ИПМ им.М.В.Келдыша РАН

Абрау-2019 • Труды конференции

Е.М. Лаврищева, А.К. Петренко

Технология сборки интеллектуальных и информационных ресурсов Интернет

Рекомендуемая форма библиографической ссылки

Лаврищева Е.М., Петренко А.К. Технология сборки интеллектуальных и информационных ресурсов Интернет // Научный сервис в сети Интернет: труды XXI Всероссийской научной конференции (23-28 сентября 2019 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2019. — С. 469-488. — URL: http://keldysh.ru/abrau/2019/theses/93.pdf doi:10.20948/abrau-2019-93

Размещена также презентация к докладу

Технология сборки интеллектуальных и информационных ресурсов Интернет

Е.М. Лаврищева[1,2], А.К. Петренко[1]

[1]Институт системного программирования им. Иванникова РАН
[2]Московский физико-технический институт

Аннотация. Предлагается технология конфигурационной сборки интеллектуальных (Reuses) и информационных (data) ресурсов, которые разрабатываются и накапливаются в библиотеках и хранилищах Интернет, в прикладные web-системы разного назначения. Представлены средства описания ресурсов, верификации, тестирования, сборки и обеспечения безопасности и качества их взаимодействия. Обмениваемые между ресурсами данные трансформируются к требуемым форматам данных серверной среды Интернет с учетом стандарта ISO/IEC 11404 GDT в соответствии исходными данными клиента. Рассмотрена конфигурационная сборка вариантов веб-систем из компонентов повторного использования (КПИ) и приведена оценка показателей качества создаваемой веб-системы.

Ключевые слова: компонент, сервис, ресурс, система, веб-система, брокер сервисов, надежность, качество, Интернет

Technology of Assembly of intellectual and information resources Semantic Web Internet

E.M. Lavrischeva[1,2], A.K. Petrenko[1]

[1]Ivannikov Institute for System Programming of the Russian Academy of Sciences
[2] Moscow Institute of Physics and Technology (State University)

Annotation. Technology offers intelligent configuration of the Assembly (Reuses) and information (data) resources developed and accumulated in the libraries and depositories of the Internet, applied web-based systems. Means of description, verification, testing, Assembly and security of interaction of resources are presented. The transmitted data is generated to the required data formats of the server environment taking into account the ISO/IEC 11404 GDT standard and converted to the source data of the client. Proposed configuration Assembly variation of the system of CPI and assessment of the required indicators of the quality of the resulting web system.

Keywords: component, service, resource, system, web-system, service broker, reliability, quality, Internet.

1. Введение

Технология сборки разнородных модулей в сложную систему реализована в 70-80 годы прошлого столетия при создании многочисленных специализированных комплексов программ в рамках ВПК (Липаева В.В.). Реализованные средства сборки вошли в ОС ЕС (IBM-360) для всеобщего употребления, опубликованы в монографии «Связь разноязыковых модулей в ОС ЕС, Москва, 1982» и адаптированы в другие общесистемные среды — Sun Microsystems, .Net, VS.MS, IBMSphere и др. [1, 2]. Сформировалось сборочное программирование [3–5]. По словам А.П. Ершова, «сборочное

программирование обеспечивает построение уже существующих (проверенных на правильность) готовых отдельных фрагментов программ модулей (типа reuses) в сложную структуру» [6]. Интерфейс модулей описывался в специальном языке связи MDL link, а затем после 90-х появились языки описания связей объектов (IDL, API, WSDL и др.) в разных средах: *link* IBM, .Net; *make* BSD, Java (1996); *config* SPAROL, building, assembling, config Grid (2006) и др. [7–12].

В дальнейшем метод сборки был стандартизован в IEEE 828-2007, 2012 (Configuration Management) и начал широко использовался при создании прикладных систем разного назначения и на фабриках программ [13, 14]:

- Product Line /Product Family (K. Pohl).
- Конвейерная мультигенерация (K. Czarnecki).
- IBMSphere (https://www.ibm.com/developerworks/ru/websphere/newto/).
- Потоковая сборка Дж. Гринфильда.
- Диаграммная сборка use case Г. Ленца.
- Непрерывная интеграция (Continuous integration) М. Фаулера.
- Конвейерная сборка по Глушкову.
- Фабрика Веб-служб Аарона Сконарда.
- Фабричные системы BEA WebLogic Oracle, SAP NetWeaver и др.

Данные средства рассматривались в проекте РФФИ 16-01-00352 (2018). С помощью операции config этого стандарта получен экспериментальный вариант ОС Linux и представлен в Итогом отчете по проекту РФФИ. Инструмент интеграции обсуждался в докладе по теме «Информатика-70» [31–34, 37, 38] на конференции Открытых систем ИСП РАН 2018. В этом докладе отображены вопросы информатизации и технологии создания интеллектуальных и информационных систем из готовых ресурсов, а также накопления их огромном количестве в репозиториях, библиотеках reusability, Mat.Lab, SMT Интернет и др.

В работе рассматриваются подходы к интеллектуализации компонентов, сервисов (SOA) и сервисных компонентов (SCA), механизмы конфигурационной сборки и обеспечения качества и безопасности создаваемых систем в плане реализации задач проекта РФФИ 19-01-00206-2019.

2. Подходы к интеллектуализации ресурсов

В Интернете в рамках глобальной e-science Internet международным научным сообществом США создан Семантик Веб, ориентированный на создание онтологических и интеллектуальных систем [8, 12, 17, 18]. Накопленные разными специалистами знания в той или иной предметной области можно:

- *отображать знания* в Базах знаний и давать пользоваться ими другим;

– *моделировать знания* (knowledge modelling) для последующего использования при решении конкретных прикладных задач;
– осуществлять п*оиск и выбор* (knowledge retrieval) *знаний* из различных хранилищ в подмножество контента для релевантного решения конкретной задачи;
– *повторно использовать знания* (knowledge reuse), представленные в виде готовых описаний, образцов (patterns) для многократного применения в разнообразных контекстах;
– п*убликовать знания* (knowledge publishing) в стандартизованной форме для последующего распространения;
– *обновлять знания* (knowledge maintenance) и сохранять их в базах знаний Интернет;
– *приобретать знания* (knowledge acquisition) для решения задач обработки данных огромных объемов, анализа неструктурированной информации (тексты, изображения, сценарии и др.), преобразования неявных знаний в явные и интегрировать Базы знаний.

При интеллектуализации информационных данных обеспечивается:

– форматирование данных больших объемов в БД глобального масштаба, их сохранность, защищенность и безопасность;
– поиск, выбор данных и их транспортирование по запросу;
– интеграция, агрегация информации в требуемые структуры для проведения разного рода вычислений.

Для ведения больших объемов данных и обеспечения функционирования систем с доставкой по глобальной сети системных, сервисных и коммуникационных ресурсов Семантик Веб Интернет. *Семантик-Web* — эта новая информационная среда (World Wide Web), которая предоставляет семантические ресурсы, языки, средства и инструменты разработки онтологий, прикладных систем и бизнес-процессов с использованием накопленных знаний. Семантические ресурсы Веба создаются разными исследовательскими институтами США и Евросоюза с открытым кодом (http://semwebprogramming.org). Данный Веб предоставляет средства для автоматизированной обработки научных задач, больших данных в разных форматах, интеграцию данных из коллажей (Mash-Ups), поиск и композирование веб-сервисов, управление интеллектуальными агентами в

мобильных приложениях и др. Он обеспечивает решение задач с использованием многочисленных сервисов и научных достижений в области предоставления бизнес-услуг. Для использования больших объемов информации имеются словари и концептуальные модели онтологизации научных задач и приложений с помощью предметно-ориентированных языков (OWL, DSL и др.) и инструментов (FODA, Protégé, DSL Tool и др.).

2.1. Интеллектуальные ресурсы в хранилищах Интернет

К хранилищам относятся библиотеки, репозитории, базы данных (БД). Библиотеки процедур и Базы знаний, которые хранят набор функциональных готовых reuses (Reusability Libraries, 1987), КПИ для решения прикладных задач с аргументами, информационные данные большого объема, передаваемые по сообщениям между клиентом и сервером. Функциональные элементы типа reuses создают высокоинтеллектуальные специалисты в области информатики, математики, биологии и др. (например, MatLab, Demral и др.). Каждый готовый reuse, КПИ при размещении в библиотеки общего использования проверяется на правильность и качественную реализацию функции. Информационные ресурсы (ИР) сберегаются в Хранилищах данных, БД малых и больших объемов со структурированными и неструктурированными данными [17–21].

Характеристика интеллектуальных ресурсов (ИНР)

ИНР — это специфицированный алгоритм некоторой функции, использующей переменные фундаментальных (FDT) и общих типов данных (GDT) (ISO/IEC 11404), а также могут оперировать с большими данными (Big Data). ИНР взаимодействуют с другими объектами и компонентами Интернет через интерфейс в языках IDL, API, WSDL и др. [16, 32, 33].

КПИ (Reuses) — это готовый интеллектуальный программный объект, компонент, сервис, реализующий алгоритм некоторой функции предметной области, который специфицируется в стандартном языке WSDL и может многократно использоваться, если удовлетворяет функциональным требованиям прикладных областей. КПИ имеют код (implementation) с интерфейсом (interface) и схемой развертки (deployment) для выполнения. КПИ с общими переменными, функциями и поведением могут образовывать **классы** или **множества**. Внешние переменные и методы класса задаются в интерфейсе экземпляров класса.

Интерфейс — это спецификатор информационной части ИНР — КПИ, компонента, reuses для обращения к другим ресурсам и обмена данными между ними. Интерфейс задает вызов метода или функции с параметрами, которые управляют внешними переменными экземпляра класса (выборка значения *get-метод,* присвоение значения *set-метод*, *Home-интерфейс* в языке JAVA и др.) при их взаимодействии [24, 25]. Интерфейс описывается в языке WSDL и содержит:

- *название* функции (метода) и *ID* — идентификатор ресурса;

– *описание* функции средствами ЯП;
– *параметры* (входные и выходные) для передачи другим КПИ;
– *инструмент* описания КПИ (C, C++, Java, Python, Ruby и др.);
– *необязательные атрибуты* (дата, состояние, версия, права доступа, автор, срок использования и т.п.).

Язык WSDL. Для описания веб-ресурсов используется язык WSDL (Web Service Definition Language) на основе XML. Среда программирования Eclipse позволяет автоматически создавать описания на основе классов Java. В языке определены следующие основные типы данных:

1) строки (xsd:string);
2) целые числа (xsd:int, xsd:long, xsd:short, xsd:integer, xsd:decimal), числа с плавающей запятой (xsd:float, xsd:double);
3) логический тип (xsd:boolean);
4) последовательности байт (xsd:base64Binary, xsd:hexBinary);
5) дата и время (xsd:time, xsd:date, xsd:g);
6) объекты (xsd:anySimpleType).

В качестве переменных для сообщений могут использоваться последовательности, созданные из фиксированного количества переменных простых типов.

Типичный WSDL-файл имеет следующую структуру.

```
<wsdl:definitions [.]>
<!-- Декларация типов, которые используются в сервисе -->
<wsdl:types>
<element name="someMethod">
<complexType>
<sequence>
<element name="arg0" type="xsd:double"/>
<element name="arg1" type="xsd:boolean"/>
</sequence>
</complexType>
</element>
<element name="someMethodResponse">
<complexType>
</wsdl: types> …
```

Приведенное WSDL-описание задает веб-сервис MyService с единственным методом String someMethod(double arg0, boolean arg1). На его основе можно сгенерировать два типа данных, которые отвечают входным и выходным аргументам метода. Эти типы применяются в описаниях someMethodRequest и someMethodResponse — входного и выходного сообщений для операции someMethod.

Операции декларируются в описании интерфейса сервиса (декларация wsdl:portType) и дальше в описании привязки сервиса к SOAP (декларация wsdl:binding), причем во втором случае также оговаривается способ вызова (<wsdlsoap:body use="literal"/>). За счет этого при вызове операции используются те же названия параметров, что и в методе класса. В конце WSDL-файла находится декларация веб-сервиса (<wsdl:service>), в которой содержится информация относительно его расположения (параметр location).

2.2. Базовые операции над ИНР [20–25]:

К базовым операциям над ресурсами типа компонент (С) относятся:

- Building ($C_1 \cup C_2 \Rightarrow C_3$) — связывание;
- Config ($C_1 \cup C_2 \cup C_3$) — конфигурирование;
- Verification and validation (V&V) КПИ и систем;
- Creat (Cj) — образование классов компонентов и др.
- integ — объединения компонентов;
- del — удаления компонента из среды или библиотеки.

Применяются также операции рефакторинга, реинженеринга и реверсной инженерии.

2.3. Сервисные и информационные ресурсы Интернет

Web-сервисы основаны на открытых стандартах Интернет, которые широко поддерживаются на платформах Unix, Windows, IBM, Linux и др. и задаются PHP, ASP, JSP-скриптов, JavaBeans или др.

Формат запросов к Web-сервисам определяет протокол Simple Object Access Protocol (SOAP), который задается в XML и отправляется через HTPP к Web-серверу. IBM, Microsoft и средства Universal Description, Discovery and Integration (UDDI) способствуют созданию общего каталога Web-сервисов. SOAP, XML и UDDI обеспечивают надежность приложений из Web-сервисов и сервисных компонентов.

К средствам создания систем в Интернет относится сервис-ориентированная архитектура SOA (Service Oriented Architecture) и сервисно-компонентная архитектура SCA (Service-Component Architecture).

SOA задает функциональность (Functions) и качество сервисов (Quality service). Веб-сервис задается:

- на транспортном уровне (transport layer) при обмене внешними данными на коммуникационным уровне (service communication layer);
- описанием сервиса и интерфейса (service description layer) с обеспечением безопасности и защиты (авторизации, аутентификации);
- операциями публикации, поиска и вызова сервисов через реестр сервисов.

Реестр сервисов обеспечивает регистрацию, поиск и вызов сервиса по запросам от поставщика и потребителя, с описанием сервисов в языках Семантик Веб [12, 28, 30]. SOA способствует выполнению следующих операций:

1) публикации сервиса через вызов сервиса и его интерфейса;
2) поиска сервиса с помощью протокола SOAP;
3) связи UDDI с моделями CORBA, DBMS, JNET и т. п.;
4) запрос к провайдеру за опубликованными интерфейсами сервиса.

SCA дает доступ к сервисным компонентам, которые упаковываются в модуль выполнения сервиса в среде WebSphere, эквивалентного EAR-файлу J2EE. Сервисы SCA интегрируются через интерфейс JAVA и реализуются в виде классов JAVA™. В модели SCA объекты данных представлены в JAVA common.sdo.DataObject, который включает в себя метод получения свойств данных. WebSphere Integration Developer на платформе Eclipse 3.0 позволяет композировать (собирать) SCA и сервисные интерфейсы. Для вызова внешнего сервиса используется JMS, Enterprise JavaBeans или готовые сервисы. Доступ к БД системы предприятия проводится через аппарат ссылок. Веб-система собирается из сервисных компонентов, описанных в языках Python, Java, Bpel, OWL, и интерфейсов с помощью операцией config (конфигурирование) [18, 19].

Библиотека SCA — это reuses композитных сетевых сервисов (http://www.ibm.com/developerworks/websphere/techjournal) и гетерогенных данных. Они создаются в среде Java Enterprise Edition [18, 19], которая позволяет проводить:

- динамическую генерацию серверных страниц (Java Server Pages);
- определение новых КПИ в виде Enterprise Java Beans;
- преобразование КПИ к формату представления других сред;
- обмен сообщениями (Java Message Queue) со службами JMS (Java Message Service).

В SCA могут создаваться новые сервисные компоненты, объекты данных (Service Data Objects — SDO) и интерфейсы, необходимые при выполнении операции конфигурирования.

Информационные ресурсы (ИР) Интернет

Основным базовым системным ресурсом является клиент-серверная архитектура Интернет [37, 38], управляющая информационными ресурсами (ИР=IR).

Клиенту соответствует Интернет-браузер и включает ИР:

IR^1 = {Chrome, Firefox, MS Internet Explorer, Safari, Opera, …}.

На веб-сервере генерируются ответы клиенту средствами информационных систем:

IR^2 = {Internet Information Server, Apache, Tomcat, JBoss, WebSphere, WebLogic, Cloudscape}.

Интерфейс между клиентом и сервером Web-системы задаются совокупностью запросов клиента в CGI следующего вида:

WebAppInterface = {Request^p}, где *Request^p* — p-й запрос.

Web-сервер Apache и Java обеспечивают функционирование компонентов в разных ЯП и имеют вид:

IR^3_{APACHE} = {PERL, PHP, PYTHON, XML, SQL}, а для Java-среды (Tomcat, JBoss, WebSphere, WebLogic и др.) имеет вид:

IR^4_{JAVA} = {JAVA, JSP, JSTL, JSF, XML, SQL}.

Сервер Интернета включает средства:

IR^5 = {ASP, JavaScript (сервер), VB Script (сервер), C, C++, ASP.NET, C#, C++, .NET, VS .NET, J# .NET, XML, SQL}.

ИР данных включает следующие характеристики:

T^0 ={модели данных, операции хранения и доступа к данным, обработки данных и др.).

Они обеспечивают доступ к данным и взаимодействие с ресурсами типа entity в модели EJB.

Клиент-серверная архитектура Интернет является трехуровневой: клиент, сервер приложений (систем) и сервер БД, выполняемые с помощью Frontend и Backend. На уровень *клиента* выносятся простые элементы веб-систем: интерфейс, операции с данными, алгоритмы шифрования и взаимодействия с сервером приложений и т.п. *Сервер* приложений (систем) обеспечивает горизонтальное масштабирование производительности веб-систем без внесения изменений в код системы.

Сервер БД запускается с сервера приложений и выполняет обслуживание БД с обеспечением целостности, сохранности данных и доступ клиента к информации БД. Для работы с Big Data решаются задачи управления и анализа данными большого объема, а также манипулирования данными с большой нагрузкой. Клиентская часть приложения Frontend отвечает за интерфейс к программно-аппаратной части веб-систем. Frontend сервера обрабатывает приложения Интернет, которые занимают достаточное место в памяти. Backend сервера запускает серверную систему с компонентов, которые записаны на ЯП (C, C++, Python, Ruby, Java, Basic и др.) и выполняют их функции, обработку возникающих аварийных ситуаций и др.

Описание интерфейса объектов и сервисов. Для описания общедоступных веб-ресурсов используется язык WSDL (Web Service Definition Language) на основе XML. Среда программирования Eclipse позволяет автоматически создавать описания на основе классов Java. В языке определены следующие основные типы данных:

– строки (xsd:string);

- целые числа (xsd:int, xsd:long, xsd:short, xsd:integer, xsd:decimal), числа с плавающей запятой (xsd:float, xsd:double);
- логический тип (xsd:boolean);
- последовательности байт (xsd:base64Binary, xsd:hexBinary);
- дата и время (xsd:time, xsd:date, xsd:g);
- объекты (xsd:anySimpleType).

В качестве переменных для сообщений могут использоваться последовательности, созданные из фиксированного количества переменных простых типов.

Типичный WSDL-файл имеет следующую структуру.

<wsdl:definitions [.]>
<!-- Декларация типов, которые используются в сервисе -->
<wsdl:types>
<element name="someMethod">
<complexType>
<sequence>
<element name="arg0" type="xsd*:double"*/>
<element name="arg1" type="xsd*:boolean"*/>
</sequence>
</complexType>
</element>
<element name="someMethodResponse">
<complexType>
</wsdl: types> …

Приведенное WSDL-описание определяет веб-сервис MyService с единственным методом String someMethod(double arg0, boolean arg1). На его основе можно сгенерировать два типа данных, которые отвечают входным и исходным аргументам метода. Эти типы применяются в описаниях someMethodRequest и someMethodResponse — входного и выходного сообщений для операции someMethod.

Операции декларируются в описании интерфейса сервиса (декларация wsdl:portType) и дальше в описании привязки сервиса к SOAP (декларация wsdl:binding), причем во втором случае также оговаривается способ вызова (<wsdlsoap:body use ="literal"/>). За счет этого при вызове операции используются те же названия параметров, что и в методе класса. В конце WSDL-файла находится декларация веб-сервиса (<wsdl:service>), в которой содержится информация относительно его расположения (параметр location)**.**

3. Технология конфигурационной сборки ресурсов КПИ в Интернет

Метод сборки обеспечивает взаимодействие разнородных КПИ в архитектуру с взаимно однозначным преобразованием типов их входных и

выходных данных. Межмодульный интерфейс — совокупность средств формального преобразования разноязыковых КПИ. Интерфейс между КПИ — совокупность формальных средств организации обмена данными между ними [26, 33]. Метод сборки основан на математических формализмах спецификации связей (по данным и по управлению) между разнородными объектами и генерации интерфейсных модулей-посредников для каждой такой пары объектов. Каждое взаимодействие описывается с помощью операторов вызова CALL/RMI в ЯП, задающих набор фактических и формальных параметров [5].

Сущность решения задачи связи пары разноязыковых модулей объектов состоит в построении взаимно однозначного соответствия между множеством фактических параметров

$V = \{v^1, v^2, ..., v^{\kappa}\}$ вызывающего объекта и множеством формальных параметров

$F = \{f^1, f^2, ..., f^{\kappa 1}\}$ вызываемого объекта и их отображения с помощью алгебраических систем [1–4].

Каждому типу данных T_{α}^{t} языка l_{α} ставится в соответствие алгебраическая система

$G_{\alpha}^{t} = <X_{\alpha}^{t} , G_{\alpha}^{t} >$,

где X_{α}^{t} — множество значений рассматриваемого типа, а G_{α}^{t} — множество операций над объектами данного типа.

То есть операциям преобразования ТД соответствует изоморфное отображение алгебраической системы G_{α}^{t} в G_{β}^{d}.

В классе алгебраических систем $\Sigma = \{G_{\alpha}^{b} , G_{\alpha}^{c} , G_{\alpha}^{i} , G_{\alpha}^{r} , G_{\alpha}^{a} , G_{\alpha}^{z}\}$ реализованы все виды преобразований ТД (b — boolean, c — character, i — integer, r — real, a — array, z — record и др.). Доказан изоморфизм алгебраических систем и его отсутствие для множеств значений X_{α}^{t} и X_{β}^{q} . Установлено, что мощности алгебраических систем равны $|G_{\alpha}^{t}| = |G_{\beta}^{q}|$.

На основе метода сборки сформировалось сборочное программирование, охватывающее модули, объекты, компоненты, сервисы, аспекты и др. В этой парадигме реализованы универсальные формальные основы преобразования типов входных / выходных данных (простых и сложных) к общим типам данных GDT стандарта ISO/IC 11404 (примитивные, агрегатные, генерированные и неструктурированные) [1–10].

3.2. Теория преобразования типов данных КПИ

Разнородные КПИ, сервисные компоненты работают с данными, которые включают множество значений, операции над этими значениями и взаимодействие выполнения операций над ними. Первоначально для обеспечения взаимосвязей разных модулей и компонентов использовалась аксиоматика FDT (Fundamental Data Types) в классе ЯП для ОС ЕС. В 70-годы аксиоматика FDT разработана известными специалистами Э. Дейкстрой,

Н. Виртом, В. Турским, В.Н. Агафоновым и др. [5, 16]. Позднее в 1996 г. разработана аксиоматика общих типов данных (General Data Types — GDT) в стандарте ISO/IEC 11404-GDT, 1996, 2007.

Фундаментальные типы данных (FDT) — это:

- простые типы (integer, real, boolean, character, bite и др.);
- сложные (массивы, таблицы, файлы, записи, множества, деревья и др.).

Эти ТД задают базовые значения данных в программах на ЯП, которые проверяется на этапе компиляции на соответствие типов. На этапе выполнения они реализуются с помощью предикатов полиморфных типов. КПИ обмениваются данными, ТД которых должны соответствовать друг другу. В случае несоответствия ТД (например, передается целое — integer, а требуется для выполнения вызываемого модуля real), проводятся эквивалентные преобразования обмениваемых данных (integer ⇔ real).

В системе АПРОП реализован метод сборки разноязыковых модулей и библиотека примитивов по преобразованию неэквивалентных данных FDT в ЯП ОС ЕС [16, 32] . Это описано в книге «Связь разноязыковых модулей в ОС ЕС. М.: 1982. — 137с. Система АПРОП внедрена в OS IBM-360, MS.VS, Oberon и в 52 организациях страны.

FDT не охватывал все виды данных и их типов GDT, которые появились после 1992 г. Поэтому специалисты GDT включили в стандарт новые сложные типы данных — контейнеры, указатели, множества, списки, файлы, неструктурные, агрегатные данные и т.п.

Стандарт GDT (ISO/IEC 11404 GDT — 1996) прошел многолетнюю апробацию и вышел новый вариант в 2007 г. В него вошли сложные типы данных (например, агрегатные, генеративные), которые требуют их генерации к фундаментальным ТД и создания новых примитивных функций преобразования неэквивалентных ТД для новых ЯП (C, C++, Python, Basic, Ruby, Java и др.) к компьютерной и суперкомпьютерной платформам. К GDT относятся:

- примитивные ТД (real, integer, char, boolean...);
- сложные ТД (массив, стек, портфель, множество, последовательность, контейнер...);
- сгенерированные ТД (параметрические, агрегатные), генерационные;
- полуструктурированные, неструктурированные данные и расширяемые ТД.

Любые данные, которые специфицированы в КПИ и которые передаются параметры взаимодействия, требуют преобразования ТД для проведения вычисления задач, описанных в ЯП для разных платформ компьютеров. Для решения проблем преобразования данных при взаимодействии КПИ и сервисов, работающих с разными ТД ЯП предложен подход к генерации GDT<=>FDT.

Данный подход обеспечивает реализацию системных задач**:**

- спецификации внешних ТД в WSDL, сохранения их в БД и в репозиториях;
- преобразования ТД разных $ЯП_1$, …, $ЯП_{n;}$;
- представления ТД FDT к виду специальных примитивных функций;
- преобразования ТД GDT к виду ТД FDT;
- эквивалентные отображения и генерация данных GDT<=>FDT с учетом платформ современных компьютеров.

3.3. Средства обработки неструктурированных данных Big Data

Big Data — набор данных большого объема, подходов, средств, инструментов и методов представления неструктурированных огромных объёмов данных для получения данных и эффективного использования их на многочисленных узлах сети Интернет при решении задач прикладных Intelligence систем с использованием СУБД [16].

Для работы с большими объемами данных используется метод ETL (Extract Transform Load), с помощью которого производится:

- извлечение данных из внешних источников;
- трансформация и очистка данных с учетом требований прикладных систем;
- загрузка данных в хранилища данных;
- анализ данных, перенос данных из одного приложения в другое.

Основные свойства Big Data**:**

- *Горизонтальная масштабируемость.* Это обработка расширяемых больших данных и большого количества кластеров и серверов.
- *Отказоустойчивость* по отношению к сбоям на процессорах кластеров. Так, Hadoop-кластер Yahoo имеет более 42000 машин, среди которых часть машин, которые могут выходить из строя.
- *Локализация данных.* Обработка данных на машине сервера, где хранятся большие данные.
- *Изменение числа работающих на кластере* средствами MySQL Cluster. Большие данные могут быть представлены как tensors, которые управляют вычислением (например, полилинейное обучение подпространств Multilinear Subspace Learning).

Методы представления и анализа данных Big Data:

- искусственный интеллект, нейронные сети;
- предикативная аналитика, сентимент-анализ;
- статистические методы анализа, краудсорсинг и др.;
- математическая лингвистика и др.

– A/B Testing, Crowdsourcing Data Fusion;
– Integration Genetic Algorithms Machine Learning;
– Natural Language Processing;
– Signal Processing Simulation and Visualization;
– Massively Parallel Processing;
– Search-Based Applications, Data Mining;
– Multilinear Subspace Learning и др.

Для работы с большими объемами данных используется метод ETL (Extract Transform Load), с помощью которого производится:

– извлечение данных из внешних источников;
– трансформация и очистка данных с учетом требований систем;
– загрузка данных в хранилища данных;
– анализ данных, перенос данных из одного приложения в другое.

К основным свойствам Big Data относятся**:**

– *горизонтальная масштабируемость.* Это обработка расширяемых больших данных и большого количества кластеров и серверов;

– *отказоустойчивость* по отношению к сбоям на процессорах кластеров. Так, Hadoop-кластер Yahoo имеет более 42000 машин, среди которых часть машин, которые могут выходить из строя;

– *локализация данных.* Обработка данных на машине сервера, где хранятся большие данные;

– *изменение числа работающих на кластере* средствами MySQL Cluster. Большие данные могут быть представлены как tensors, которые управляют вычислением (например, полилинейное обучение подпространств Multilinear Subspace Learning).

4. Парадигмы программирования

Парадигма объектного программирования. Эта парадигма основана на графовом представлении структуры моделируемой прикладной системы [15, 21, 27]. В вершинах графа располагаются функциональные объекты и интерфейсные объекты. Разработана алгебра операций взаимодействия функциональных и интерфейсных объектов между собою и объектной средой.

Граф функций и их интерфейсов представлен на рис. 1 [22].

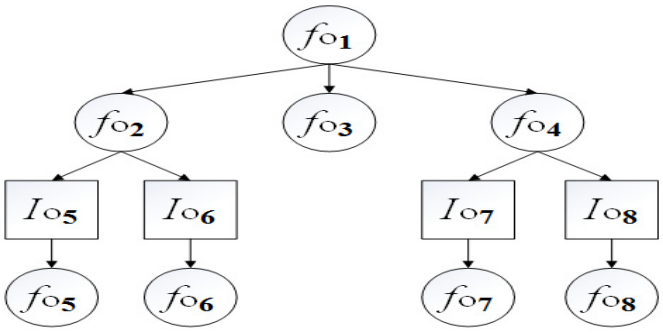

Рис 1. Граф функций и интерфейсов

На основе этого графа G можно задать несколько подсистем (M_{sysi}) обработки функций F_i:

$M_{sys1} = <F_1 (f_{O2}((Io_5, f_{O5}), (Io_6, f_{O5})); f_{O3}; f_{O4}((Io_7, f_{O7}), (Io_8, f_{O8}))>,$
$M_{sys2} = <F_2 (f_{O2}((Io_5, f_{O5}), (Io_6, f_{O5})>;$
$M_{sys3} = <F_3 (f_{O3})>;$
$M_{sys4} = <F_4 ((Io_7, f_{O7}), (Io_8, f_{O8}))>.$

С помощью треугольника Фреге для объектов задаются уникальные имена:

- знак — идентификатор, который задает сущность объектной реальности;
- денотат — сущность, смысл;
- концепт — совокупность свойств или предикатов из унарных предикатов «иметь свойство» с помощью логических связок.

Эти данные определяют сущность объекта описания для его использования в конфигурационной сборке варианта функций F_i или соответствующей программы ее реализации в среде систем VS.MS, IBMSphere, Java, Linux, Intel [20].

Сервисно-компонентное программирование. Основано на системных и сервисно-компонентных ресурсах Интернет [22–26]. Сервис Интернета — это ресурс, который реализует некоторые функции (в том числе бизнес-функции), задаваемые КПИ с технологически независимым интерфейсом с другими ресурсами. Например, сервисы транзакций, именования, безопасности и др. Они образуют службу сервисов при создании систем в среде Интернет.

Веб-сервис имеет URL-адрес, интерфейс и механизм взаимодействия с другим сервисом через протоколы Интернет или связи с другими программами, БД и деловыми операциями. Обмен данными между веб-сервисом и КПИ осуществляется с помощью XML-документов, оформленных в виде сообщений. Веб-сервисы обеспечивают решение задачи *интеграции (сборки) приложений* разной природы, являясь при этом инструментом построения распределенных систем. Основные средства описания и разработки новых систем средствами веб-сервисов:

1) язык XML для описания и построения SOA-архитектуры;
2) язык WSDL (Web Services Description Language) для описания веб-сервисов и их интерфейсов в XML, а также типов данных, сообщений и протоколов связи сервисов;
3) SOAP для определения форматов запросов к веб-сервисам;
4) SCA для создания более сложной системы на основе компонентов и сервисов;
5) UDDI для описания и интеграции сервисов, их хранения в библиотеках.

Для сборки сервисов имеется системный инструмент — ***Jopera for Eclipse*** (http://www.jopera.ethz.ch/). В нем есть набор Eclipse-плагинов для связи

различных программных элементов и реализации итеративной композиции сервисов (через маршрутизаторы SOAP и RESTful Web-сервис, Grid-сервисы, Java snipets и др.) и моделирования процессов в сети. Для поиска сервисов по их семантическим описаниям используются *Feta Client* и *Feta Engine,* где *Feta Client* — это GUI-плагин системы Интернет Taverna для описания сервиса, а *Feta Engine* для задания Web-сервиса.

Клиент серверная архитектура Интернет. Эта архитектура Интернет - трехуровневая: клиент, сервер приложений и сервер Баз Данны (БД).

На уровень *клиента* выносятся простые элементы веб-систем (приложений): интерфейс, операции с данными, алгоритмы шифрования и взаимодействия с сервером приложений и т.п. Клиент посылает запрос (request) в языке XML на сервер с помощью протоколов (HTTP, SMTP).

Сервер приложений обеспечивает горизонтальное масштабирование производительности веб-систем без внесения изменений в код выполняемой системы. Сервер принимает запрос от клиента, обрабатывает его и формирует ответ (response) клиенту.

Сервер БД запускается с сервера приложений и выполняет обслуживание БД с обеспечением целостности, сохранности данных и доступа клиента к информации БД. Для работы с Big Data на сервере решаются задачи управления и анализа данными большого объема, а также манипулирование данными с большой нагрузкой. Клиентская часть приложения Front-end отвечает за интерфейс к программно-аппаратной части веб-приложения. Front-end сервера обрабатывают приложения Интернет, которые занимают достаточное место в памяти. Back-end сервера запускает серверное приложение на ЯП (C, C++, Python, Ruby, Java, Basic и т.п.), организует их вычисление и обработки аварийных ситуаций.

4.2. Конфигурирование сервисных ресурсов Web-систем

Под *конфигурацией системы* понимается структура некоторой ее версии, включающая функции, объединенные между собой операциями связи с параметрами, задающими режимы функционирования системы [1, 2, 16–18, 31].

Версия или конфигурация системы согласно IEEE Standard 828-2012 (Configuration) включает:

- базис конфигурации — BC (Configuration Baseline);
- элементы конфигурации (Configuration Item);
- компоненты, ГОР, входящие в описание моделей M_{sys}, M_{wsys};

Управление конфигурацией (Configuration Management) заключается в наблюдении за модификацией параметров конфигурации и компонентов системы, а также в проведении систематического контроля, учета и аудита внесенных изменений, поддержки целостности и работоспособности системы. Согласно стандарту, конфигурация включает следующие задачи:

1. Идентификация конфигурации (Configuration Identification).
2. Контроль конфигурации (Configuration Control).
3. Учет статуса конфигурации (Configuration Status Accounting).
4. Аудит конфигурации (Configuration Audit).
5. Трассировка изменений конфигурации на этапах сопровождения и эксплуатации системы;
6. Верификация компонентных сервисов по моделями систем и веб-систем.

При конфигурационной сборке ГОР используется модели систем и модель характеристик MF (Model Feature). КПИ хранятся в репозиториях или библиотеках системы. Они выбираются их библиотек, адаптируются и интегрируются в систему. Основную роль в этих процессах выполняет конфигуратор, например в (http://7dragons.ru/ru). Он обеспечивает конфигурационную сборку разнородных КПИ и их интерфейсов в вариант готового продукта, которые будут сохраняться в репозиториях.

М*одель среды конфигуратора* включает:

- графовую структуру системы из ресурсов;
- модель вариантов системы;
- конфигурационную модель и операции сборки КПИ;
- аудит конфигурации системы;
- верификация моделей и ресурсов;
- оценку качества КПИ и систем.

Конфигуратор, исходя из моделей системы, MF (Feature Model) и указанных в них внешних компонентов и интерфейсов собирает их в веб-систему. Операция config создает конфигурационный файл, тестирует его выполнение и способствует определению приемлемых показателей качества и безопасности.

5. Обеспечение качества систем

Согласно стандарту ISO/IEC 12207 ЖЦ ПО регламентируется планирование, управление качеством и оценку затрат на создание системы. На этих процессах ЖЦ проводится анализ достижения качества; верификация и валидация (V&V) ресурсов и оценивание степени достижения отдельных показателей качества; тестирование готовой системы; сбор данных об отказах, дефектах и др. ошибках; оценивание надежности по соответствующим моделям надежности с учетом результатов тестирования [33, 35, 36].

Модель качества стандарта ISO/IEC 9000 (1-4) Quality задает шесть показателей (характеристик) q_1 — q_6 (q — quality) качества:

q_1 — функциональность (functionality),
q_2 — надежность (reliability),
q_3 — удобство (usability),
q_4 — эффективность (efficiency),

q_5 — сопровождаемость (maintainability),
q_6 — переносимость (portability).

Каждая характеристика q_i рассчитывается по специальным формулам и метрикам стандарта. Надежность оценивается согласно полученных на процессе тестирования ошибок, дефектов и отказов в ПО и по соответствующим моделям надежности (оценочным, измерительным и др.).

Данные по всем показателям качества q_1 — q_6 оцениваются по формуле:

$$q_1 = \sum_{j=1}^{6} a_{1j} m_{1j} w_{1j}$$

где a_i — атрибуты каждого показателя качества (i = 1..6); m_i — метрики каждого атрибута качества; w_i — вес каждого атрибута показателя качества системы. Полученные значения по показателям модели качества входят в сертификат качества продукта. Варианты оценки показателей качества сконфигурированных систем приведены на сайте ИТК, в [18, 19].

6. Заключение

Технология сборки Web-систем из интеллектуальных и сервисных ресурсов сформировалась в рамках проекта РФФИ №16-01-00352 «Теория и методы разработки изменяемых программных систем» [21–23]. В данной работе рассмотрены базовые понятия конфигурационной сборки систем, веб-систем из готовых ресурсов — КПИ, reuses, service-components, Web-services, которые описываются в современных ЯП (C, C++, JAVA, Python, Ruby, Basic и др.), а их интерфейсы в языке WSDL. Приведены операции конфигурационной сборки систем из готовых сетевых ресурсов Дан анализ подходов к определению интеллектуальных элементов в среде Семантик Веб и открытых моделей SOA и SCA для представления этих элементов с помощью системных сервисов, серверов, клиентов и средств обработки данных Big Data в Cloud Computing.

Рассмотрены фундаментальные FDT и общие GDT типы данных, которые используются для описания передаваемых данных с эквивалентным преобразованием неструктурированных данных среды Интернет. Отработана сборка верифицированных сервисов, компонентов и КПИ в вариант системы в Jopera for Eclipse, проведено тестирование системы и оценивание показателей качества с помощью стандарта ISO/IEC 9000 (1-4) Quality. Результаты сборки варианта веб-системы отображены в ПТК на сайте http://www.ispras.ru/7dragons.ru/. В рамках проекта РФФИ 16-01-00209 (2019–2021) готовится вариант создания веб-сайтов в среде Интернет из готовых ресурсов (системных, сервисных, сервисных компонентов), обеспечивающих функционирование систем в Cloud Computing с большими данными.

Литература

1. Лаврищева Е.М., Грищенко В.Н. Связь разноязыковых модулей в ОС ЕС. — М.: Финансы и статистика. — 1982. — 136 с.
2. Лаврищева Е.М. Интерфейс в программировании // Проблемы программирования. — Киев: 2006. — №2. — С.126–139; Formal fundamentals of component interoperability in programming / Lavrischeva K.M. // Cybernetics and Systems Analysis. — 2010. — Volume 46. — Issue 4. — P. 639–652. — doi:10.1007/s10559-010-9240-z
3. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. — Киев: Наук. думка, 1991. — 236 с.
4. Липаев В.В., Позин Б.А., Штрик А.А. Технология сборочного программирования. — М.: Радио и связь, 1992.
5. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Основы индустрии программных продуктов. — Киев: Наук. Думка, 2009. — 371 с.
6. Ершов А.П. Опыт интегрального подхода к актуальной проблеме ПО // Кибернетика. — 1984. — С. 11–21; Научные основы доказательного программирования // Вестник АН СССР. — 1984. — № 10. — С. 9–19.
7. Web Services Description Language (WSDL) 1.1//W3C Note 15 March 2001. — URL: http://www.w3.org/TR/wsdl.
8. Semantic Web. — URL: http://www.w3.org/2001/sw/ .
9. Reference Model for Service Oriented Architecture 1.0. — URL: http://docs.oasis- open.org/soa-rm/v1.0/soa-rm.html
10. Web Services Resource 1.2 (WS-Resource). — URL: http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf.
11. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) // W3C Recommendation. 27 April 2007. — URL: http://www.w3.org/TR/SOAP12-part1.
12. Semantic Web programming / S. Hebeler, M. Fisher, R.Blace, A/Peter-Lopes. — Willey Publishing Inc., 2008. — URL: http://semwebprogramming.org.
13. Лаврищева Е.М. Теория и практика фабрик программ // Кибернетика и системный анализ. — No. 6. — 2011. — С. 145–158; Theory and practice of software factories / K.M. Lavrischeva // Cybernetics and Systems Analysis. — 2011. — Volume 47. — Issue 6. — P. 961–972.
14. E.M. Lavrischeva. Classification of software engineering disciplines // Cybernetics and Systems Analysis. — 2008. — Volume 44. — Issue 6. — P. 791–796.
15. Лаврищева Е.М. Теория объектно-компонентного моделирования программных систем // Препринт ИСП РАН 29, 2016. — С. 1–52. — URL: ISBN978-5-9-474-25..
16. Иванников В.П., Дышлевый К.В., Мажелей С.Г и др. Распределенные объектно-ориентированные среды // Труды Института системного программирования РАН. — Том 1. — 2000. — С. 100–121.

17. Лаврищева Е.М. Рыжов А.Г. Применение теория общих типов данных стандарта ISO/IEC 12207 GDT применительно к Big Data // Conference “Actual problems in science and ways their development”, 27 December 2016, http://euroasia-science.ru/. — С. 99–110.
18. Лаврищева Е. М., Карпов Л. Е., Томилин А. Н. Семантические ресурсы для разработки онтологии научной и инженерной предметных областей // Научный сервис в сети Интернет: Труды XVIII Всероссийской научной конференции (19–24 сентября 2016 г., г. Новороссийск). — М.: ИПМ им. М.В. Келдыша, 2016. — с.126–138. — doi:10.20948/abrau-2016-8.
19. Лаврищева Е. М., Карпов Л. Е., Томилин А. Н. Подходы к представлению научных знаний в Интернет науке // Сб. XIX Всероссийский научной конференции «Научный сервис в сети Интернет», Новороссийск, 18–23 сентября 2017. — С. 310–326.
20. Лаврищева Е.М. Компонентная теория и коллекция технологий для разработки индустриальных приложений из готовых ресурсов // Труды 4-й научно-практической конференции «Актуальные проблемы системной и программной инженерии», АПСПИ-2015, 20–21мая 2015. — С. 101–119.
21. Лаврищева Е.М. Программная инженерия. Тема 1. Теория Программирования. 50 с.; Тема 2. Технология программирования, 48; Тема 3. Базовые основы программной инженерии. — 52с. / /Методические пособия, Москва, МФТИ, 2016.
22. Лаврищева Е.М., Петренко А.К. Моделирование систем и их семейств // Труды ИСП РАН. — М.: 2016. — Том 28. — Вып. 6. — С.180 –190.
23. Кулямин В.В., Лаврищева Е.М., Мутилин В.С., Петренко А.К. Верификация и анализ вариабельных операционных систем // Труды ИСП РАН. — М.: 2017. — Том 28. — Вып. 3. — С.189–209.
24. Лаврищева Е.М. Программная инженерия. Парадигмы, Технологии, CASE-средства программирования, 2 изд. — М.: Юрайт, 2016. — 280 с.
25. Лаврищева Е.М. Программная инженерия и технология программирования сложных систем // М.: Юрайт, 2017. — 431с.
26. Островский А.И. Подход к обеспечению взаимодействия программных сред JAVA и MS.Net. — Проблемы программирования. — 2011. — №2. — с.37–44; Лаврищева Е.М. Взаимодействие программ, систем и операционных сред // Там же. — 2011. — № 3. — С. 11–23.
27. Лаврищева Е.М., Колесник А.Л., Стеняшин А.Ю. Объектно-компонентное проектирование программных систем // Теоретические и прикладные вопросы. — Весник КНУ, серия физ.-мат. наук. — Киев, 2013. — №4. — С. 150–164; Ekaterina Lavrischeva, Andrey Stenyashin, Andrii Kolesnyk. Object-Component Development of Application and Systems. Theory and Practice // Journal of Software Engineering and Applications. — 2014. — Vol. 7. — No. 9. — doi:10.4236/jsea.2014.79070.

28. Ekaterina M. Lavrischeva. Assembling Paradigms of Programming in Software Engineering // Journal of Software Engineering and Applications. — 2016. — Vol. 9. — No. 6. — P. 296–317. — doi:10.4236/jsea.2016.96021.
29. Lavrischeva Ekaterina. Ontological Approach to the Formal Specification of the Standard Life Cycle // Science and Information Conference-2015, July 28–30, London, UK, www.conference.thesai.org. — P. 965–972.
30. MacGregor S.D., Sykes D.A. Practical Guide to Testing Object-oriented software. — Addison-Wesley, 2001.
31. Sayyad A. S., Ingram J., Menzies T., Ammar H. Scalable product line configuration: a straw to break the camel's back // Proc. of IEEE/ACM 28-th International Conference on Automated Software Engineering (ASE 2013). — IEEE, 2013. — P. 465–474. — doi:10.1109/ASE.2013.6693104.
32. Лаврищева Е.М., Рыжов А.В. Подход к созданию систем и сайтов из готовых ресурсов // Научный сервис в сети Интернет: труды XX Всероссийской научной конференции (17–22 сентября 2018 г., г. Новороссийск). — М.: ИПМ им. М.В. Келдыша, 2018. — С. 321–346. — ISBN 978-5-98354-046-0.
33. E.M. Lavrischeva, A.K.Petrenko. Informatics-70. Computerization aspects of programming software and informatic systems technologies // Proc. ISP RAS. — 2018. — Vol. 1. — Issue 2. — P. 3–4.
34. Лаврищева Е.М., Аветисян А.И., Петренко А.К. Информатика. ЭВМ-70. Анализ и аспекты развития. — Доклад на конференции ISPRAS-2018.
35. E.M. Lavrischeva. The Scientific Basis of Software Engineering // International Journal of Applied and Natural Sciences (IJANS). — Vol. 7. — Issue 5. — Aug–Sep 2018. — P. 15–32. — ISSN(P): 2319-4014; ISSN(E): 2319-4022.
36. Лаврищева Е.М., Пакулин Н.В., Рыжов А.Г., Зеленов С.В. Анализ методов оценки надежности оборудования и систем. Практика применения методов // Труды ИСП РАН. — Том 30. — Вып. 3. — 2018. — С. 99–120. — doi:10.15514/ISPRAS-2018-30(3)-8.
37. Лаврищева Е.М. Современные системы искусственного интеллекта // Симпозиум «Искусственный интеллект: различные подходы к его воплощению (компьютерно-сетевые, нейросетевые, квантовые технологии и другие)», Москва, 13 октября 2018. — Федеральное государственное образовательное учреждение при Правительстве Российской Федерации. Колледж информатики и программирования. — Презентация доклада.