлению, показанному в таблице 1:

Табл. 2

```
<?xml version="1.0" encoding="utf-8"?>
<AxTable xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <Name>Table1</Name>
    <SourceCode>
        <Declaration><![CDATA[
public class Table1 extends common
{
}
]]></Declaration>
        <Methods>
            <Method>
                <Name>Method1</Name>
                <Source><![CDATA[
  /// <summary>
  ///
  /// </summary>
  private void Method1()
  {
  }

]]></Source>
            </Method>
            <Method>
                <Name>insert</Name>
                <Source><![CDATA[
  /// <summary>
  ///
  /// </summary>
  public void insert()
  {
    super();
  }
```

```
]]></Source>
                </Method>
            </Methods>
        </SourceCode>
        <Label>@SYS1234</Label>
        <DeleteActions />
        <FieldGroups>
            <AxTableFieldGroup>
                <Name>AutoReport</Name>
                <Fields />
            </AxTableFieldGroup>
            <AxTableFieldGroup>
                <Name>AutoLookup</Name>
                <Fields />
            </AxTableFieldGroup>
            <AxTableFieldGroup>
                <Name>AutoIdentification</Name>
                <AutoPopulate>Yes</AutoPopulate>
                <Fields />
            </AxTableFieldGroup>
            <AxTableFieldGroup>
                <Name>AutoSummary</Name>
                <Fields />
            </AxTableFieldGroup>
            <AxTableFieldGroup>
                <Name>AutoBrowse</Name>
                <Fields />
            </AxTableFieldGroup>
        </FieldGroups>
        <Fields>
            <AxTableField xmlns=""
                i:type="AxTableFieldString">
                <Name>Field1</Name>
            </AxTableField>
```

```
        </Fields>
        <FullTextIndexes />
        <Indexes />
        <Mappings />
        <Relations />
        <StateMachines />
    </AxTable>
```

[0023] В вышеприведенном примере, метаданные и код преобразуются в последовательную форму в один XML-файл. Иными словами, фрагменты кода (т.е. неструктурированные строки) и метаданные (т.е. структурированные наборы свойств и значений) вставляются вперемежку в XML-файл. Тем не менее, специалисты в данной области техники должны понимать, что могут быть использованы другие форматы.

[0024] Разработчик 106 может взаимодействовать с интерактивной системой 102 разработки либо через отдельное устройство разработчика (к примеру, персональный компьютер, планшетный компьютер, другое мобильное устройство и т.д.), либо непосредственно. Разработчик 106 также может взаимодействовать с системой 102 по сети (например, удаленно). Разработчик 106 показан взаимодействующим непосредственно (например, локально) с системой 102 на фиг. 1 только для примера.

[0025] Интерактивная система 102 разработки, в одном примере, включает в себя процессор 110 и модуль 112 пользовательского интерфейса. Модуль 112 пользовательского интерфейса формирует экраны 116 пользовательского интерфейса с помощью механизмов 118 пользовательского ввода для взаимодействия разработчика 106. Разработчик 106 взаимодействует с механизмами 118 пользовательского ввода, чтобы управлять и манипулировать интерактивной системой 102 разработки. В одном примере, разработчик 106 может выполнять это для того, чтобы реализовывать функциональность 104 разработки, а также использовать модуль 120 поиска и навигационный модуль 122. Система 102 также может включать в себя другие элементы 114.

[0026] Разработчик 106 может использовать существующий код и метаданные в хранилище 108 моделей либо формировать новый код и метаданные или комбинацию существующего и нового кода и метаданных. При этом существующие элементы в хранилище 108 моделей могут изменяться, либо удаленные и новые элементы могут добавляться. Чтобы упрощать разработку, разработчик 106 может хотеть поиска в хранилище 108 моделей с тем, чтобы находить интересующие элементы. Например, разработчик 106 может хотеть обнаруживать конкретный элемент, который следует индивидуально настраивать в приложении.

[0027] Тем не менее, частично вследствие размера кодовой базы, которая зачастую является довольно большой, может быть затруднительным находить элементы, которые удовлетворяют определенным критериям поиска разработчика. Одна реализация поиска основывается на компоновке индекса заранее, для которого выполняется запрос разработчика. Например, предусмотрены поисковые роботы, которые осуществляют навигацию по контенту и компонуют индексы, которые затем используются для того, чтобы выполнять поиск. В случае платформы разработки, после того как элемент изменяется или добавляется, индекс становится неактуальным. Дополнительно, с учетом размера кодовой базы, многократная перекомпоновка индекса отнимает большое

количество времени.

[0028] В проиллюстрированном примере, архитектура 100 семантического поиска получает результаты поиска посредством использования модуля 120 поиска, чтобы выполнять семантический поиск модели 108, с учетом имен, типов и/или свойств элементов. Поиск является семантическим в том, он использует понимание структуры типов элементов и смысла типов элементов и свойств в этих типах элементов. Как подробнее пояснено ниже, конкретная структура типов элементов может быть релевантной для поиска элементов хранилища 108 моделей. В качестве иллюстрации, но не в качестве ограничения, каждый тип элемента имеет конкретную структуру свойств, методов и/или вычислений, которые задают поведение во время выполнения для элементов этого типа элементов. Например, тип элемента таблицы может включать в себя имя (например, "таблица клиентов") и набор свойств, которые идентифицируют атрибуты для клиента (например, идентификатор, адрес клиента и т.д.). Кроме того, в этом примере, тип элемента таблицы может включать в себя метод для вычисления значения для клиента и/или метод для отображения значения.

[0029] Перед более подробным описанием общей работы архитектуры 100, предоставляется краткий обзор. В одном примере, модуль 120 поиска содержит поисковый механизм, который принимает пользовательский поисковый запрос, задающий критерии поиска в форме одного или более маркеров. Маркеры задают параметры поиска и могут включать в себя один или более символов, формирующих строку или термин. Поисковый механизм синтаксически анализирует поисковый запрос от разработчика 106, чтобы идентифицировать параметр или ограничение семантического поиска, и выполняет поисковый запрос в хранилище 108 моделей, чтобы получать набор результатов поиска, которые предоставляются разработчику 106. В одном примере, выполнение запроса содержит сопоставление одного или более маркеров со свойствами и/или методами в элементах приложения.

[0030] Параметр семантического поиска может явно предоставляться в самом поисковом запросе или может подразумеваться или извлекаться из поискового запроса. Например, в примере, описанном ниже относительно фиг. 5, разработчик 106 вводит поисковый запрос:

type:table, method name:insert property:"source=crosscompany"

[0031] Здесь, параметр семантического поиска, идентифицированный из запроса, содержит ограничение на основе типа. Иными словами, разработчик 106 хочет элементов, которые имеют тип элемента "table", имеют метод с именем, совпадающим с маркером "insert", и исходное свойство со значением, совпадающим с маркером "crosscompany". Хотя варианты осуществления поясняются в данном документе в контексте ограничений на основе типа, следует отметить, что могут использоваться другие параметры или ограничения семантического поиска.

[0032] В проиллюстрированном примере, чтобы выполнять поиск, модуль 120 поиска осуществляет доступ к хранилищу 128 компонентов поиска, которое сохраняет множество компонентов поиска (т.е. компоненты 124 и 126 поиска), которые сформированы посредством генератора 130 кода компонентов поиска. Один пример формирования компонентов поиска с использованием генератора 130 кода компонентов поиска подробнее поясняется ниже относительно фиг. 2. Кратко, генератор 130 кода компонентов поиска включает в себя процессор 131, выполненный с возможностью формировать компонент поиска для каждого различного типа элемента, моделируемого в хранилище 108 моделей. Каждый компонент поиска формируется для конкретного одного из типов элементов. Таким образом, каждый компонент поиска является

конкретным для структуры конкретного типа элемента, для которого он сформирован. В одном примере, компоненты поиска формируются и сохраняются в хранилище 128 для всех возможных типов элементов, которые могут использоваться разработчиком 106. Например, в одном примере предварительно заданный набор типов элементов доступен разработчику 106, и любые новые типы элементов добавляются в систему 102 посредством обновления системы 102.

[0033] Модуль 120 поиска использует ограничения поиска на основе типа из поискового запроса для того, чтобы идентифицировать один или более компонентов поиска из хранилища 128 компонентов поиска, которые должны использоваться для того, чтобы возвращать список результатов, из элементов в хранилище 108 моделей. Один пример выполнения поиска в хранилище 108 моделей с использованием компонентов поиска подробнее поясняется ниже относительно фиг. 4. Кратко, модуль 120 поиска идентифицирует соответствующий компонент поиска для каждого ограничения поиска на основе типа. В вышеприведенном примере, модуль 120 поиска идентифицирует компонент поиска (т.е. компонент 124 или 126 поиска), который сформирован для типа элемента таблицы. Экземпляр идентифицированного компонента поиска создается для каждого элемента в хранилище 108 моделей, имеющем тип элемента таблицы, чтобы идентифицировать элементы, которые совпадают с именем метода и значениями свойств в поисковом запросе. Модуль 120 поиска агрегирует результаты поиска, полученные из компонента поиска, экземпляр которого создан. Навигационный модуль 122 упрощает навигацию пользователя по результатам поиска.

[0034] Поисковая архитектура 100 в силу этого использует семантическую информацию относительно элементов 107 приложения при выполнении поиска в хранилище 108 моделей, без необходимости компоновать или поддерживать индекс заранее. Это позволяет снижать нагрузку по обработке и требования по времени и к запоминающему устройству при выполнении функциональности поиска в системе разработки и позволяет улучшать релевантность результатов поиска относительно запроса пользователя.

[0035] Для иллюстрации, в примере по фиг. 1, для каждого отличного от других типа элемента приложения архитектура 100 поддерживает конкретные компоненты поиска, которые выполнены с возможностью осуществлять поиск существующих элементов хранилища 108 моделей для этого типа элемента. Тем не менее, эти компоненты поиска также позволяют выполнять поиск любых новых элементов, добавленных разработчиком 106 в хранилище 108 моделей, независимо от типа (т.е. все типы элементов имеют предварительно заданный компонент поиска) или конкретных свойств нового элемента. С другой стороны, в случае проиндексированной поисковой системы, добавление новых элементов в хранилище 108 моделей должно требовать обновления индекса таким образом, что он включает в себя новые элементы.

[0036] Для дополнительной иллюстрации, допустим, что хранилище 108 моделей включает в себя два различных типа элементов (т.е. тип 132 элемента таблицы и тип 134 элемента формы). Первый компонент 124 поиска формируется для типа 132 элемента, а второй компонент 126 поиска формируется для типа 134 элемента. В примере по фиг. 1, генератор 130 кода должен выполняться только один раз для каждого типа элемента. Таким образом, как только компоненты 124 и 126 поиска сформированы, генератор 130 кода не должен повторно формировать или модифицировать их, даже если существующие элементы типов 132 и 134 модифицируются в хранилище 108 моделей, и/или новые элементы типов 132 и 134 добавляются в хранилище 108 моделей.

[0037] Экземпляр компонента 124 поиска создается, когда модуль 120 поиска

выполняет поиск элементов типа 132, а экземпляр компонента 126 поиска создается, когда модуль 120 поиска выполняет поиск элементов типа 134. В одном примере, когда выполняется поиск обоих типов 132 и 134 элементов, компоненты 124 и 126 поиска могут работать параллельно, чтобы уменьшать время поиска. Следует отметить, что хотя только два типа элементов и компонента поиска на основе типа показаны на фиг 1, любое число типов элементов и компонентов семантического поиска может реализовываться.

[0038] Хотя хранилище 108 моделей и хранилище 128 компонентов поиска проиллюстрированы на фиг. 1 как отдельные от интерактивной системы 102 разработки, следует отметить, что хранилище 108 моделей и/или хранилище 128 компонентов поиска могут быть частью интерактивной системы 102 разработки. Тем не менее, вследствие соображений полосы пропускания и времени задержки, в некоторых реализациях хранилище 108 моделей и хранилище 128 компонентов поиска могут поддерживаться в идентичной вычислительной системе, хотя это представляет собой только один пример. Таким образом, хотя запросы и результаты поиска могут отправляться по сети, поисковая архитектура 100 не требует передачи хранилища 108 моделей. С другой стороны, это представляет собой только один пример архитектуры.

[0039] Кроме того, фиг. 1 показывает множество различных функциональных блоков. Следует отметить, что блоки могут консолидироваться, так что дополнительная функциональность выполняется посредством каждого блока, либо они могут разделяться, так что функциональность дополнительно распределяется.

[0040] Также следует отметить, что вышеприведенное пояснение показывает ряд хранилищ данных, включающих в себя хранилище 108 моделей и хранилище 128 компонентов поиска. Хотя они показаны как два независимых хранилища данных, они также могут формироваться в одном хранилище данных. Помимо этого, данные в этих хранилищах данных также могут сохраняться в нескольких дополнительных хранилищах данных. Кроме того, хранилища данных могут быть локальными для окружений, агентов, модулей и/или компонентов, которые осуществляют доступ к ним, либо они могут быть удаленными относительно них и доступными посредством этих окружений, агентов, модулей и/или компонентов. Аналогично, некоторые могут быть локальными, тогда как другие являются удаленными.

[0041] В проиллюстрированном примере, процессоры 110 и 131 содержат процессоры компьютера с ассоциированным запоминающим устройством и синхронизирующей схемой (не показаны отдельно). Они представляют собой функциональный компонент агента или окружения, которому они принадлежат, и иллюстративно активируются и упрощают функциональность других элементов в этом окружении или агенте.

[0042] Фиг. 2 является блок-схемой последовательности операций, иллюстрирующей один пример способа 200 для формирования компонентов семантического поиска. Для иллюстрации, но не в качестве ограничения, способ 200 описывается в контексте архитектуры 100, формирующей компоненты поиска на основе типа.

[0043] Способ 200 может инициироваться периодически и/или в ответ на условие или событие. Например, способ 200 может инициироваться в ответ на обновление системы 102, которое добавляет или модифицирует типы элементов, которые поддерживаются посредством системы 102. В другом примере, способ 200 может инициироваться в ответ на ввод от разработчика 106 (например, посредством выбора элемента управления, такого как открытие, закрытие, сохранение и т.д., в пользовательском интерфейсе 116).

[0044] На этапе 202, генератор 130 кода компонентов поиска осуществляет доступ к хранилищу 108 моделей и определяет, на этапе 204, то, имеются или нет какие-либо

новые типы элементов, для которых можно формировать компонент поиска на основе типа. В одном примере, генератор 130 кода компонентов поиска анализирует некоторые (например, последние изменения и добавления) или все элементы в хранилище 108 моделей и сравнивает эти элементы с существующими или известными типами элементов (т.е. типами 132 и 134). Например, генератор 130 кода компонентов поиска идентифицирует элементы, которые изменены или добавлены разработчиком 106.

[0045] Если новый тип элемента идентифицируется, генератор 130 кода компонентов поиска анализирует структуру нового типа элемента на этапе 206, чтобы формировать компонент поиска на основе типа для нового типа элемента на этапе 208. В одном примере, генератор 130 кода компонентов поиска синтаксически анализирует структуру нового типа элемента на любые подтипы и определяет то, какие свойства содержат тип и/или подтипы, любые типы дочерних элементов, какие типы элементов вытекают из типа элемента, и реализацию для методов-получателей свойств типа элемента. Каждый метод-получатель свойств задает функцию для извлечения свойства типа элемента, например, на основе местоположения свойства в типе элемента и/или взаимосвязей с другими свойствами. В одном примере, генератор 130 кода компонентов поиска формирует различный код методов-получателей свойств, чтобы выполнять поиск в других частях структуры типов элементов. Например, один фрагмент кода может выполнять поиск методов в данной части элемента, и один фрагмент кода может проверять элементы управления и т.д. Относительно примера типа элемента таблицы клиентов, поясненного выше, один метод-получатель свойств может быть выполнен с возможностью возвращать свойство "customer ID"", и другой метод-получатель свойств может быть выполнен с возможностью возвращать свойство "address".

[0046] Каждый компонент поиска выполнен с возможностью соответствовать заданному шаблону элементов (например, шаблону дочерних элементов, свойств, методов и т.д.), который основан на типе элемента, для которого формируется компонент поиска. Например, но не в качестве ограничения, на фиг. 1, типы 132 и 134 элементов имеют отличающиеся друг от друга шаблоны дочерних элементов. Компонент 124 поиска выполнен с возможностью вызывать метод(ы) поиска, чтобы анализировать и возвращать значения дочерних элементов, ассоциированных с типом 132 элемента, и компонент 126 поиска выполнен с возможностью вызывать метод(ы) поиска, чтобы анализировать и возвращать значения дочерних элементов, ассоциированных с типом 134 элемента.

[0047] В качестве примера, один элемент метаданных содержит древовидную структуру данных и задается посредством имени и типа элемента метаданных. Тип элемента метаданных дополнительно задается посредством набора свойств, причем каждое свойство задается посредством имени и типа значения свойства. Тип значения свойства, например, может представлять собой, но не в качестве ограничения, примитив (конвертируемый в строку ("Да/нет", "Дата", "Теги" и т.д.)). Такое свойство упоминается в качестве "простого свойства". Другой тип значения свойства является типом элемента метаданных, содержащего дочерние элементы метаданных. Такое свойство может упоминаться в качестве "узлового свойства". Корневые элементы метаданных представляют собой элементы, которые сохраняются непосредственно в хранилище метаданных и не имеют родительских элементов. Дочерние элементы метаданных представляют собой элементы, которые содержатся в части другого узлового свойства элемента. Путь к метаданным содержит строку, которая уникально идентифицирует элемент метаданных и упрощает определение местоположения элемента метаданных. В одном примере, форма пути следующая:

dynamics://<Root_type>/<Root_element_name>
[/<Subtype_1>/<Subelement_name_1>
[/<Subtype_2>/<Subelement_name_2>
[...]]] где:
<Root_type> - тип корневого элемента метаданных
<Root_element_name> - имя корневого элемента
<Subtype_i> - типы каждого дочернего элемента метаданных в дереве
<Subelement_name_i> - имена каждого дочернего элемента метаданных в дереве

[0048] На этапе 210, сформированный компонент семантического поиска сохраняется в хранилище 128 компонентов поиска. Если какие-либо дополнительные новые типы элементов идентифицируются на этапе 212, этапы 206, 208 и 210 повторяются для нового типа(ов) элемента.

[0049] Фиг. 3 является блок-схемой, иллюстрирующей функциональность семантического поиска, согласно одному примеру. Для иллюстрации, но не в качестве ограничения, фиг. 3 описывается в контексте функциональности семантического поиска в архитектуре 100.

[0050] Блок 250 предоставляет интерфейс с интерактивной системой 102 разработки. Через блок 250, модуль 120 поиска принимает поисковый запрос, который предоставляется в синтаксический анализатор запросов в блоке 252. Запрос предоставляет один или более критериев поиска, которые задают фильтр(ы), и может иметь любой подходящий синтаксис или грамматику.

[0051] Один относительно простой пример синтаксиса предоставляется ниже. Поисковый запрос представляет собой *search_string,* где:

search_string = empty_string

search_string = text_without_colon

search_string = filter

search_string = search_string filter

filter = filter_name:filter_value

filter_value = text_without_comma

filter_value = "any_text"

filter_value = filter_value,filter_value

filter_name = name OR type OR model OR property

Таким образом, строка поиска состоит из набора фильтров в общей форме:

<filter_1>:<filter_1_value>[<filter_2>:<filter_2_value>...[

<filter_N>:<filter_N_value>]]

При этом <filter_i> является одним из допустимых имен фильтров, и <filter_i_value> являются разделенными запятыми и возможно заключенными в кавычки значениями фильтрации.

[0052] Как проиллюстрировано выше и показано на фиг. 3, один пример критериев поиска пользователя представляет собой имя элемента, которое может указывать одну строку или набор строк. Считается, что элемент удовлетворяет этим критериям, если имя элемента содержит, по меньшей мере, одну из строк. Каждое разделенное запятыми значение может быть допустимым именем элемента. В одном примере, имя элемента

представляет собой фильтр по умолчанию. Таким образом, если поисковый запрос включает в себя один маркер, он предполагается в качестве имени элемента. В этом примере, если ограничение на основе типа не идентифицируется, поисковая архитектура может создавать экземпляр компонентов поиска для всех доступных типов элементов.

[0053] Другие примерные критерии представляют собой тип элемента, который может указывать один тип элемента или набор типов элементов. Считается, что элемент удовлетворяет этим критериям, если он имеет один из указанных типов. Каждое разделенное запятыми значение может представлять собой имя одного из типов элементов (т.е. таблица, класс, поле). Поисковый запрос может указывать как корень, так и подтипы в качестве значения. В одном примере, логика фильтрации может заключаться в следующем:

(roottype_1 OR roottype_2 OR... OR roottype_N) AND (subtype_1 OR subtype_2 OR... OR subtype_N)

[0054] Другие примерные критерии представляют собой свойство элемента, которое может указывать набор пар ключ/значение "имя свойства - значение свойства". Считается, что элемент удовлетворяет критериям, если для каждой пары следует признавать, что a) элемент содержит "простое" свойство с указанным именем, и b) значение этого свойства, преобразованное в строку, содержит указанное значение. Каждое разделенное запятыми значение может иметь форму property_name=property_value.

[0055] В блоке 254, экземпляры одного или более компонентов поиска на основе типа (например, компонента 124 и/или 126) создаются на основе идентифицированного типа(ов) элемента. Например, это может выполняться посредством осуществления доступа к информации типа (например, из хранилища 128 компонентов поиска на основе типа) в блоке 256 на основе критериев фильтрации типов из блока 252 синтаксического анализатора. Блок 256 предоставляет информацию относительно типа (ов) элемента, включающую в себя, но не только, то, какие свойства содержит тип(ы), типы дочерних элементов для типа элемента, какие типы элементов извлекаются из типа элемента, и реализацию методов-получателей свойств для типа элемента.

[0056] В одном примере, блок 254 использует информацию типа, предоставленную посредством блока 256, чтобы обрабатывать параметры поиска, чтобы предоставлять критерии типов с критериями свойства. Если критерии поиска включают в себя одно или более свойств, использование блока 254 информации типа может отфильтровывать все типы элементов, которые не могут содержать искомые свойства.

[0057] Для каждого типа элемента, поиск которого должен быть выполнен, экземпляр соответствующего компонента поиска на основе типа создается в соответствии с кодом, сформированным посредством генератора 130 кода.

[0058] В блоке 258, ссылки на элементы в хранилище 108 моделей получаются согласно критериям поиска из блока 254 и компонентам семантического поиска, экземпляры которых созданы в блоке 256. Например, ссылки элемента метаданных могут упрощать получение имени корневого элемента (быструю операцию, которая не связана с обращением к запоминающему устройству), и/или загрузку элемента (относительно длительную операцию, связанную с обращением к запоминающему устройству).

[0059] В блоке 260, ссылки на элемент, полученные в блоке 258, приоритезированы в порции, например, на основе заданных критериев имени элемента или другой эвристики. Например, корневой элемент, имеющий имя, которое содержит любое из искомых имен, должен обрабатываться до корневых элементов, которые не содержат имя.

[0060] Блок 262 обрабатывает конкретные элементы в хранилище 108 моделей, чтобы определять то, удовлетворяют они или нет критериям поиска. В одном примере, считается, что элемент удовлетворяет критериям поиска, если тип элемента представляет собой один из требуемых типов, указываемых в блоке 254, имя элемента содержит одно из требуемых имен или его часть, и для каждой пары "имя свойства - значение свойства", указываемой посредством критериев поиска, следует признавать, что элемент содержит свойство с таким именем, и значение этого свойства содержит значение указанного свойства.

[0061] В одном примере, блок 262 получает функцию предиката элемента или другую информацию из блока 264, который используется для того, чтобы определять то, удовлетворяет элемент или нет критериям поиска. Блок 264 создает функцию предиката для каждого из типов элементов, предоставленных посредством 262, с использованием информации из блока 256. Например, блок 264 предоставляет типы элементов в блок 256 и принимает информацию относительно реализации методов-получателей свойств для каждого типа элемента.

[0062] Если элемент в хранилище 108 моделей удовлетворяет критериям поиска, результат предоставляется разработчику 106 через интерфейсный блок 250.

[0063] Фиг. 4 является блок-схемой последовательности операций, иллюстрирующей один пример способа 300 для выполнения поиска с использованием компонентов семантического поиска. Для иллюстрации, но не в качестве ограничения, способ 300 описывается в контексте архитектуры 100, выполняющей поиск с использованием компонентов поиска на основе типа.

[0064] На этапе 302, поверхность разработки отображается, например, с использованием экрана 116 пользовательского интерфейса. На этапе 304, поисковый ввод принимается, и на этапе 306, поисковый ввод синтаксически анализируется, чтобы идентифицировать критерии поиска. Примеры критериев поиска включают в себя, но не только, ограничения на основе типа, имена методов и значения свойств. Модуль 120 поиска затем выполняет поиск в хранилище 108 моделей элементов, которые удовлетворяют критериям поиска.

[0065] На этапе 308, один или более компонентов поиска на основе типа идентифицируются, и их экземпляры создаются, чтобы выполнять поиск в хранилище 108 моделей. Например, как пояснено выше относительно фиг. 3, ограничение поиска на основе типа может быть явно задано в поисковом вводе. В другом примере, ограничение поиска на основе типа может логически выводиться из маркеров, предоставляемых в поисковом вводе. Например, для значения свойства, предоставленного в поисковом вводе, этап 308 может определять то, какие типы элементов имеют соответствующее свойство.

[0066] Затем экземпляры одного или более компонентов поиска на основе типа создаются посредством модуля 120 поиска, чтобы выполнять поиск элементов в хранилище 108 моделей на основе поискового запроса. В одном примере, отдельный экземпляр компонента поиска создается для каждого элемента с соответствующим типом элемента.

[0067] Компонент(ы) поиска, экземпляр которого создан, используется для того, чтобы выполнять поиск элементов в хранилище 108 моделей на этапе 310 и, на этапе 312, идентифицировать элементы, которые соответствуют критериям, идентифицированным из этапа 306. Как пояснено выше, в одном примере, компоненты поиска могут выполнять поиск преобразованных в последовательную форму представлений (например, XML-файлов) элементов, вместо непосредственного поиска

элементов, разрабатываемых разработчиком 106.

[0068] В качестве примера, но не в качестве ограничения, при поиске преобразованного в последовательную форму представления элемента в хранилище 108 моделей, компонент поиска идентифицирует часть элемента, которая удовлетворяет критериям поиска, посредством нахождения ссылок (например, позиций номеров строк и столбцов) на соответствующие элементы в преобразованных в последовательную форму представлениях. Компонент поиска может отличать код от метаданных и, для совпадения, идентифицированного в преобразованном в последовательную форму представлении, вычисляет позицию в коде, как если выполняется поиск в интегрированного представления кода, которое представляется разработчику. Таким образом, с точки зрения разработчика 106, модуль 120 поиска выполняет поиск и возвращает результаты в представлениях редактора кода и/или редактора метаданных, а не преобразованном в последовательную форму представлении.

[0069] В одном примере, компонент поиска считывает элемент, преобразует его в объектно-ориентированное выражение и применяет его методы-получатели свойств, чтобы идентифицировать и сопоставлять свойство с критериями поиска на основе свойств из поискового запроса. Компонент поиска преобразует совпадающее свойство в соответствующий путь, который уникально идентифицирует свойство в объекте. Например, путь содержит универсальный идентификатор ресурса (URI), который может представлять собой путь к метаданным, как пояснено выше.

[0070] На этапе 314, результаты возвращаются в качестве набора ссылок, которые указывают совпадения элементов с критериями поиска. Например, компонент поиска идентифицирует совпадение элемента посредством возврата соответствующего URI в компонент агрегатора модуля 120 поиска. URI агрегатора предоставляется в модуль 112 пользовательского интерфейса для представления разработчику 106.

[0071] На этапе 316, выбор разработчиком 106 конкретного URI принимается, например, через пользовательское взаимодействие, к примеру, щелчок кнопкой мыши или другой пользовательский ввод. Навигационный модуль 122 декодирует выбранный URI, чтобы идентифицировать соответствующее местоположение элемента. В одном примере, URI содержит ссылку на отличительное свойство (например, "source= crosscompany"), при котором выбор URI открывает редактор метаданных в местоположении, идентифицированном посредством URI. В другом примере, URI содержит ссылку на тело метода, которое включает в себя значение, при котором выбор URI открывает редактор кода.

[0072] В одном примере, результаты поиска получаются и отображаются асинхронно. Это представлено на фиг. 4 посредством стрелки 320. Иными словами, когда компоненты семантического поиска, экземпляры которых созданы, идентифицируют элемент, который удовлетворяет критериям поиска на этапе 312, URI для идентифицированного элемента отображается разработчику, тогда как поиск продолжается в фоновом режиме.

[0073] Фиг. 5 иллюстрирует один пример экрана 400 пользовательского интерфейса, который предоставляет поверхность разработки, через которую разработчик 106 может разрабатывать элементы приложения и выполнять поиск с использованием архитектуры 100. Для иллюстрации, но не в качестве ограничения, экран 400 пользовательского интерфейса описывается в контексте архитектуры 100.

[0074] Экран 400 пользовательского интерфейса включает в себя представление 402 редактора кода, которое принимает вводы разработчика, чтобы создавать авторскую разработку элементов 107 приложения, и интерфейс 404 семантического поиска, который принимает поисковый запрос разработчика. В качестве примера, следующий поисковый

запрос введен в элемент 404:

type:table, method name:insert property:"source=crosscompany"

[0075] С использованием примерного синтаксиса, описанного относительно фиг. 3, поисковый запрос указывает фильтр типа "table", фильтр имени метода "insert" и фильтр имени свойства "crosscompany". Модуль 120 поиска создает экземпляр компонента поиска на основе типа, соответствующего типу таблицы. Поисковый запрос может выполняться асинхронно, что заполняет окно результатов 406 с URI результатов поиска по мере того, как они получаются. Иными словами, поиск может начинаться посредством отображения одного или более URI результата поиска в окне 406 и затем добавлять дополнительные URI результатов поиска в окно 406 по мере того, как они получаются. Таким образом, разработчик 106 может продолжать взаимодействовать с экраном 400 пользовательского интерфейса, например, посредством щелчка требуемого URI, направлять представление 402 в соответствующий результат поиска, тогда как поиск продолжает работать в фоновом режиме, чтобы возвращать все дополнительные результаты. В проиллюстрированном примере, каждый URI включает в себя информацию 408 метки и информацию 410 местоположения, которая идентифицирует элемент и местоположение элемента.

[0076] В одном примере, возможности поиска модуля 120 поиска доступны в качестве интерфейса прикладного программирования (API) вместе с объектной моделью, которая является независимой от синтаксиса поисковых запросов. С использованием API, операция поиска может активироваться в качестве услуги в сети, которая может быть использована удаленно из любого из множества различных устройств (например, согласно правам и безопасности доступа). Параметры для поискового API представляют собой объекты в объектной модели, а не в строке запроса, которые следует подтверждать для синтаксиса. Таким образом, синтаксис поисковых запросов не связан с искателем.

[0077] В качестве примера, схема взаимосвязи классов для объектной модели может включать в себя множество различных классов, при этом каждый класс задает одно или более ограничений семантического поиска и методов, которые должны вызываться для поиска и анализа соответствующих элементов. Примеры ограничений семантического поиска, заданных посредством классов объектных моделей, включают в себя, но не только, ограничения по типам, ограничения по свойствам, ограничения по коду и ограничения по именам.

[0078] Фиг. 6 иллюстрирует примерный пользовательский интерфейс 450, который подготавливает посредством рендеринга результаты поиска с использованием поискового API. Пользовательский интерфейс 450 включает в себя поле 452 ввода запроса, которое принимает поисковый запрос, задающий параметры поиска, и поле 454 результатов запроса, которое отображает соответствующие результаты запроса, возвращаемые из модуля поиска. В проиллюстрированном примере, параметры поиска включают в себя класс ограничений по коду и идентифицируют строку (т.е. "while select") для ограничения по коду. Класс ограничений по коду включает в себя методы для сопоставления строки, приоритезации поиска и т.д. Модуль поиска создает экземпляр объекта класса ограничений по коду и выполняет поиск в хранилище моделей.

[0079] В одном примере, различный синтаксис может предоставляться в зависимости от устройства, из которого инициируется поиск. Например, из настольного компьютера разработчика с экраном с большим форм-фактором, разработчику может разрешаться вводить строку запроса в формальном синтаксисе. С другой стороны, из мобильного устройства с меньшим форм-фактором, ввод в формальном синтаксисе может быть более затруднительным для разработчика. Поисковая архитектура может быть

выполнена с возможностью упрощать ввод запроса в более простой форме. Например, при использовании мобильного устройства и т.п., разработчику могут представляться элементы управления, имеющими предварительно заданные функции поиска, такие как кнопка, назначаемая конкретному набору ограничений поиска (например, поиску на основе конкретного типа).

[0080] Настоящее пояснение упоминает процессоры и серверы. В одном примере, процессоры и серверы включают в себя процессоры компьютера с ассоциированным запоминающим устройством и синхронизирующей схемой (не показаны отдельно). Они представляют собой функциональные компоненты систем или устройств, которым они принадлежат и активируются, и упрощают функциональность других модулей, компонентов и/или элементов в этих системах.

[0081] Кроме того, пояснен ряд экранов пользовательского интерфейса. Они могут принимать широкий спектр различных форм и могут иметь широкий спектр различных активируемых пользователем механизмов ввода, расположенных в них. Например, активируемые пользователем механизмы ввода могут представлять собой текстовые поля, флажки, значки, ссылки, раскрывающиеся меню, поля поиска и т.д. Они также могут активироваться множеством различных способов. Например, они могут активироваться с использованием устройства на основе принципа "указать и щелкнуть" (к примеру, шарового манипулятора или мыши). Они могут активироваться с использованием аппаратных кнопок, переключателей, джойстика или клавиатуры, ползунковых переключателей или пальцевых панелей и т.д. Они также могут активироваться с использованием виртуальной клавиатуры или других виртуальных приводов. Помимо этого, если экран, на котором они отображаются, представляет собой сенсорный экран, они могут активироваться с использованием жестов касания. Кроме того, если устройство, которое отображает их, имеет компоненты распознавания речи, они могут активироваться с использованием речевых команд.

[0082] Также пояснен ряд хранилищ данных. Следует отметить, что они могут разбиваться на несколько хранилищ данных. Все они могут быть локальными для систем, осуществляющих доступ к ним, все они могут быть удаленными, либо некоторые могут быть локальными, тогда как другие являются удаленными. Все эти конфигурации предполагаются в данном документе.

[0083] Кроме того, чертежи показывают ряд этапов с функциональностью, приписанной каждому этапу. Следует отметить, что меньшее число этапов может использоваться, так что функциональность выполняется посредством меньшего числа компонентов. Кроме того, большее число этапов может использоваться, при этом функциональность распределяется между большим числом компонентов.

[0084] Фиг. 7 является блок-схемой архитектуры 100, показанной на фиг. 1, за исключением того, что ее элементы располагаются в архитектуре 500 облачных вычислений. Облачные вычисления предоставляют услуги вычислений, программные услуги, услуги доступа и хранения данных, которые не требуют знания конечным пользователем физического местоположения или конфигурации системы, которая предоставляет услуги. В различных примерах, облачные вычисления предоставляют услуги по глобальной вычислительной сети, к примеру, по Интернету, с использованием надлежащих протоколов. Например, поставщики услуг облачных вычислений доставляют приложения по глобальной вычислительной сети, и к ним может осуществляться доступ через веб-обозреватель или любой другой вычислительный компонент. Программное обеспечение, модули или компоненты архитектуры 100, а также соответствующие данные могут сохраняться на серверах в удаленном

местоположении. Вычислительные ресурсы в облачном вычислительном окружении могут консолидироваться в удаленном местоположении центра обработки и хранения данных, либо они могут распределяться. Инфраструктуры облачных вычислений могут предоставлять услуги через центры обработки и хранения совместно используемых данных, даже если они выглядят как одна точка доступа для пользователя. Таким образом, модули, компоненты и функции, описанные в данном документе, могут предоставляться от поставщика услуг в удаленном местоположении с использованием архитектуры облачных вычислений. Альтернативно, они могут предоставляться из традиционного сервера, или они могут быть установлены непосредственно на клиентских устройствах, или другими способами.

[0085] Описание имеет намерение включать в себя и открытые облачные вычисления и закрытые облачные вычисления. Облачные вычисления (открытые и закрытые) предоставляют практически прозрачное объединение ресурсов, а также меньшую потребность управлять и конфигурировать базовую аппаратную инфраструктуру.

[0086] Открытое облако управляется посредством поставщика и типично поддерживает несколько клиентов с использованием идентичной инфраструктуры. Кроме того, открытое облако, в отличие от закрытого облака, может освобождать конечных пользователей от управления аппаратными средствами. Закрытое облако может управляться посредством самой организации, и инфраструктура типично не используется совместно с другими организациями. Организация по-прежнему поддерживает аппаратные средства в некоторой степени, к примеру, установки и ремонт и т.д.

[0087] В примере, показанном на фиг. 7, некоторые элементы являются аналогичными элементам, показанным на фиг. 1, и они аналогично пронумерованы. Фиг. 7, в частности, показывает то, что интерактивная система 102 разработки, хранилище 108 моделей, хранилище 128 компонентов поиска и генератор 130 кода компонентов поиска могут быть расположены в облаке 502 (которое может быть открытым, закрытым или комбинацией, в которой части являются открытыми, тогда как другие являются закрытыми). Следовательно, разработчик 106 использует пользовательское устройство 504, чтобы осуществлять доступ к этим системам через облако 502.

[0088] Фиг. 7 также иллюстрирует другой пример облачной архитектуры. Фиг. 7 показывает то, что также предполагается, что некоторые элементы архитектуры 100 могут располагаться в облаке 502, тогда как другие не располагаются. В качестве примера, хранилище 108 моделей может располагаться за пределами облака 502 и быть доступным через облако 502. В другом примере, хранилище 128 компонентов поиска также может находиться за пределами облака 502. В другом примере, генератор 130 кода компонентов поиска также может находиться за пределами облака 502. Независимо от того, где они расположены, к ним может осуществляться доступ непосредственно посредством устройства 504 через сеть (глобальную вычислительную сеть или локальную вычислительную сеть), они могут хоститься на удаленном веб-узле посредством услуги, или они могут предоставляться в качестве услуги через облако или быть доступными посредством услуги соединения, которая постоянно размещается в облаке. Все эти архитектуры предполагаются в данном документе.

[0089] Также следует отметить, что архитектура 100 или ее части может располагаться в широком спектре различных устройств. Некоторые из этих устройств включают в себя серверы, настольные компьютеры, переносные компьютеры, планшетные компьютеры или другие мобильные устройства, такие как карманные компьютеры, сотовые телефоны, смартфоны, мультимедийные проигрыватели, персональные

цифровые устройства и т.д.

[0090] Фиг. 8 является упрощенной блок-схемой одного примера карманного или мобильного вычислительного устройства, которое может использоваться в качестве переносного устройства 16 пользователя или клиента, в котором может развертываться настоящая система (или ее части). Фиг. 9-12 являются примерами карманных или мобильных устройств.

[0091] Фиг. 8 предоставляет общую блок-схему компонентов клиентского устройства 16, которое может запускать модули или компоненты архитектуры 100 либо которое взаимодействует с архитектурой 100, либо которое выполняет и то, и другое. В устройстве 16, предоставляется линия 13 связи, которая обеспечивает возможность карманному устройству обмениваться данными с другими вычислительными устройствами и в некоторых примерах предоставляет канал для автоматического приема информации, к примеру, посредством сканирования. Примеры линии 13 связи включают в себя инфракрасный порт, последовательный/USB-порт, кабельный сетевой порт, к примеру, Ethernet-порт и беспроводной сетевой порт, обеспечивающий связь через один или более протоколов связи, включающих в себя общую службу пакетной радиопередачи (GPRS), LTE, HSPA, HSPA+ и другие протоколы 3G-и 4G-радиосвязи, 1Xrtt и службу коротких сообщений, которые являются беспроводными службами, используемыми для того, чтобы предоставлять сотовый доступ к сети, а также протоколы 802.11 и 802.11b (Wi-Fi) и протокол Bluetooth, которые предоставляют локальные беспроводные подключения к сетям.

[0092] В других примерах, приложения или системы могут приниматься на съемной карте по стандарту Secure Digital (SD), которая соединяется с интерфейсом 15 SD-карты. Интерфейс 15 SD-карты и линии 13 связи обмениваются данными с процессором 17 (который также может осуществлять процессоры 110 из фиг. 1) по шине 19, которая также соединяется с запоминающим устройством 21 и компонентами 23 ввода-вывода, а также синхросигналом 25 и системой 27 определения местоположения.

[0093] Компоненты 23 ввода-вывода, в одном примере, предоставляются для того, чтобы упрощать операции ввода и вывода. Компоненты 23 ввода-вывода для различных примеров устройства 16 могут включать в себя компоненты ввода, такие как кнопки, датчики касания, датчики мультикасания, оптические или видеодатчики, речевые датчики, сенсорные экраны, бесконтактные датчики, микрофоны, датчики наклона и гравитационные переключатели, и компоненты вывода, такие как устройство отображения, динамик и или порт принтера. Также могут использоваться другие компоненты 23 ввода-вывода.

[0094] Синхросигнал 25 содержит компонент синхросигнала реального времени, который выводит время и дату. Он также может предоставлять функции синхронизации для процессора 17.

[0095] Система 27 определения местоположения включает в себя компонент, который выводит текущее географическое местоположение устройства 16. Она может включать в себя, например, приемное устройство на основе глобальной системы позиционирования (GPS), LORAN-систему, систему счисления пути, систему сотовой триангуляции или другую систему позиционирования. Она также может включать в себя, например, картографическое программное обеспечение или навигационное программное обеспечение, которое формирует требуемые карты, навигационные маршруты и другие географические функции.

[0096] Запоминающее устройство 21 сохраняет операционную систему 29, сетевые настройки 31, приложения 33, конфигурационные настройки 35 приложений, хранилище

37 данных, драйверы 39 связи и конфигурационные настройки 41 связи. Она также может сохранять клиентскую систему 24, которая может представлять собой часть или всю архитектуру 100. Запоминающее устройство 21 может включать в себя все типы материальных энергозависимых и энергонезависимых машиночитаемых запоминающих устройств. Оно также может включать в себя компьютерные носители хранения данных (описаны ниже). Запоминающее устройство 21 сохраняет машиночитаемые инструкции, которые, при выполнении посредством процессора 17, инструктируют процессору выполнять машинореализованные этапы или функции согласно инструкциям. Процессор 17 может активироваться посредством других модулей компонентов, чтобы также упрощать их функциональность.

[0097] Примеры сетевых настроек 31 включают в себя, к примеру, информацию прокси-сервера, информацию Интернет-подключений и привязки. Конфигурационные настройки 35 приложений включают в себя настройки, которые индивидуально адаптируют приложение для конкретной организации или пользователя. Конфигурационные настройки 41 связи предоставляют параметры для обмена данными с другими компьютерами и включают в себя такие элементы, как GPRS-параметры, SMS-параметры, имена пользователей и пароли для подключения.

[0098] Приложения 33 могут представлять собой приложения, которые ранее сохранены на устройстве 16, или приложения, которые устанавливаются в ходе использования, хотя также они могут быть частью операционной системы 29 или хоститься внешне по отношению к устройству 16.

[0099] Фиг. 9 показывает один пример, в котором устройство 16 представляет собой планшетный компьютер 600.

На фиг. 9, компьютер 600 показан с экраном 602 пользовательского интерфейса. Экран 602 может представлять собой сенсорный экран (так что жесты касания из пальца пользователя могут использоваться для того, чтобы взаимодействовать с приложением) или интерфейс с поддержкой перьевого ввода, который принимает вводы из пера или стилуса. Он также может использовать экранную виртуальную клавиатуру. Конечно, он также может быть присоединен к клавиатуре или другому устройству пользовательского ввода через подходящий механизм присоединения, такой как, например, линия беспроводной связи или USB-порт. Кроме того, компьютер 600 также может принимать речевые вводы.

[00100] Фиг. 10 и 11 предоставляют дополнительные примеры устройств 16, которые могут быть использованы, хотя также могут использоваться другие. На фиг. 10, традиционный мобильный телефон, смартфон или мобильный телефон 45 предоставляется в качестве устройства 16. Телефон 45 включает в себя набор клавишных панелей 47 для набора телефонных номеров, дисплей 49, допускающий отображение изображений, включающих в себя изображения приложений, значки, веб-страницы, фотографии и видео, и кнопки 51 управления для выбора элементов, показанных на дисплее. Телефон включает в себя антенну 53 для приема сотовых телефонных сигналов, таких как сигналы по стандарту общей службы пакетной радиопередачи (GPRS) и 1Xrtt и службы коротких сообщений (SMS). В некоторых примерах, телефон 45 также включает в себя гнездо 55 для вставки карт по стандарту Secure Digital (SD), которое принимает SD-карту 57.

[00101] Мобильное устройство по фиг. 11 представляет собой персональное цифровое устройство 59 (PDA) или мультимедийный проигрыватель, или планшетное вычислительное устройство и т.д. (в дальнейшем называемое PDA 59). PDA 59 включает в себя индуктивный экран 61, который опознает позицию стилуса 63 (или других

указателей, к примеру, пальца пользователя), когда стилус помещается на экране. Это позволяет пользователю выбирать, подсвечивать и перемещать элементы на экране, а также рисовать и писать. PDA 59 также включает в себя ряд клавиш или кнопок пользовательского ввода (к примеру, кнопку 65), которые обеспечивают возможность пользователю прокручивать пункты меню или другие отображаемые пункты, которые отображаются на дисплее 61, и обеспечивают возможность пользователю изменять приложения или выбирать функции пользовательского ввода без контакта с дисплеем 61. Хотя не показано, PDA 59 может включать в себя внутреннюю антенну и инфракрасное приемо-передающее устройство, которые обеспечивают беспроводную связь с другими компьютерами, а также порты подключения, которые обеспечивают аппаратные подключения к другим вычислительным устройствам. Такие аппаратные подключения типично осуществляются через подставку, которая подключается к другому компьютеру через последовательный или USB-порт. В связи с этим, эти подключения являются несетевыми подключениями. В одном примере, мобильное устройство 59 также включает в себя гнездо 67 для вставки SD-карт, которое принимает SD-карту 69.

[00102] Фиг. 12 является аналогичным фиг. 10 за исключением того, что телефон представляет собой смартфон 71.

Смартфон 71 имеет сенсорный дисплей 73, который отображает значки или плитки или другие механизмы 75 пользовательского ввода. Механизмы 75 могут использоваться пользователем для того, чтобы запускать приложения, выполнять вызовы, выполнять операции передачи данных и т.д. В общем, смартфон 71 компонуется на основе мобильной операционной системы и предлагает расширенные вычислительные возможности и возможности подключения по сравнению с традиционным мобильным телефоном.

[00103] Следует отметить, что возможны другие формы устройств 16.

[00104] Фиг. 13 является одним примером вычислительного окружения, в котором (например) может развертываться архитектура 100 или ее части. Со ссылкой на фиг. 13, примерная система для реализации некоторых примеров включает в себя вычислительное устройство общего назначения в форме компьютера 810. Компоненты компьютера 810 могут включать в себя, но не только, модуль 820 обработки (который может содержать процессор 110), системное запоминающее устройство 830 и системную шину 821, которая соединяет различные системные компоненты, включающие в себя системное запоминающее устройство, с модулем 820 обработки. Системная шина 821 может представлять собой любую из нескольких типов шинных структур, включающих в себя шину запоминающего устройства или контроллер запоминающего устройства, периферийную шину и локальную шину с использованием любой из множества шинных архитектур. В качестве примера, а не ограничения, такие архитектуры включают в себя шину со стандартной промышленной архитектурой (ISA), шину с микроканальной архитектурой (MCA), шину с усовершенствованной ISA-архитектурой (EISA), локальную шину с архитектурой Ассоциации по стандартизации в области видеоэлектроники (VESA) и шину на основе стандарта взаимодействия периферийных компонентов (PCI), также известную как дополнительная шина. Запоминающее устройство и программы, описанные относительно фиг. 1, могут быть развернуты в соответствующих частях на фиг. 13.

[00105] Компьютер 810 типично включает в себя множество машиночитаемых носителей. Машиночитаемые носители могут представлять собой любые носители, которые могут быть доступными посредством компьютера 810, и они включают в себя

энергозависимые и энергонезависимые носители, съемные и стационарные носители. В качестве примера, а не ограничения, машиночитаемые носители могут содержать компьютерные носители хранения данных и среды связи. Компьютерные носители хранения данных отличаются от и не включают в себя модулированный сигнал данных или несущую. Они включают в себя аппаратные носители хранения данных, включающие в себя энергозависимые и энергонезависимые, съемные и стационарные носители, реализованные любым способом или технологией хранения информации, такой как машиночитаемые инструкции, структуры данных, программные модули или другие данные. Компьютерные носители хранения данных включают в себя, но не только, RAM, ROM, EEPROM, флэш-память, или другую технологию запоминающих устройств, CD-ROM, цифровые универсальные диски (DVD) или другое устройство хранения данных на оптических дисках, магнитные кассеты, магнитную ленту, устройство хранения данных на магнитных дисках или другие магнитные устройства хранения данных, либо любой другой носитель, который может быть использован для того, чтобы сохранять требуемую информацию, и к которому может быть осуществлен доступ посредством компьютера 810. Среда связи обычно осуществляет машиночитаемые инструкции, структуры данных, программные модули или другие данные в транспортном механизме и включает в себя любые среды доставки информации. Термин "модулированный сигнал данных" означает сигнал, который имеет одну или более характеристик, заданных или измененных таким образом, чтобы кодировать информацию в сигнале. В качестве примера, а не ограничения, среды связи включают в себя проводные среды, такие как проводная сеть или прямое проводное соединение, и беспроводные среды, такие как акустические, радиочастотные (RF), инфракрасные и другие беспроводные среды. Комбинации любых из вышеприведенных элементов также должны быть включены в объем машиночитаемых носителей.

[00106] Системное запоминающее устройство 830 включает в себя компьютерные носители хранения данных в виде энергозависимого и/или энергонезависимого запоминающего устройства, такого как постоянное запоминающее устройство (ROM) 831 и оперативное запоминающее устройство (RAM) 832. Базовая система 833 ввода/вывода (BIOS), содержащая базовые процедуры, которые помогают передавать информацию между элементами в пределах компьютера 810, к примеру, в ходе запуска, типично сохранена в ROM 831. RAM 832 типично содержит в себе данные и/или программные модули, которые непосредственно доступны и/или в данный момент управляются посредством процессора 820. В качестве примера, а не ограничения, фиг. 13 иллюстрирует операционную систему 834, прикладные программы 835, другие программные модули 836 и программные данные 837.

[00107] Компьютер 810 также может включать в себя другие съемные/стационарные, энергозависимые/энергонезависимые компьютерные носители хранения данных. Только в качестве примера, фиг. 13 иллюстрирует жесткий диск 841, который считывает или записывает из/в стационарные энергонезависимые магнитные носители, и накопитель 855 на оптических дисках, который считывает или записывает из/в съемный энергонезависимый оптический диск 856, такой как CD-ROM или другие оптические носители. Другие съемные/стационарные, энергозависимые/энергонезависимые компьютерные носители хранения данных, которые могут быть использованы в примерном операционном окружении, включают в себя, но не только, кассеты с магнитной лентой, карты флэш-памяти, цифровые универсальные диски, цифровую видеоленту, полупроводниковое RAM, полупроводниковое ROM и т.п. Жесткий диск 841 типично подключен к системной шине 821 через интерфейс стационарного

запоминающего устройства, например, интерфейс 840, а накопитель 855 на оптических дисках типично подключен к системной шине 821 посредством интерфейса съемного запоминающего устройства, например, интерфейса 850.

[00108] Альтернативно или помимо этого, функциональность, описанная в данном документе, может выполняться, по меньшей мере, частично, посредством одного или более аппаратных логических компонентов. Например, и без ограничения, типы аппаратных логических компонентов, которые могут быть использованы, включают в себя программируемые пользователем вентильные матрицы (FPGA), специализированные интегральные схемы (ASIC), специализированные микросхемы для массового производства (ASSP), внутрикристальные системы (SOC), комплексные программируемые логические устройства (CPLD) и т.д.

[00109] Накопители и ассоциированные компьютерные носители хранения данных, поясненные выше и проиллюстрированные на фиг. 13, обеспечивают хранение машиночитаемых инструкций, структур данных, программных модулей и других данных для компьютера 810. На фиг. 13, например, жесткий диск 841 проиллюстрирован как сохраняющий операционную систему 844, прикладные программы 845, другие программные модули 846 и программные данные 847. Следует отметить, что эти компоненты могут быть идентичными или отличными от операционной системы 834, прикладных программ 835, других программных модулей 836 и программных данных 837. Операционной системе 844, прикладным программам 845, другим программным модулям 846 и программным данным 847 в настоящем документе присвоены другие номера, чтобы иллюстрировать то, что, как минимум, они являются различными копиями.

[00110] Пользователь может вводить команды и информацию в компьютер 810 через устройства ввода, такие как клавиатура 862, микрофон 863 и указательное устройство 861, к примеру, мышь, шаровой манипулятор или сенсорная панель. Другие устройства ввода (не показаны) могут включать в себя джойстик, игровой планшет, спутниковую антенну, сканер и т.п. Эти и другие устройства ввода зачастую подключены к процессору 820 через интерфейс 860 пользовательского ввода, который соединен с системной шиной, но могут быть подключены посредством других интерфейсов и шинных структур, таких как параллельный порт, игровой порт или универсальная последовательная шина (USB). Видеодисплей 891 или другой тип устройства отображения также соединяется с системной шиной 821 посредством такого интерфейса, как видеоинтерфейс 890. Помимо монитора, компьютеры также могут включать в себя другие периферийные устройства вывода, например, динамики 897 и принтер 896, которые могут быть соединены через периферийный интерфейс 895 вывода.

[00111] Компьютер 810 работает в сетевом окружении с использованием логических подключений к одному или более удаленных компьютеров, таких как удаленный компьютер 880. Удаленный компьютер 880 может представлять собой персональный компьютер, карманное устройство, сервер, маршрутизатор, сетевой PC, равноправное устройство или другой общий сетевой узел и типично включает в себя многие или все из элементов, описанных выше относительно компьютера 810. Логические соединения, показанные на фиг. 13, включают в себя локальную вычислительную сеть 871 (LAN) и глобальную вычислительную сеть 773 (WAN), но также могут включать в себя другие сети. Такие сетевые окружения широко распространены в офисах, корпоративных компьютерных сетях, сетях intranet и в Интернете.

[00112] При использовании в сетевом LAN-окружении, компьютер 810 подключается к LAN 871 через сетевой интерфейс или адаптер 870. При использовании в сетевом

WAN-окружении, компьютер 810 типично включает в себя модем 872 или другое средство для установления связи по WAN 873, к примеру, по Интернету. Модем 872, который может быть внутренним или внешним, может быть подключен к системной шине 821 через интерфейс 860 пользовательского ввода или другой подходящий механизм. В сетевом окружении, программные модули, проиллюстрированные относительно компьютера 810, или их части могут быть сохранены в удаленном запоминающем устройстве. В качестве примера, но не ограничения, фиг. 13 иллюстрирует удаленные прикладные программы 885 как находящиеся на удаленном компьютере 880. Следует принимать во внимание, что показанные сетевые соединения являются иллюстративными, и могут использоваться другие средства установления линии связи между компьютерами.

[00113] Также следует отметить, что различные варианты осуществления, описанные в данном документе, могут комбинироваться по-разному. Иными словами, части одного или более вариантов осуществления могут комбинироваться с частями одного или более других вариантов осуществления. Все это предполагается в данном документе.

[00114] Пример 1 представляет собой систему разработки, содержащую модуль разработки, считывающий пользовательские вводы по разработке и преобразующий элементы компьютерной системы на основе пользовательских вводов по разработке. Элементы содержат типы, моделируемые в компьютерной системе. Модуль пользовательского интерфейса формирует экран пользовательского интерфейса с помощью механизма пользовательского ввода и считывает пользовательский поисковый ввод, принимаемый через механизм пользовательского ввода, указывающий пользовательский поисковый запрос для поиска элементов компьютерной системы. Поисковый механизм идентифицирует параметр поиска на основе типа для пользовательского поискового запроса. Поисковый механизм управляется с возможностью активировать компонент поиска на основе типа на основе параметра поиска на основе типа. Компонент поиска на основе типа выполняет элементный поиск для того, чтобы возвращать набор результатов поиска на экране пользовательского интерфейса.

[00115] Пример 2 представляет собой систему разработки по любым из предыдущих примеров, в которой модуль разработки является частью интерактивной среды разработки (IDE).

[00116] Пример 3 представляет собой систему разработки по любым из предыдущих примеров, в которой пользователь является разработчиком, и элементы компьютерной системы содержат элементы приложения, которые индивидуально настраиваются разработчиком.

[00117] Пример 4 представляет собой систему разработки по любым из предыдущих примеров, в которой параметр поиска на основе типа идентифицирует конкретный тип элемента, выбранный из типов, моделируемых в компьютерной системе, и поисковый механизм управляется с возможностью ограничивать элементный поиск элементами, имеющими конкретный тип элемента.

[00118] Пример 5 представляет собой систему разработки по любым из предыдущих примеров, в которой пользовательский поисковый запрос включает в себя строку символов и конкретный тип элемента.

[00119] Пример 6 представляет собой систему разработки по любым из предыдущих примеров, в которой набор результатов поиска содержит элементы конкретного типа элемента, которые имеют значения свойств, которые совпадают со строкой символов.

[00120] Пример 7 представляет собой систему разработки по любым из предыдущих

примеров, в которой элементы компьютерной системы содержат множество различных типов, причем каждый тип имеет набор свойств и методов, которые задают поведение во время выполнения для элементов этого типа элемента. Система дополнительно содержит хранилище компонентов поиска, сохраняющее множество компонентов поиска, причем каждый компонент поиска соответствует данному одному из различных типов и выполнен с возможностью осуществлять поиск в наборе свойств и методов на предмет элементов данного типа.

[00121] Пример 8 представляет собой систему разработки по любым из предыдущих примеров, в которой поисковый механизм идентифицирует компонент поиска на основе типа из хранилища компонентов поиска, который соответствует конкретному типу элемента, идентифицирует каждый из множества элементов компьютерной системы, который имеет конкретный тип элемента, и выполняет поиск идентифицированных элементов на основе пользовательского поискового запроса с использованием идентифицированного компонента поиска.

[00122] Пример 9 представляет собой систему разработки по любым из предыдущих примеров, в которой экземпляр идентифицированного компонента поиска создается для каждого из множества идентифицированных элементов, имеющих конкретный тип элемента, и поисковый механизм получает набор результатов поиска посредством агрегирования результатов поиска из множества компонентов поиска, экземпляры которых созданы, и отображает агрегированные результаты поиска на экране пользовательского интерфейса.

[00123] Пример 10 представляет собой систему разработки по любым из предыдущих примеров, в которой набор результатов поиска получается и отображается асинхронно.

[00124] Пример 11 представляет собой систему разработки по любым из предыдущих примеров, дополнительно содержащую хранилище моделей, сохраняющее, для каждого из элементов, преобразованное в последовательную форму представление элемента, содержащее код и метаданные элемента. Поисковый механизм выполняет элементный поиск посредством осуществления доступа к преобразованным в последовательную форму представлениям в хранилище моделей.

[00125] Пример 12 представляет собой систему разработки по любым из предыдущих примеров, в которой поисковый механизм идентифицирует конкретное преобразованное в последовательную форму представление в хранилище моделей, соответствующее данному одному из элементов, на основе параметра поиска на основе типа и выполняет поиск конкретного преобразованного в последовательную форму представления на основе пользовательского поискового запроса.

[00126] Пример 13 представляет собой систему разработки по любым из предыдущих примеров, в которой поисковый механизм идентифицирует часть данного элемента из конкретного преобразованного в последовательную форму представления, которая совпадает с пользовательским поисковым запросом, и идентифицирует информацию пути, которая уникально идентифицирует часть данного элемента.

[00127] Пример 14 представляет собой систему разработки по любым из предыдущих примеров, в которой информация пути содержит универсальный идентификатор ресурса (URI), модуль пользовательского интерфейса формирует выбираемое пользователем представление URI, которое может выбираться, чтобы представлять часть данного элемента в пользовательском интерфейсе редактора.

[00128] Пример 15 представляет собой систему разработки, содержащую хранилище данных, которое моделирует множество различных типов элементов, модуль разработки, считывающий вводы разработчика и преобразующий элементы приложения различных

типов элементов на основе вводов разработчика, и генератор компонентов поиска, формирующий различный компонент поиска для каждого из типов элементов, моделируемых в хранилище данных. Система разработки также содержит хранилище компонентов поиска, сохраняющее компоненты поиска, сформированные посредством генератора компонентов поиска для множества типов элементов, и поисковый механизм, считывающий пользовательский поисковый ввод и управляемый с возможностью активировать выбранный один из компонентов поиска, чтобы выполнять поиск элементов приложения данного одного из типов элементов.

[00129] Пример 16 представляет собой систему разработки по любым из предыдущих примеров, в которой генератор компонентов поиска выполнен с возможностью формировать каждый компонент поиска посредством анализа структуры данного одного из типов элементов и формирования соответствующих функций поиска на основе структуры данного типа элемента.

[00130] Пример 17 представляет собой систему разработки по любым из предыдущих примеров, в которой структура данного типа элемента задается посредством набора свойств и методов, которые задают поведение во время выполнения элементов, имеющих данный тип элемента.

[00131] Пример 18 представляет собой систему разработки по любым из предыдущих примеров, в которой поисковый механизм принимает поисковый запрос, имеющий, по меньшей мере, один критерий поиска, идентифицирует параметр поиска на основе типа для поискового запроса и идентифицирует один из компонентов поиска из хранилища компонентов поиска на основе параметра поиска на основе типа. Экземпляр идентифицированного компонента поиска создается для того, чтобы выполнять поиск одного или более элементов приложения на основе критерия поиска.

[00132] Пример 19 представляет собой машинореализованный способ для разработки элементов компьютерной системы и управления поиском элементов. Способ содержит считывание пользовательских вводов по разработке и преобразование элементов компьютерной системы на основе пользовательских вводов по разработке. Компьютерная система содержит множество различных типов элементов, причем каждый тип элемента задается посредством структуры свойств для элементов типа элемента. Способ содержит формирование экрана поискового интерфейса и считывание пользовательского ввода, через экран поискового интерфейса, указывающий пользовательский поисковый запрос, чтобы выполнять поиск элементов компьютерной системы. Способ содержит управление поиском элементов компьютерной системы на основе пользовательского поискового запроса и ограничения семантического поиска, которое основано на структурах свойств элементов компьютерной системы. Способ содержит возврат результатов поиска из поиска и формирование экрана результатов, который отображает результаты поиска.

[00133] Пример 20 представляет собой машинореализованный способ по любым из предыдущих примеров, дополнительно содержащий осуществление доступа к хранилищу данных, которое моделирует множество различных типов элементов, и для каждого различного типа элемента, формирование соответствующего компонента поиска на основе типа на основе структуры свойств типа элемента. Способ содержит использование, по меньшей мере, одного из сформированных компонентов поиска на основе типа, который выбирается на основе ограничения семантического поиска, чтобы выполнять поиск в хранилище данных на основе пользовательского поискового запроса.

[00134] Хотя изобретение описано на языке, характерном для структурных признаков и/или технологических этапов, следует понимать, что объем изобретения, определяемый

прилагаемой формулой изобретения, не обязательно ограничен характерными признаками или этапами, описанными выше. Наоборот, характерные признаки и этапы, описанные выше, раскрыты в качестве примерных форм реализации формулы изобретения, и другие эквивалентные признаки и этапы подразумеваются находящимися в пределах объема формулы изобретения.

(57) Формула изобретения

1. Система разработки для разработки элементов компьютерной системы и управления поиском элементов, причем система разработки содержит:

модуль разработки, считывающий пользовательские вводы по разработке и преобразующий элементы компьютерной системы на основе пользовательских вводов по разработке, причем элементы содержат типы, моделируемые в компьютерной системе;

модуль пользовательского интерфейса, формирующий отображение пользовательского интерфейса с помощью механизма пользовательского ввода и считывающий пользовательский поисковый ввод, принимаемый через механизм пользовательского ввода, указывающий пользовательский поисковый запрос для поиска элементов компьютерной системы; и

поисковый механизм, идентифицирующий параметр поиска на основе типа для пользовательского поискового запроса, причем поисковый механизм управляется для активации компонента поиска на основе типа, основываясь на параметре поиска на основе типа, причем компонент поиска на основе типа выполняет элементный поиск, чтобы выдавать набор результатов поиска в отображении пользовательского интерфейса,

при этом элементы компьютерной системы содержат множество различных типов элементов, причем каждый тип имеет набор свойств и методов, которые задают поведение во время исполнения для элементов этого типа элемента,

при этом система дополнительно содержит хранилище компонентов поиска, хранящее множество компонентов поиска, причем каждый компонент поиска соответствует заданному одному из упомянутых различных типов элементов и выполнен с возможностью осуществлять поиск в наборе свойств и методов на предмет элементов этого заданного типа.

2. Система разработки по п. 1, в которой модуль разработки является частью интерактивной среды разработки (IDE).

3. Система разработки по п. 1, в которой пользователь является разработчиком и элементы компьютерной системы содержат элементы приложения, которые индивидуально настраиваются разработчиком.

4. Система разработки по п. 1, в которой параметр поиска на основе типа идентифицирует конкретный тип элемента, выбранный из типов, моделируемых в компьютерной системе, и поисковый механизм управляется для ограничения элементного поиска элементами, имеющими этот конкретный тип элемента.

5. Система разработки по п. 4, в которой пользовательский поисковый запрос включает в себя строку символов и упомянутый конкретный тип элемента, при этом набор результатов поиска содержит элементы этого конкретного типа элемента, которые имеют значения свойств, которые совпадают со строкой символов.

6. Система разработки по п. 1 или 4, в которой поисковый механизм идентифицирует компонент поиска на основе типа из хранилища компонентов поиска, который соответствует упомянутому конкретному типу элемента, идентифицирует каждый из

множества элементов компьютерной системы, который имеет этот конкретный тип элемента, и выполняет поиск идентифицированных элементов на основе пользовательского поискового запроса с использованием идентифицированного компонента поиска.

7. Система разработки по п. 6, в которой экземпляр идентифицированного компонента поиска на основе типа создается для каждого из множества идентифицированных элементов, имеющих упомянутый конкретный тип элемента, и поисковый механизм получает набор результатов поиска посредством агрегирования результатов поиска из множества компонентов поиска, экземпляры которых созданы, и отображает агрегированные результаты поиска в отображении пользовательского интерфейса.

8. Система разработки по п. 7, в которой набор результатов поиска получается и отображается асинхронно.

9. Система разработки по п. 1, дополнительно содержащая хранилище моделей, хранящее, для каждого из элементов, преобразованное в последовательную форму представление элемента, содержащее код и метаданные элемента; при этом поисковый механизм выполняет элементный поиск посредством осуществления доступа к преобразованным в последовательную форму представлениям в хранилище моделей.

10. Система разработки по п. 9, в которой поисковый механизм идентифицирует конкретное преобразованное в последовательную форму представление в хранилище моделей, соответствующее заданному одному из элементов, на основе параметра поиска на основе типа и выполняет поиск этого конкретного преобразованного в последовательную форму представления на основе пользовательского поискового запроса.

11. Система разработки по п. 10, в которой поисковый механизм идентифицирует часть упомянутого заданного элемента из упомянутого конкретного преобразованного в последовательную форму представления, которая совпадает с пользовательским поисковым запросом, и идентифицирует информацию пути, которая уникально идентифицирует эту часть заданного элемента.

12. Система разработки по п. 11, в которой информация пути содержит универсальный идентификатор ресурса (URI), причем модуль пользовательского интерфейса формирует выбираемое пользователем представление URI, которое может быть выбрано, чтобы представлять упомянутую часть заданного элемента в пользовательском интерфейсе редактора.

13. Система разработки, содержащая:

хранилище данных, которое моделирует множество различных типов элементов;

модуль разработки, считывающий пользовательские вводы по разработке и преобразующий элементы приложения этих различных типов элементов на основе пользовательских вводов по разработке;

генератор компонентов поиска, формирующий отличный от других компонент поиска для каждого из типов элементов, моделируемых в хранилище данных;

хранилище компонентов поиска, хранящее компоненты поиска, сформированные посредством генератора компонентов поиска для множества типов элементов; и

поисковый механизм, считывающий пользовательский поисковый ввод и управляемый для активации выбранного одного из компонентов поиска, чтобы выполнять поиск элементов приложения заданного одного из типов элементов.

14. Компьютерно-реализуемый способ разработки элементов компьютерной системы и управления поиском элементов, содержащий этапы, на которых:

считывают пользовательские вводы по разработке и преобразуют элементы

компьютерной системы на основе пользовательских вводов по разработке, причем компьютерная система содержит множество различных типов элементов, причем каждый тип элемента имеет набор свойств и методов, которые задают поведение во время исполнения для элементов этого типа элемента;

формируют отображение поискового интерфейса;

считывают пользовательский ввод, через отображение поискового интерфейса, указывающий пользовательский поисковый запрос, чтобы выполнять поиск элементов компьютерной системы;

управляют поиском элементов компьютерной системы на основе пользовательского поискового запроса и ограничения семантического поиска, которое основано на структурах свойств элементов компьютерной системы;

возвращают результаты поиска из этого поиска;

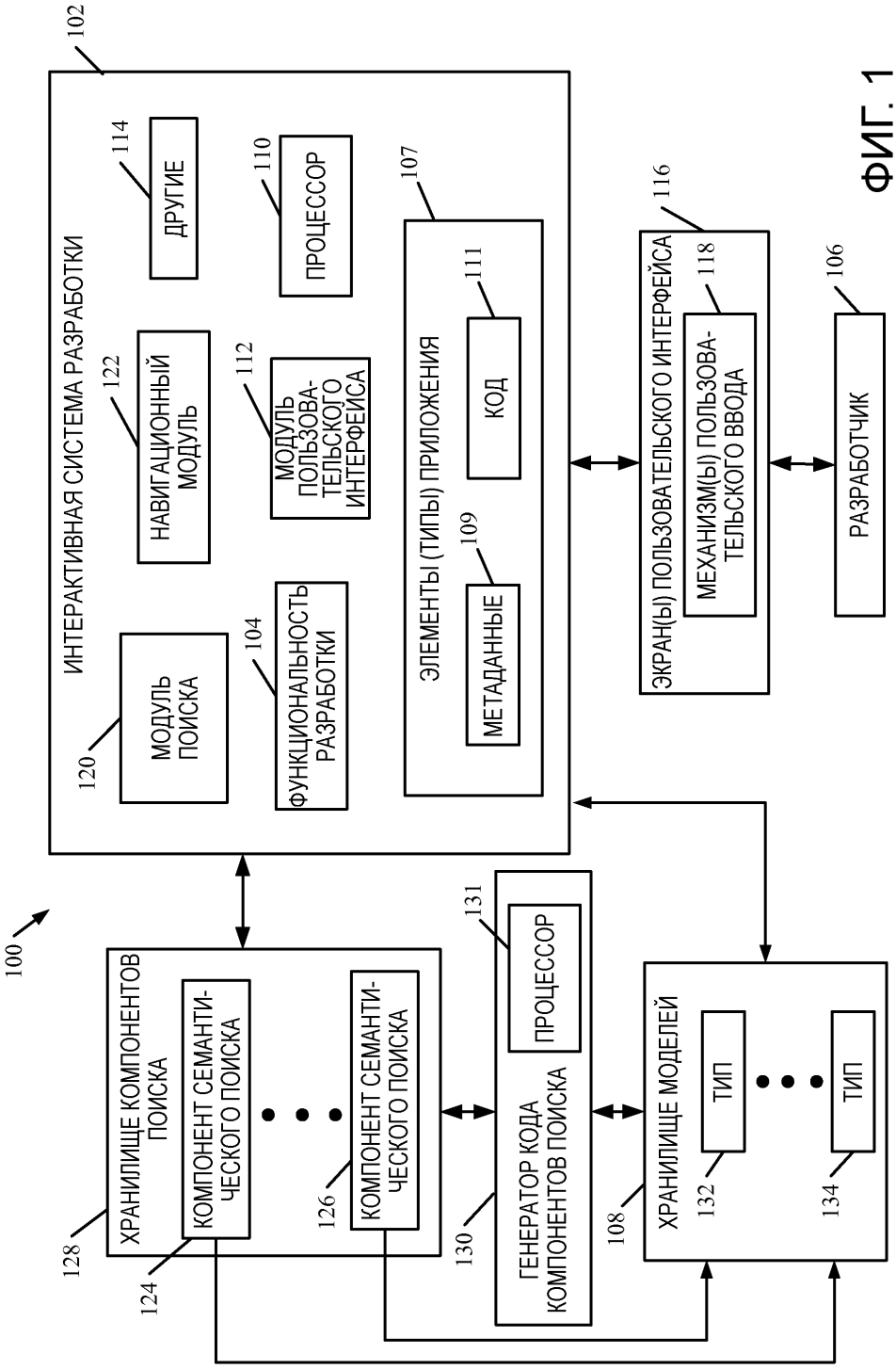
формируют отображение результатов, которым отображаются результаты поиска; и

сохраняют множество компонентов поиска, причем каждый компонент поиска соответствует заданному одному из упомянутых различных типов элементов и выполнен с возможностью осуществлять поиск в наборе свойств и методов на предмет элементов этого заданного типа.

1

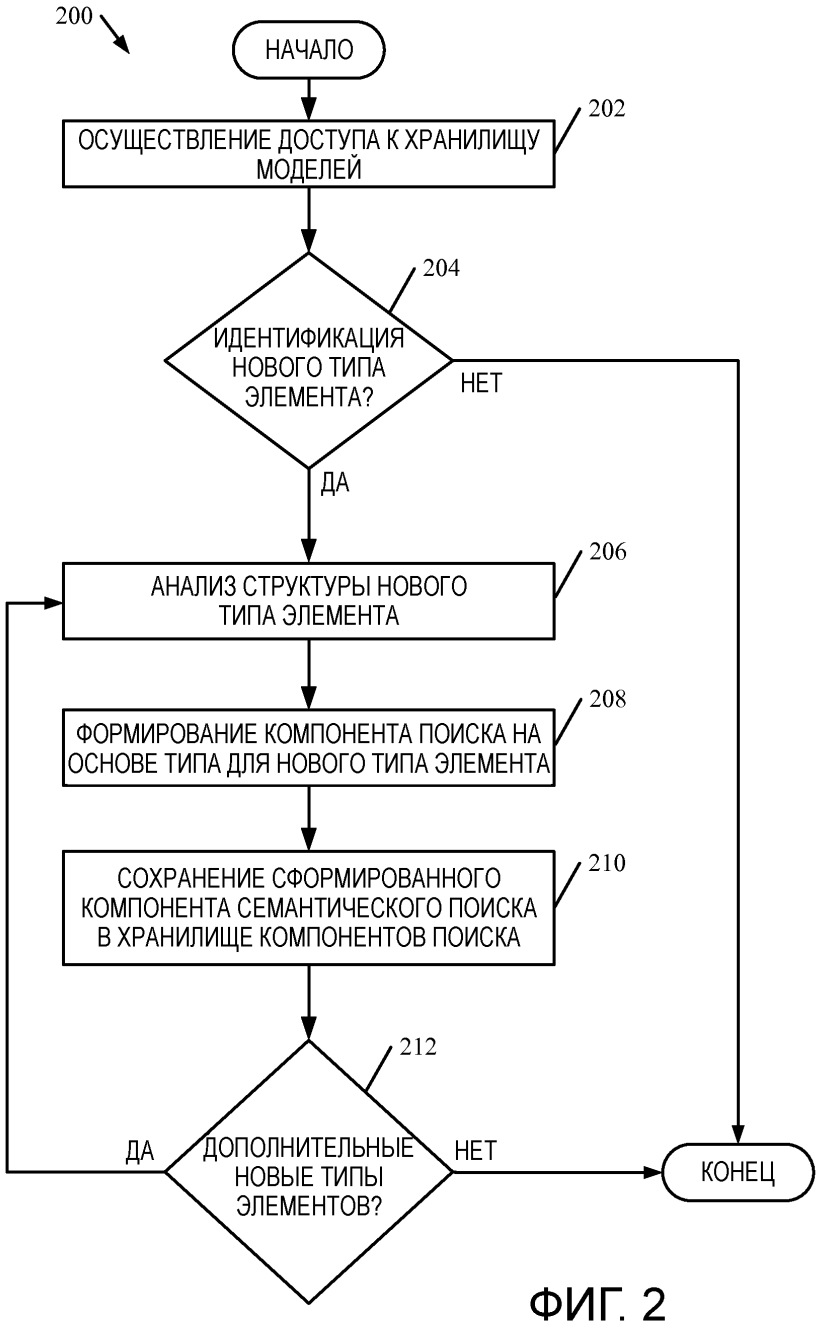
538411

1/13



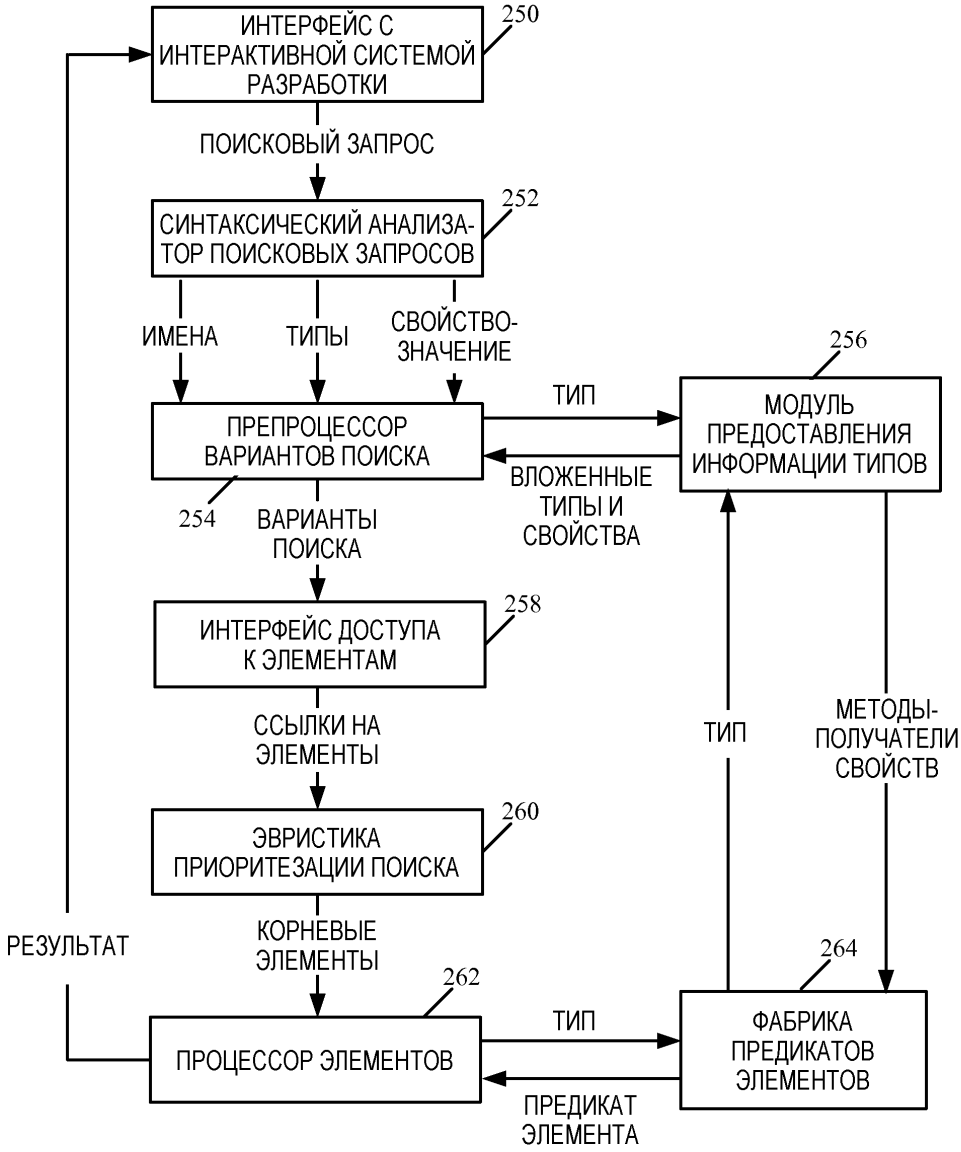
ФИГ. 1

2


200
НАЧАЛО
ОСУЩЕСТВЛЕНИЕ ДОСТУПА К ХРАНИЛИЩУ МОДЕЛЕЙ
202
ИДЕНТИФИКАЦИЯ НОВОГО ТИПА ЭЛЕМЕНТА?
204
НЕТ
ДА
АНАЛИЗ СТРУКТУРЫ НОВОГО ТИПА ЭЛЕМЕНТА
206
ФОРМИРОВАНИЕ КОМПОНЕНТА ПОИСКА НА ОСНОВЕ ТИПА ДЛЯ НОВОГО ТИПА ЭЛЕМЕНТА
208
СОХРАНЕНИЕ СФОРМИРОВАННОГО КОМПОНЕНТА СЕМАНТИЧЕСКОГО ПОИСКА В ХРАНИЛИЩЕ КОМПОНЕНТОВ ПОИСКА
210
ДОПОЛНИТЕЛЬНЫЕ НОВЫЕ ТИПЫ ЭЛЕМЕНТОВ?
212
ДА
НЕТ
КОНЕЦ

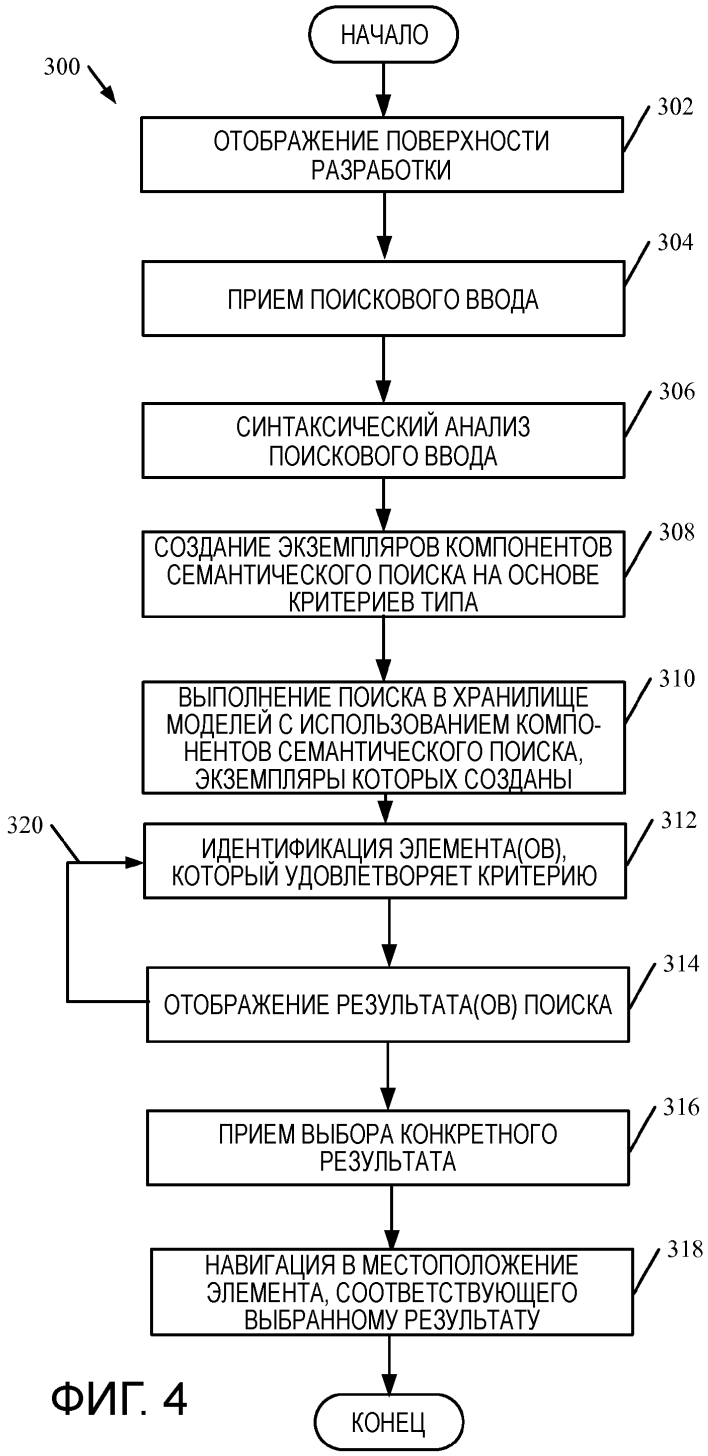
ФИГ. 2

3


ИНТЕРФЕЙС С ИНТЕРАКТИВНОЙ СИСТЕМОЙ РАЗРАБОТКИ
250
ПОИСКОВЫЙ ЗАПРОС
СИНТАКСИЧЕСКИЙ АНАЛИЗАТОР ПОИСКОВЫХ ЗАПРОСОВ
252
ИМЕНА
ТИПЫ
СВОЙСТВО-ЗНАЧЕНИЕ
ПРЕПРОЦЕССОР ВАРИАНТОВ ПОИСКА
ТИП
ВЛОЖЕННЫЕ ТИПЫ И СВОЙСТВА
МОДУЛЬ ПРЕДОСТАВЛЕНИЯ ИНФОРМАЦИИ ТИПОВ
256
254
ВАРИАНТЫ ПОИСКА
ИНТЕРФЕЙС ДОСТУПА К ЭЛЕМЕНТАМ
258
ССЫЛКИ НА ЭЛЕМЕНТЫ
ТИП
МЕТОДЫ-ПОЛУЧАТЕЛИ СВОЙСТВ
ЭВРИСТИКА ПРИОРИТЕЗАЦИИ ПОИСКА
260
РЕЗУЛЬТАТ
КОРНЕВЫЕ ЭЛЕМЕНТЫ
264
262
ТИП
ФАБРИКА ПРЕДИКАТОВ ЭЛЕМЕНТОВ
ПРОЦЕССОР ЭЛЕМЕНТОВ
ПРЕДИКАТ ЭЛЕМЕНТА

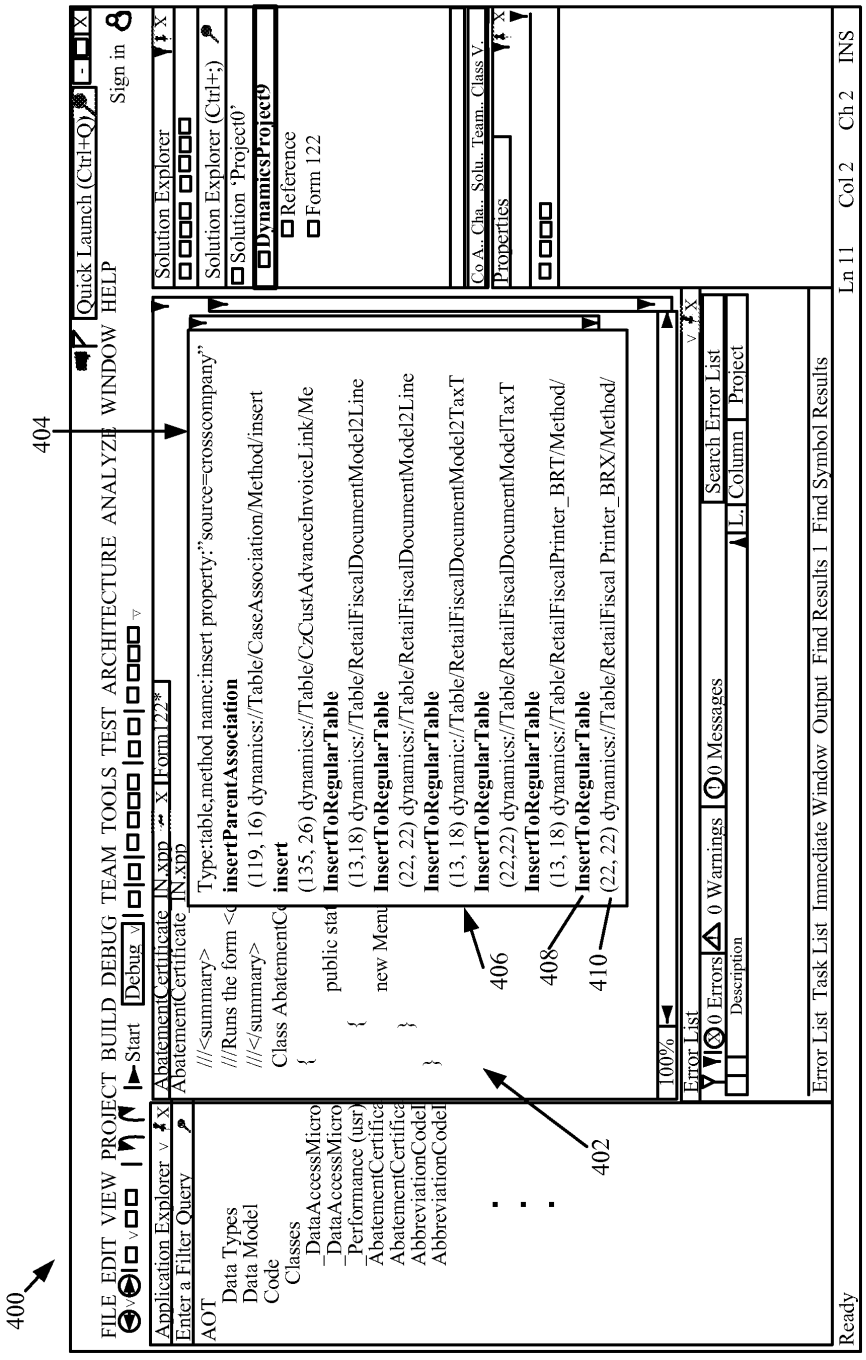
ФИГ. 3

4


НАЧАЛО
300
ОТОБРАЖЕНИЕ ПОВЕРХНОСТИ РАЗРАБОТКИ
302
ПРИЕМ ПОИСКОВОГО ВВОДА
304
СИНТАКСИЧЕСКИЙ АНАЛИЗ ПОИСКОВОГО ВВОДА
306
СОЗДАНИЕ ЭКЗЕМПЛЯРОВ КОМПОНЕНТОВ СЕМАНТИЧЕСКОГО ПОИСКА НА ОСНОВЕ КРИТЕРИЕВ ТИПА
308
ВЫПОЛНЕНИЕ ПОИСКА В ХРАНИЛИЩЕ МОДЕЛЕЙ С ИСПОЛЬЗОВАНИЕМ КОМПОНЕНТОВ СЕМАНТИЧЕСКОГО ПОИСКА, ЭКЗЕМПЛЯРЫ КОТОРЫХ СОЗДАНЫ
310
320
ИДЕНТИФИКАЦИЯ ЭЛЕМЕНТА(ОВ), КОТОРЫЙ УДОВЛЕТВОРЯЕТ КРИТЕРИЮ
312
ОТОБРАЖЕНИЕ РЕЗУЛЬТАТА(ОВ) ПОИСКА
314
ПРИЕМ ВЫБОРА КОНКРЕТНОГО РЕЗУЛЬТАТА
316
НАВИГАЦИЯ В МЕСТОПОЛОЖЕНИЕ ЭЛЕМЕНТА, СООТВЕТСТВУЮЩЕГО ВЫБРАННОМУ РЕЗУЛЬТАТУ
318
КОНЕЦ

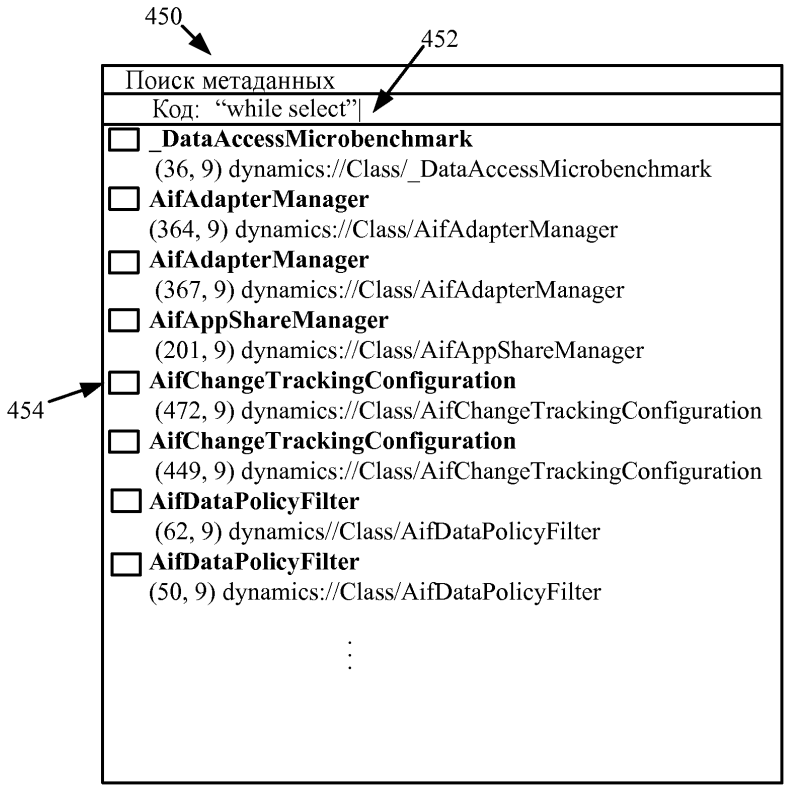
ФИГ. 4

5


400
404
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Start
Debug
Application Explorer
Enter a Filter Query
AOT
Data Types
Data Model
Code
Classes
DataAccessMicro
DataAccessMicro
Performance (usr)
AbatementCertifica
AbatementCertifica
AbbreviationCodel
AbbreviationCodel
Abatement(Certificate_IN.xpp
Abatement(Certificate_IN.xpp
Form 122*
Abatement(Certificate_IN.xpp
///<summary>
///Runs the form <
///</summary>
Class AbatementC
{
public sta
{
new Menu
}
}
}
Type:table,method name:insert property:"source=crosscompany"
insertParentAssociation
(119, 16) dynamics://Table/CaseAssociation/Method/insert
insert
(135, 26) dynamics://Table/CzCustAdvanceInvoiceLink/Me
InsertToRegularTable
(13,18) dynamics://Table/RetailFiscalDocumentModel2Line
InsertToRegularTable
(22, 22) dynamics://Table/RetailFiscalDocumentModel2Line
InsertToRegularTable
(13, 18) dynamic://Table/RetailFiscalDocumentModel2TaxT
InsertToRegularTable
(22,22) dynamics://Table/RetailFiscalDocumentModelTaxT
InsertToRegularTable
(13, 18) dynamics://Table/RetailFiscalPrinter_BRT/Method/
InsertToRegularTable
(22, 22) dynamics://Table/RetailFiscal Printer_BRX/Method/
100%
406
408
410
402
Solution Explorer
Solution Explorer (Ctrl+;)
Solution 'Project0'
DynamicsProject9
Reference
Form 122
Co A.. Cha.. Solu.. Team.. Class V..
Properties
Error List
0 Errors
0 Warnings
0 Messages
Search Error List
Description
Line
Column
Project
Error List Task List Immediate Window Output Find Results 1 Find Symbol Results
Ready
Ln 11
Col 2
Ch 2
INS

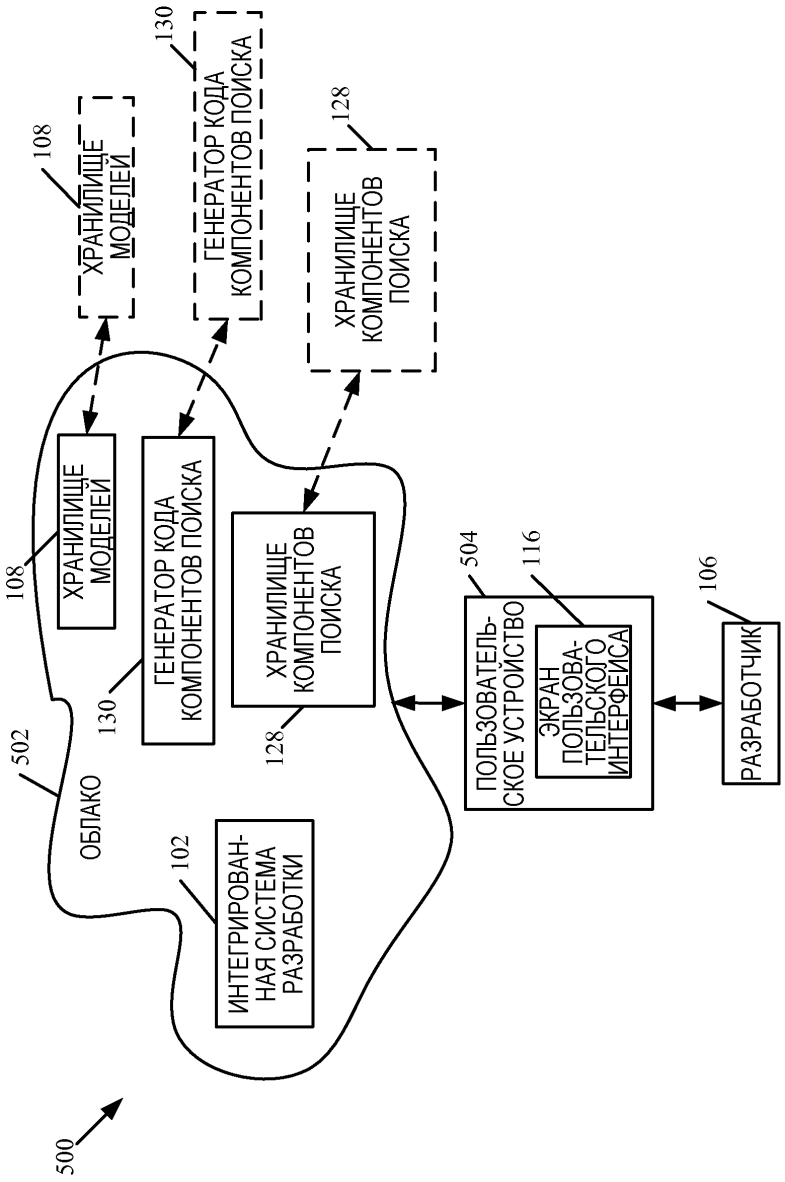
ФИГ. 5

6


450
452
Поиск метаданных
Код: “while select”
_DataAccessMicrobenchmark
(36, 9) dynamics://Class/_DataAccessMicrobenchmark
AifAdapterManager
(364, 9) dynamics://Class/AifAdapterManager
AifAdapterManager
(367, 9) dynamics://Class/AifAdapterManager
AifAppShareManager
(201, 9) dynamics://Class/AifAppShareManager
454
AifChangeTrackingConfiguration
(472, 9) dynamics://Class/AifChangeTrackingConfiguration
AifChangeTrackingConfiguration
(449, 9) dynamics://Class/AifChangeTrackingConfiguration
AifDataPolicyFilter
(62, 9) dynamics//Class/AifDataPolicyFilter
AifDataPolicyFilter
(50, 9) dynamics://Class/AifDataPolicyFilter

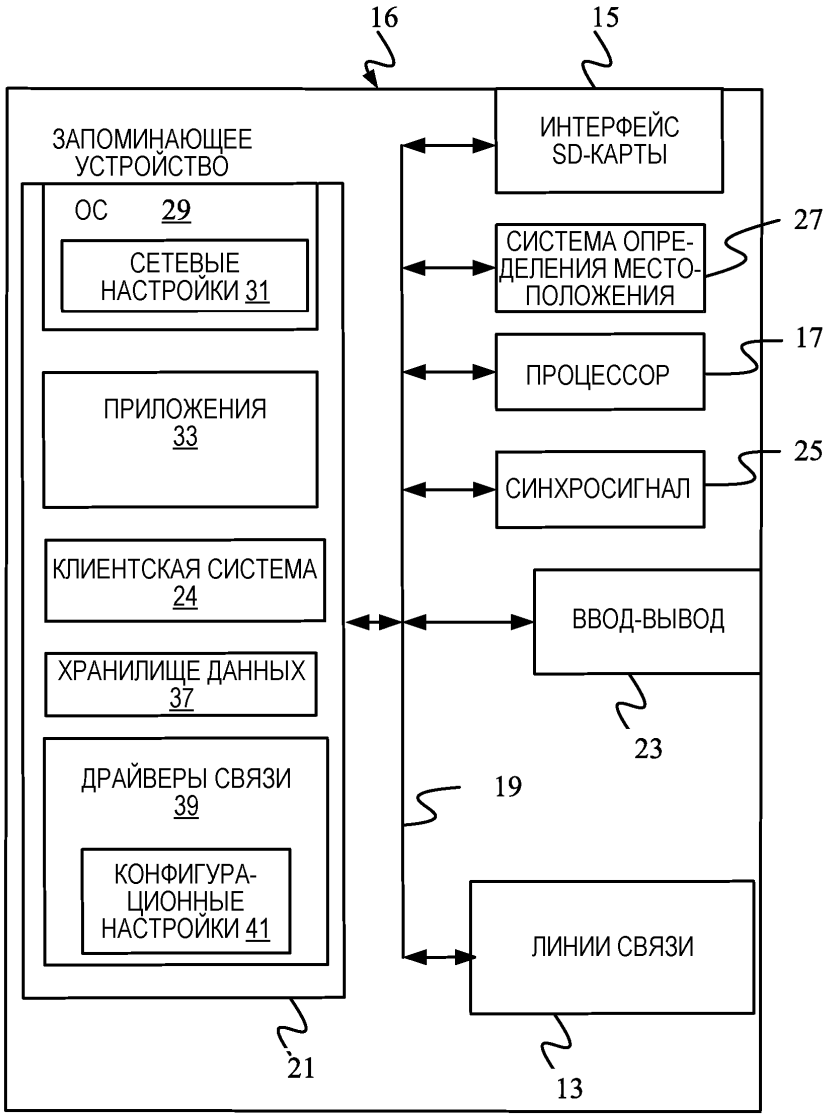
ФИГ. 6

7


500
502
ОБЛАКО
ИНТЕГРИРОВАННАЯ СИСТЕМА РАЗРАБОТКИ
102
ХРАНИЛИЩЕ МОДЕЛЕЙ
108
ГЕНЕРАТОР КОДА КОМПОНЕНТОВ ПОИСКА
130
ХРАНИЛИЩЕ КОМПОНЕНТОВ ПОИСКА
128
ПОЛЬЗОВАТЕЛЬСКОЕ УСТРОЙСТВО
504
ЭКРАН ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА
116
РАЗРАБОТЧИК
106

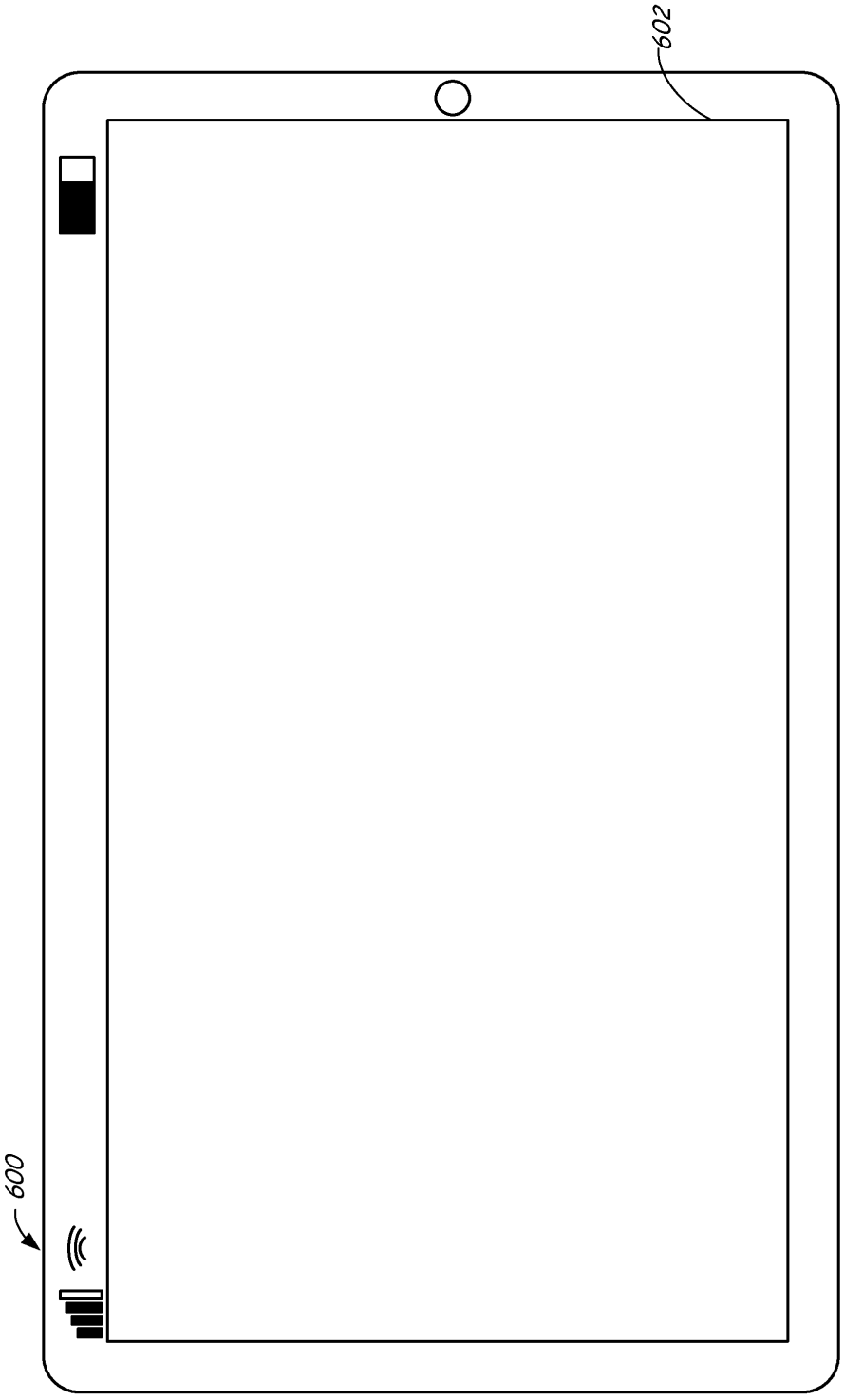
ФИГ. 7

8


16
15
ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО
ОС 29
СЕТЕВЫЕ НАСТРОЙКИ 31
ПРИЛОЖЕНИЯ 33
КЛИЕНТСКАЯ СИСТЕМА 24
ХРАНИЛИЩЕ ДАННЫХ 37
ДРАЙВЕРЫ СВЯЗИ 39
КОНФИГУРА-ЦИОННЫЕ НАСТРОЙКИ 41
21
ИНТЕРФЕЙС SD-КАРТЫ
СИСТЕМА ОПРЕ-ДЕЛЕНИЯ МЕСТО-ПОЛОЖЕНИЯ
27
ПРОЦЕССОР
17
СИНХРОСИГНАЛ
25
ВВОД-ВЫВОД
23
19
ЛИНИИ СВЯЗИ
13

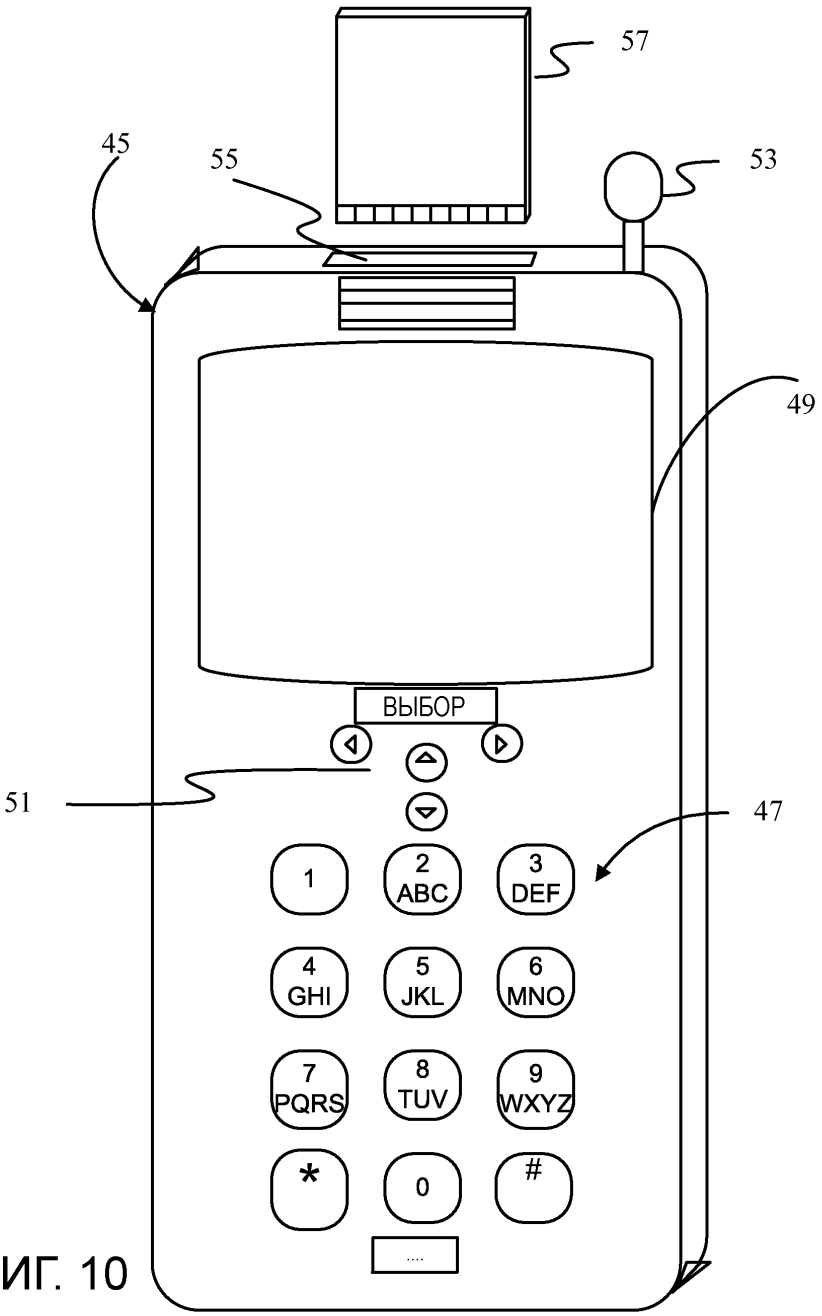
ФИГ. 8

9



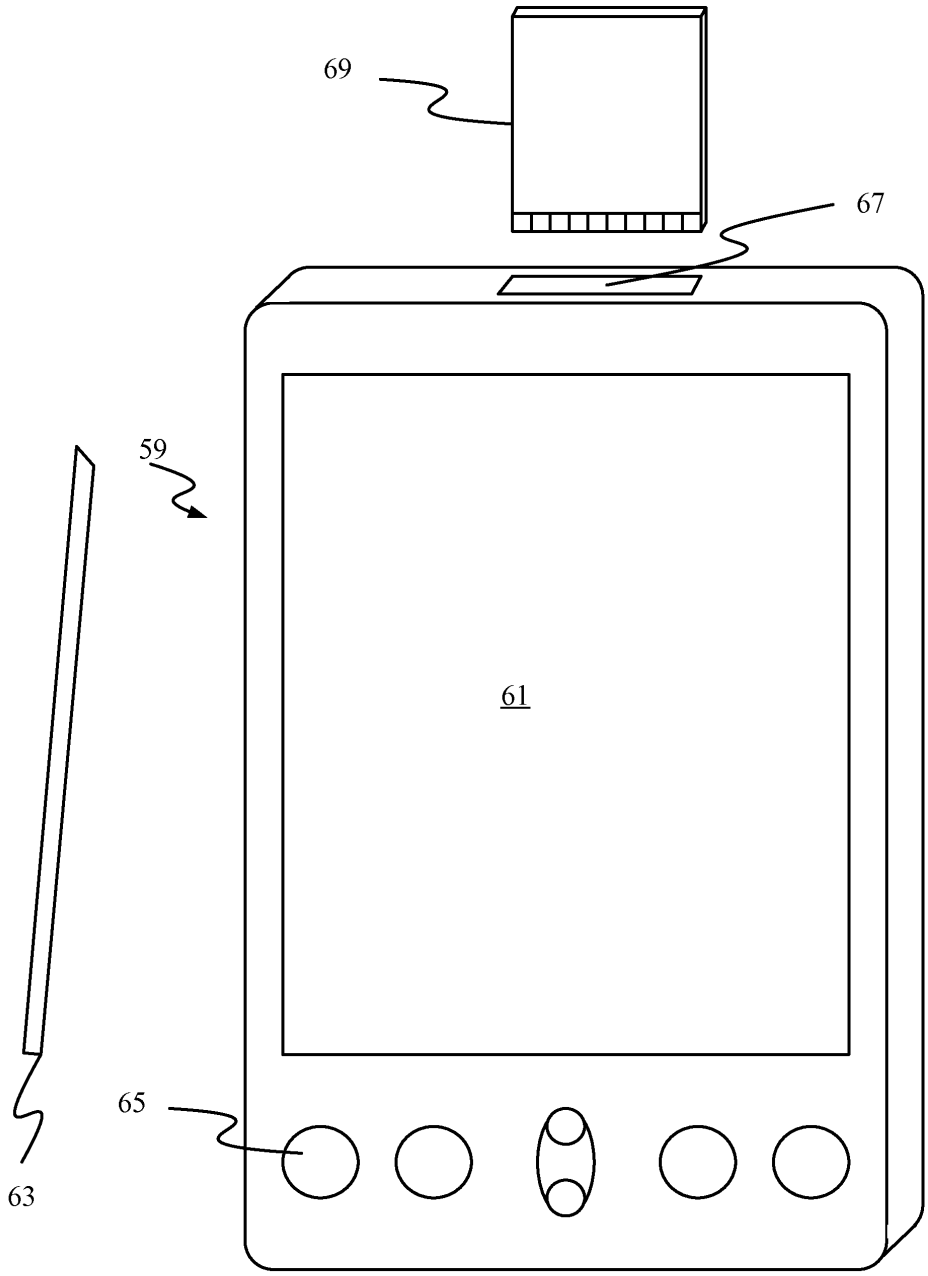
ФИГ. 9

10

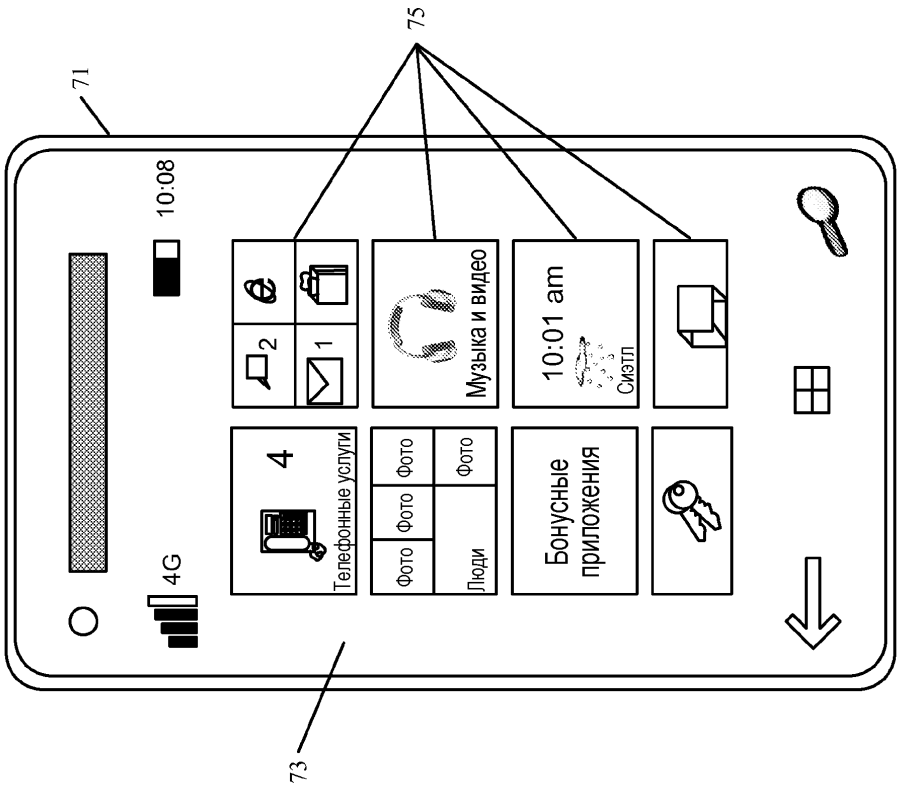

57
45
55
53
49
ВЫБОР
51
47
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0
#

ФИГ. 10

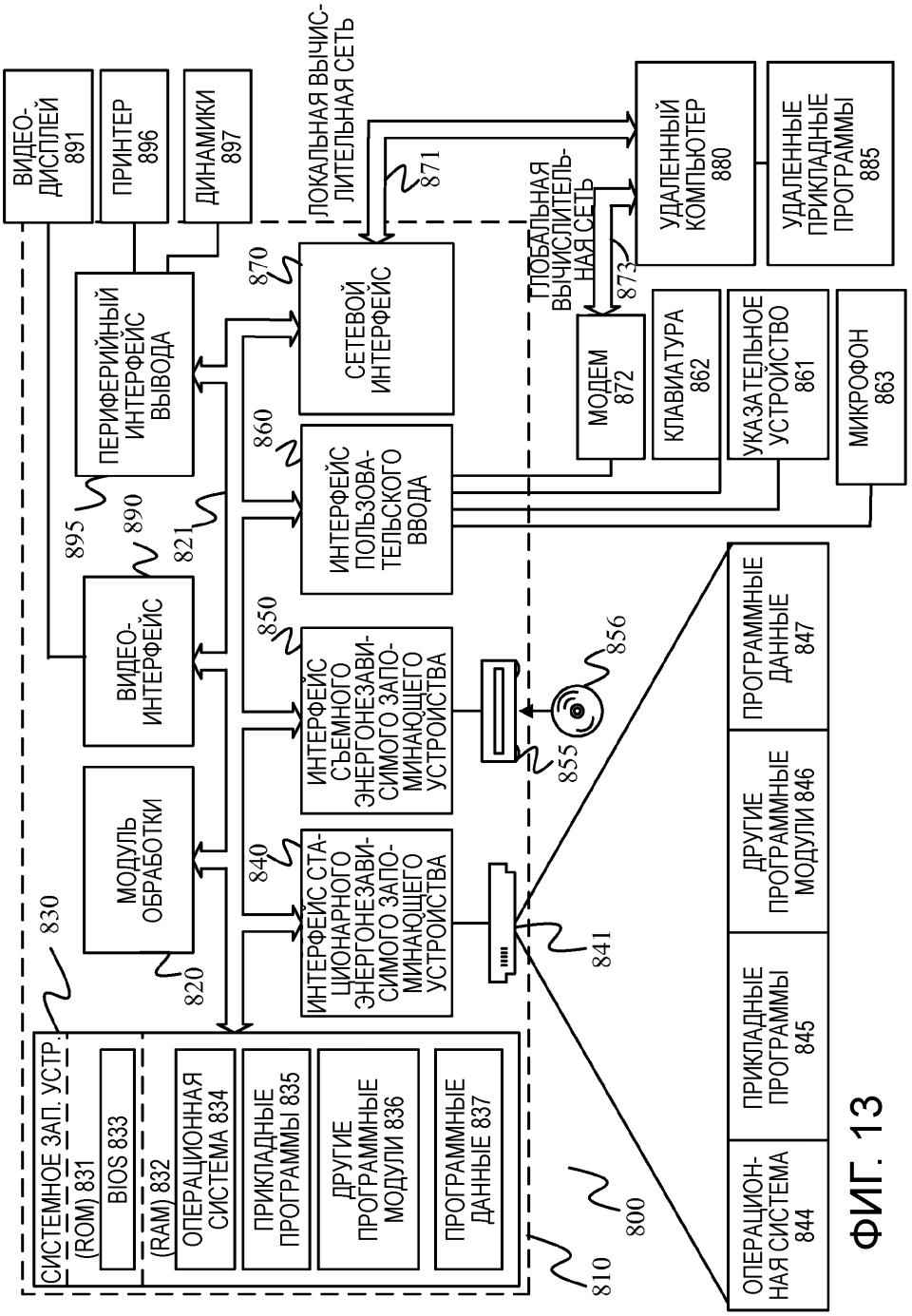
11



ФИГ. 11


71
75
73
10:08
4G
4
Телефонные услуги
Фото
Фото
Фото
Фото
Люди
Музыка и видео
10:01 am
Сиэтл
Бонусные приложения

ФИГ. 12


ВИДЕО-ДИСПЛЕЙ 891
ПРИНТЕР 896
ДИНАМИКИ 897
ЛОКАЛЬНАЯ ВЫЧИСЛИТЕЛЬНАЯ СЕТЬ
871
ГЛОБАЛЬНАЯ ВЫЧИСЛИТЕЛЬНАЯ СЕТЬ
873
УДАЛЕННЫЙ КОМПЬЮТЕР 880
УДАЛЕННЫЕ ПРИКЛАДНЫЕ ПРОГРАММЫ 885
ПЕРИФЕРИЙНЫЙ ИНТЕРФЕЙС ВЫВОДА
870
СЕТЕВОЙ ИНТЕРФЕЙС
МОДЕМ 872
КЛАВИАТУРА 862
УКАЗАТЕЛЬНОЕ УСТРОЙСТВО 861
МИКРОФОН 863
860
ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА
895
890
821
ВИДЕО-ИНТЕРФЕЙС
850
ИНТЕРФЕЙС СЪЕМНОГО ЭНЕРГОНЕЗАВИСИМОГО ЗАПОМИНАЮЩЕГО УСТРОЙСТВА
856
855
ПРОГРАММНЫЕ ДАННЫЕ 847
МОДУЛЬ ОБРАБОТКИ
840
ИНТЕРФЕЙС СТАЦИОНАРНОГО ЭНЕРГОНЕЗАВИСИМОГО ЗАПОМИНАЮЩЕГО УСТРОЙСТВА
ДРУГИЕ ПРОГРАММНЫЕ МОДУЛИ 846
830
820
841
СИСТЕМНОЕ ЗАП. УСТР.
(ROM) 831
BIOS 833
(RAM) 832
ОПЕРАЦИОННАЯ СИСТЕМА 834
ПРИКЛАДНЫЕ ПРОГРАММЫ 835
ДРУГИЕ ПРОГРАММНЫЕ МОДУЛИ 836
ПРОГРАММНЫЕ ДАННЫЕ 837
ПРИКЛАДНЫЕ ПРОГРАММЫ 845
ОПЕРАЦИОННАЯ СИСТЕМА 844
810
800

ФИГ. 13