Technology of Assembly of Intellectual and Information Resources Internet

E.M. Lavrischeva[1,2][0000-0002-1160-1077],
A.K. Petrenko[1,4] [0000-0001-7411-3831] and B.A.Pozin[3,5] [0000-0002-0012-2230]

[1]Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., 109004, Moscow, Russia
[2] Moscow Institute of physics and technology, 9, Institutsky per., Dolgoprudny, 141700, Moscow region, Russia
[3]NRU Higher School of Economics, 20 Myasnitskaya Ulitsa, 101000, Moscow, Russia
[4] Lomonosov Moscow State University, GSP-1, Leninskie Gory, 119991, Moscow, Russia
[5]EC-leasing, Building 1, 125 Warshavskoye Shosse, 117587, Moscow, Russia
lavr@ispras.ru, apetrenko@hse.ru, bpozin@ec-leasing.ru

Abstract. The technology of Assembly of intellectual (Reuses) and information (data) resources which are developed and saved up in libraries and storages of the Internet, in applied web-systems of different function is offered. The intellectual tasks of subject areas of knowledge in the Semantic Web in mind service, software and system resources used in creating Web systems are presented. Methods of assembling resources (link, make, assemble, weaver, kconfig) are considered for modern environments. The problem of data exchange and generation to the formats of the global network environment according to ISO/IEC 11404 GDT is defined. The concept of a network resource collector is proposed as a generalization of ways to assemble resources with functional security, resource protection and quality assessment of Web-systems. Perspective paradigms of programming of different areas of knowledge are defined.

Keywords: Resource, Service, Web System, Configuration, Functionality, Security, Reliability, Quality.

1 Introduction

The technology of Assembly of heterogeneous modules was implemented in the 70-90 years of the XX- centuries in the creation of numerous specialized complexes of programs within the military-industrial complex (Lipaev V.V.) for VPK Minradioprom USSR. The method of assembling modules in different Programming Languages (PL) (ALGOL-60, Fortran, Cobol, PL/I, Modula, etc.) was based on the developed primitive 64 functions of converting non-equivalent exchanged data between modules in the APROP of the EU OS (IBM-360) [1, 2] and transferred to Ernutz (1985). APROP system is implemented in 52 organizations of the USSR. Module Assembly tools are adapted to other system environments-Sun Microsystems, .Net, VS.MS, Oberon, IBMSphere, Intel, Unix, etc. [1-5]. Assembly programming has been formed in the country [3-5]. According to A. P. Ershov, "Assembly programming provides the construction of already existing (checked for correctness) ready-made individual fragments of modules (reuses type) into a complex structure" [6]. The interface of modules was described in a special language of communication description in link, and then in the 90s there were languages of communication description of modules (IDL, API, ABI, WSDL, etc.) and implemented in different system environments: *link* IBM, .Net, Corba; *make* BSD, GNU, c*make* MSBuild(.NET), ApacheAnt (Java, C#)); *config* REST, CDR, SPAROL; building, assembling, config SWMS, Grid, Semantic-OGSWA, OML, JSON; *weaver* BEA WebLogic Oracle, SAP Net, ASP and etc. [7–12]. The experimental version of OS Linux kernel within RFBR project 16-01-00352 (2018) was obtained using the method of configuration Assembly of modular resources and presented in the final report of R & d 11020510073 "Theory and methods of development of variable software systems" dated 05.03.19. The generalization of different variants of the Assembly method became software factories that contribute to the industrialization of application systems for different purposes and the task is to ensure the quality and safety of the functioning of resources and systems (RFBR Project 19-01-000206-19) [13, 14].

2 Approaches to the Intellectualization of Systems and resources

Web Internet Semantics are focused on the creation of intelligent systems [8, 12, 17, 18]. In them the process of scientific knowledge management consists in:

- display knowledge in Knowledge Bases and give them to others to use;
- modeling (knowledge modeling) knowledge for their use in solving specific application problems;
- search and selection (knowledge retrieval) of knowledge from different repositories in a subset of content for a relevant solution to a specific task;
- reuse (knowledge reuse) knowledge in the form of ready-made descriptions, samples (patterns), models in a variety of contexts;
- knowledge publishing (knowledge publishing) of knowledge in a standardized form for later distribution;
- updating (knowledge maintenance) and saving knowledge in Knowledge Bases;
- acquisition (knowledge acquisition) of knowledge for the analysis of unstructured information (texts, images, scripts, etc.),
- transformation of implicit knowledge into explicit, solving problems of processing huge amounts of data and storing them in the Knowledge Base.

An aspect of intellectualization of information resources (documents, texts, tables, pages, etc.) are processes:

- formatting large amounts of data in a database of global scale to ensure the safety, security and security;
- search, select data fragments and transport them according to user requests;

- analysis, integration, aggregation of information (for example, documents, in the required structures for processing at different levels of management of organizations of the country, etc.) for the calculation of tasks, etc.

2.1 Intellectualization of Knowledge

Modern means of knowledge representation in terms of descriptive logic include: FaCT (in LISP), FaCT++ (in C++), CEL, KAON-2 (JAVA), MSPASS (C) and Pellet (JAVA), etc. The necessary component of knowledge representation are applied web-services focused on collective problem solving in the subject areas of knowledge in modern environments (problem-solving environments, PSE) in the world community. KBS (Knowledge-based systems), CommonKADS, etc. are used in the process of creating knowledge models of subject areas. They provide the construction of knowledge libraries containing elements of problem solving, which can then be re-used in other areas of knowledge. Based on the fact that the PL program is some formal or mathematical text, the system of text information and terminology management (System Quirk – SQ, http://www.computing.surrey.ac.uk/SystemSQ. This system is focused on the creation and maintenance of terms in the terminology database (TBD) and knowledge bases (BZ), as well as the organization of collections of texts on the computer using the tools Virtual Corpus, KonText, Ferret, Grid, etc. To maintain large amounts of data and ensure the functioning of systems in the global Internet, system, service and communication resources are used, including web Semantics tools-this information environment in the World Wide Web, which provides semantic resources, languages, tools and tools for developing ontologies of subject areas of knowledge, application systems and business processes using the accumulated knowledge in the environment http://semwebprogramming.org. In the web Semantics environment, automated processing of scientific tasks, big data in different formats, integration of data from collages (Mash-Ups), search and composition of web services, management of intelligent agents in mobile applications, etc. is carried out. One of the tools of knowledge intellectualization in Semantics about different subject areas of knowledge (mathematics, physics, biology, medicine, etc.) is ontology (www.semantic.web.ontology). Ontology is a conceptual model building tool for knowledge domains / domains. The conceptual model includes objects, object classes, data structures, relationships, and relationships (theorems, constraints) for a particular field of knowledge.

There is experience in developing the domain ontology SE-Life Cycle ISO / IEC 12207 -2007 in OWL (Web Ontology Language) in Protégé 2.3 environment with the output result in XML. The approach to automation of Life Cycle (LC) management was presented at the International conference "Science and Information-2015" (www.conference.thesai.org) in London and at the XVII all-Russian scientific conference-2015-2016 "Scientific service on the Internet" in Novorossiysk, KIAM by M. V. Keldysh [18-20, 29, 33]. With the help of the obtained ontology of LC, it is possible to solve individual problems using numerous services and resources of other subject areas of knowledge of the Internet. Dictionaries, conceptual models and applications with the help of domain-oriented languages (OWL, DSL, etc.), tools FODA, Protégé, DSL Tool, KBuild, Kconfig can be used to apply large amounts of information in solving scientific problems of the domain of knowledge (http://www.Sap.org) etc.

2.2. Intellectual and Information Resources

Intelligent and information resource Assembly technology is the process of assembling resources (modular elements in different subject areas – SA), data structures, procedures, classes, components and services. Resources are developed for different applied fields of knowledge (mathematics, medicine, biology, genetics, etc.). They are stored in libraries, Internet repositories, and system-wide environments (IBM, MS.Net, Unux, Intel, etc.). Repositories include libraries, repositories, databases (DB). Procedure libraries, knowledge Bases, Reusability Libraries and other. Functional elements of the reuses type are created by highly intelligent specialists in the field of computer science, mathematics, biology, etc. (for example, MatLab, Demral, etc.). Each ready-made reuse when placed in a shared library is checked for the correctness and quality of the implementation of the functions of the knowledge domain.

Intellectual resources (INR) are reuses, Components Ready Using (CRU) that reflect the knowledge of professionals in different fields of knowledge. Any the knowledge domain Pro function is described in the PL as a module (object, component, service, etc.) and can interact through the link Assembly operator and Call/RPC/RMI operations in the texts of PL programs, in the parameters of which data are specified in IDL, API, ABI, WSDL, etc. CRU process data that belong to the fundamental (FDT) and General data types (GDT) of the ISO/IEC 11404 standard, and can also operate with large amounts of data (Big Data). INR interact with modules, objects or service components of the Internet through the interface specified in the languages IDL, API, ABI, WSDL, etc. [16, 32, 33].

Reuse is a ready-made intelligent software object, component, service that implements the algorithm of a function in some subject area of knowledge. It is specified in the standard WSDL language and can be reused if it meets the functional requirements of individual subject areas. CRU as Reuses has the code (implementation) with the interface (interface) and the scheme of deployment (deployment) for their performance. CRU, Reuses can have common variables that form classes or sets. External shared variables and class methods are specified in the class instance interface.

Information resources (IR) are data of different volumes, including Big Data, presented in Databases, files, directories, tables, documents, etc. IR are processed by service, system, client and server services of the Internet and placed in data Warehouses, databases of small and large volumes with structured and unstructured data [17-21]. An interface is a specifier of the information part of the INR-CRU, component, reuses for accessing other resources and exchanging data between them. An interface contains a method or function call (RPC/RMI) with parameters that control external variables of the class instance (fetching the value of the get method, assigning the value of the set method, Home interface in JAVA, etc.) when they interact [24, 25].

The interface is described in WSDL and generally contains:

- function (method) name and resource ID;

- description (specification) of a function by means of PL;
- parameters (input and output) for data transmission to other CRUS;
- CRU description language (C, C++, Java, Python, Ruby, etc.);
- optional attributes (date, status, version, access right, author, term of use, etc.).

The programming environment (Eclipse, Protege, etc.) allows you to automatically create resource descriptions based on JAVA classes.

The following main data types are defined:

1) strings (xsd: string);

2) integers (xsd: int, xsd:long, xsd:short, xsd:integer, xsd:decimal), floating point numbers (xsd:float, xsd:double);

3) logical type (xsd:boolean);

4) byte sequence (xsd:base64Binary, xsd:hexBinary);

5) date and time (xsd: time, xsd:date, xsd:g);

6) the object type (xsd:anySimpleType).

As variables can be used sets, sequences, including a fixed number of variables of simple types. A typical WSDL file has the following structure:

```
<wsdl:definitions> <!-- Declaration of the types that are used in the service -->
<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> …
```

This WSDL description specifies a MyService web service with a single string method someMethod (double arg0, boolean arg1). On its basis, you can generate two types of data that correspond to the input and output parameters of the method. These types are used in the someMethodRequest and someMethodResponse descriptions-input and output messages for the someMethod operation. The operations are declared in the service interface description (WSDL:portType Declaration) and then the description of binding the service to the SOAP Protocol (Simple Object Access Protocol) through the WSDL:binding Declaration is given. This sets the call to <wsdlsoap:bodyuse= "literal" /> with the parameters specified in the class method. At the end of the WSDL file is a web service Declaration (<wsdl:service>) that contains location information <location parameter>.

2.3. Service and Information Resources Internet

Web services are based on open Internet standards, which are widely supported on Unix, Intel, Windows, IBM, Linux, etc. platforms and are specified by PHP, ASP, JSP script, JavaBeans, etc. The format of requests to Web services defines the SOAP Protocol, which is specified in XML and sent via HTPP to the Web server. IBM, Microsoft, and Universal Description, Discovery and Integration (UDDI) contribute to the creation of a common web services catalog. SOAP, XML, and UDDI also ensure that applications from Web services and service components function reliably.

The means of creating application systems in the Internet include service-oriented architecture SOA (Service Oriented Architecture) and service-component architecture SCA (Service-Component Architecture). SOA defines the functionality and Quality of services at the levels:

– transport (transport layer) for the exchange of external data at the communication layer (service communication layer);

– service and interface descriptions (service description layer) with security and protection (authorization, authentication);

– operations of publishing, searching and calling services through the services registry.

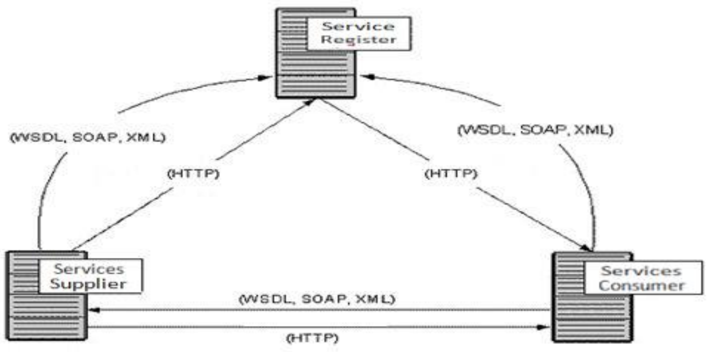


Fig. 1. Service services in a network environment.

Consider the services of SOA and SCA. Operations on SOA services are as follows:

1) publishing the service through the service call and its interface;

2) search for a service using the SOAP Protocol;

3) UDDI communication with CORBA, DBMS, JNET, etc.;

4) request to the provider for published service interfaces.

The SCA service provides access to service components that are packaged into a service module in a WebSphere environment equivalent to a J2EE EAR file. SCA services are integrated through the JAVA interface and implemented as JAVA™classes.

In the SCA model, data objects are represented in JAVA common.sdo.A DataObject that includes a method for defining data properties. WebSphere Integration Developer platform Eclipse 3.0 allows you to compose (assemble) SCA services and service interfaces. You use JMS, Enterprise JavaBeans, or off-the-shelf services to call an external service. Access to the database of the system is carried out through the link server. The web system is assembled from the service components described in Python, JAVA, BPEL, OWL and interfaces using the SAP kconfig configuration operation [18, 19]. The SCA library contains a set of reuses of composite network service components and heterogeneous data.

(http://www.ibm.com/developerworks/websphere/techjournal).

These services can be created in the Java Enterprise Edition environment [18, 19], which allows you to:

– dynamic generation of server pages (Java Server Pages);

– definition of CRU as Enterprise Java Beans;

– conversion of CRU parameters to the presentation format of other media;

– exchange messages (Java Message Queue) with JMS (Java Message Service).

SCA can create new service components for problem areas of knowledge, as well as data objects (Service Data Objects-SDO) and interfaces that provide data transfer to other CRUs.

You use JMS, Enterprise JavaBeans, or off-the-shelf services to call an external service. Access to the database from the system (enterprise / company) is carried out through the link apparatus. The Web system you create is assembled from the service components described in Python, Java, BPEL, OWL, and interfaces using the SAP kconfig configuration operation (Fig. 2).

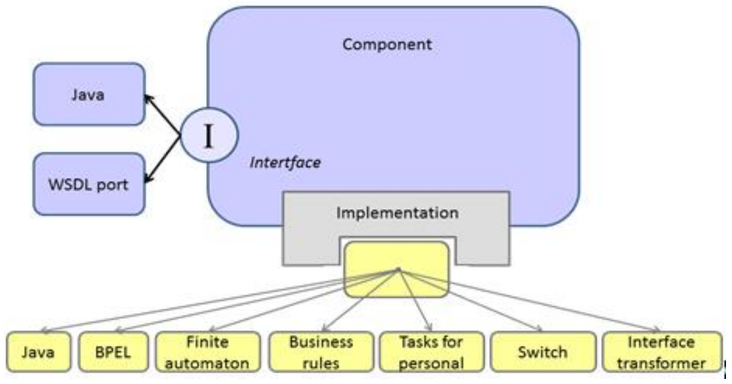


Fig. 2. General scheme of IBM SCA service.

When working with web services with data from the Big Data class, the following Internet tools are used: IBM WSDK+WebSphere; Apach Axis+Tomcat; Apach Axis+ Classfish; Microsoft.Net + IIS, etc.

2.4. Internet System Resources

The main system resources of the Internet include the client-server architecture of the Internet [37, 38], which is three-level: client, application server (systems) and database server. The client side is responsible for the interface and processing of Internet applications, which are written in PL (C, C++, Python, Ruby, Java, Basic, etc.) and perform application functions, processing of emerging emergencies, security and quality control, etc.

The database server starts from the application server and performs database maintenance to ensure the integrity, safety of data when the client accesses the database information. To work with Big Data, you perform tasks of analyzing large amounts of data, as well as manipulating data with small and large loads.

The client corresponds to the Internet browser (Chrome, Firefox, MS Internet Explorer, Safari, Opera, etc.). It sends messages to the server through an interface like this: WebAppInterface = {Requestp}, where Requestp is the PTH request. Request processing is performed by server components such as: Internet Information Server, Apache, Tomcat, JBoss, WebSphere, WebLogic, Cloudscape. Apache Web server and JAVA provide INR processing in modern PL and have the form:

IR APACHE = {PERL, PHP, PYTHON, XML, SQL},

IRJAVA = {JAVA, JSP, JSTL, JSF, XML, SQL} in system components envirinment (Tomcat, JBoss, WebSphere, WebLogic и др.).

Server Internet have the tools: ASP, JavaScript, VB Script, ASP.NET, .NET, J#.NET, XML, SQL) and languages (C, C++, C#, Python, Ruby, Cobol, Prolog and ets.).

Data IR includes: data models, data storage and access operations, data processing, etc. They perform data access and interaction with entity resources in the EJB model. When working with large amounts of data, the ETL (Extract Transform Load) method transfers data from one application to another through the client-server architecture and extracts data from external sources, analyzes it and transforms it to meet the requirements of conceptual models and executes it on the Internet.

3. Technology Build Resources on the Internet

The technology is based on the method of assembling resources and communication interface (assembler) of heterogeneous CRUs into an architecture with a one-to-one conversion of input and output data types. An intermodule interface is a set of tools for formal conversion of multilingual CRUs. The interface between CRUs includes a set of formal means of data exchange between modular elements [26, 33]. The Assembly method is based on mathematical formalisms of specification of connections (by data and by control) between heterogeneous modular elements and generation of interface intermediate modules for each pair of connected elements of the Graph structure of the system. Each link is defined by a call operator of type CALL/RPC/RMI in PL with a set of actual and formal parameters [5].

An Assembly technology is a method, means, tools, and process for assembling heterogeneous resources that consists of defining:

- schemes, graphs and models of the subject area of knowledge; - repository modules (CRU, Reuses, procedures, etc.);

- module algorithms with CALL / RPC /RMI module operations; – interface modules with the parameters of the transmitted data;

- storage operations and selection of modules, CRU from repositories; - verification, testing of modules, their interfaces, etc.;

- functional reliability and quality of resources and systems from them.

In practice of programming systems of Assembly of modules by means of operations of Assembly were formed: link APROP (CORBA); make of compiler programs BSD, GNU, MSBuild; assemble, Kconfig of system, service and computing resources EuroGrid; config of service SOA, SCA of resources in Semantic Web; weaver BEA WebLogic Oracle, SAP Net. Consider these build systems.

3.1. Link Assembly in APROP

Link Assembly in APROP connects modular resources in different YAP (Algol, Fortran, Cobol, Prolog, Modula, etc.) through the interface in MDL, IDL (Interface Definition Language) type stub, skeleton (CORBA). On the basis of link, data and control links of modules are generated and non-equivalent data exchanged between them is converted [1]. The Assembly of modules in PL is implemented in the system APROP OS EC EM (IBM-360), which implements 64 primitive functions of converting non-equivalent data in PL [1, 2]. The APROP system was transferred to Ernuc (1985) and implemented in 52 organizations in Russia, Central Asia, the Baltic States, Moldova, GDR, etc. the Assembly Method is adapted to the environment: Sun Microsystems, .Net, VS.MS, IBMSphere, Unix, Intel, etc. [3–5]. Scheme of connection of program C and modules A, B through the interface (Fig. 3) has the form:

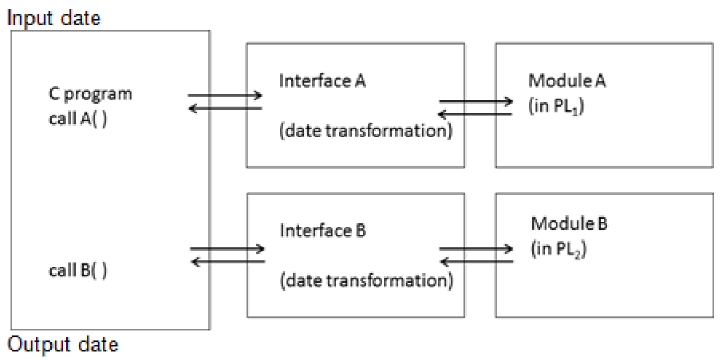


Fig. 3. The scheme of Assembly of programs and modules in APROP.

This scheme is a client program C, in which CALL A and CALL B are specified to call modules A and B and transmit data to them through the interface modules A’, B’ with the necessary conversion to the form of data A and B, which are performed on the server. The server accepts the message from the programs and starts of the modules A, b with demarshaling arguments. After executing modules A, B, the result is obtained, which the server packages into a message and sends it back to program C.

An example of communication modules P1, P2 in PL (Pascal, Delphi) through the interface Unit 1 is given in Table 1.

Table 1. Assembly scheme programs P1 and P2.

Program P1 in Pascal	Program Unit1	Program P_2 in Delphi
program P1;	unit Unit1;	program Pro-
uses Crt;	interface uses	ject1;
var fact, i,	Windows, Messages, Sys-	uses Forms,
N : longint;	Utils,	Unit1 in
begin	Variants, Classes,	'Unit1.pas'
clrscr;	Graphics, Controls,	{Form1};
writeln	Forms, Dialogs, StdCtrls;	{$R *.res}
('Vvedit N:	Type TForm1 = class(TForm)	begin
');	Label1: TLabel; Edit1:	Applica-
readln (N);	TEdit;	tion.Initialize;
fact:=1;	Label2: TLabel;	Applica-
for i:=1 to	Edit2: TEdit;	tion.CreateForm(T
N do	Button1: Tbutton	Form1, Form1);
begin	procedure But-	Application.Run;
fact:=fact*i	ton1Click(Sender:	end.
;	TObject);	
end;	private {Private declara-	
writeln('Fac	tions}	.
torial 4isla	public {Public declara-	
',	tion} var Form1: TForm1;	
N, ' = ',	implementation {$R *.dfm}	
fact);	procedure	
readln;	TForm1.Button1Click(Sender	
end.	: TObject);	
	var i, fact : Integer;	
	begin	
	fact:=1;	
	for i:=1 to	
	StrToInt(Edit1.Text) do	
	begin fact:=fact*i;	
	end;	
	Ed-	
	it2.Text:=IntToStr(fact);	
	end; end.	

Assembly according to the scheme Fig.3 implemented on a variety of software and hardware complexes MIC (PROMETHEUS, RUZA, YAUZA, etc.) of the USSR 1981–1990 [1, 4]. As a result, the Soviet Union formed the domestic technology of Assembly programming. Lipaev V. V. in [4] said that "Assembly programming is based on a large number of CRU and is a fundamentally new industrial method of assembling products into complex software products while improving the efficiency and quality of products."Ershov A. P. in [6] said: "Assembly programming carries out the construction of programs from existing (tested for correctness) ready-made KPI (reuses) by assembling them into a complex structure." The interface of modules was described in a special language MDL, and in the 90s there were languages for describing the links of modules IDL CORBA; API, ABI (BSD, GNU, Grid); WSDL (Semantic-OGSWA, etc.) [7–12].

3.2. The build in CORBA

In this system, modules are built in different link by type APROP using the object model (OM) and assembled through interface intermediaries - Stub-client and Skeleton-server, performing the same functions as data transfer and processing between objects written in the NPS class (Fig. 4), based on the library of primitive functions APROP with the addition of primitive functions for PL: C++. Ada-95, Basic, Prolog, Smaltalk et al.

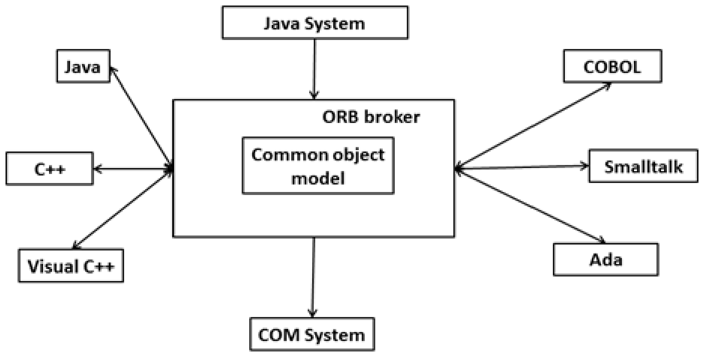


Fig. 4. Assembly modules in PL by ORB broker (stub and skeleton).

3.3. Assemblies in .Net environment

The Assembly operator processes the transmitted data between modules using the CTS system (Common Type System, Fig. 5).

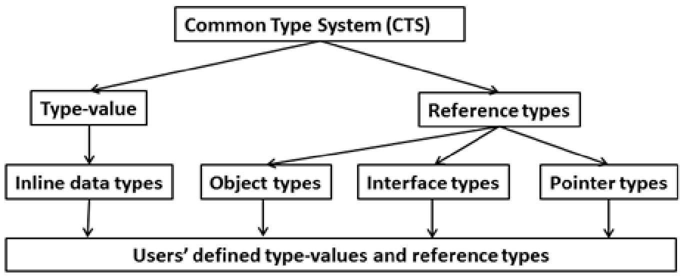


Fig. 5. System for processing common data types in .Net.

The objectives of the Assembly included:
- display value types, reference and inline data types in computer data format;

- description of common Language Runtime (CLR) routines and processing of integer data, floating point, strings, bits;
- specifications of classes, interfaces, built-in data types, enumerations, etc., specified in the CLS (Common Language Specification);
- translation of data in PL to intermediate MSIL, which is converted to CPU code for execution on different architectures of network computers.

Implementation of this scheme on the platform MS.NET is given for PL in [25]. When you build objects, the IDL interface description begins with the interface keyword, followed by the interface name, parameter types, and operations (op_dcl) for calling objects:

```
interface A { ... }
interface B { ... }
interface C: B, A { ... }.
```

The operation parameters (op_dcl) in the interface assignment are: 1) data type (type_dcl); 2) the constant (const_dcl); 3) the name of the exception (except_dcl) that may occur during the execution of the object method; 4) parameter attributes (attr_dcl). The description of data types (TD) begins with the typedef keyword, followed by the base or constructed type and its identifier. As a constant, there can be some value of the given type or an expression composed of constants. TD and constants are described as fundamental data types: integer, bool, string, float, char and etc.

The description of op_dcl data transfer operations includes:

1) the name of the operation interface;
2) parameter list (zero or more);
3) types of arguments and results, otherwise-void;
4) control parameter or description of an exceptional situation, etc.

Attributes of transmitted parameters begin with service words: in – when sending a parameter from the client to the server; out - when sending parameters-results from the server to the client; inout – when passing parameters in both directions (from the client to the server and back).

The interface description for one object can be inherited by another object and then this description becomes basic. An example of this is given below:

```
const long l=2
interface A {
void f (in float s [l]); }
interface B {
const long l=3 )
interface C: B, A { }.
```

Interface C uses interface B and A. this means that interface C inherits the description of data types A and B, which are external to C. But the syntax and semantics remain unchanged. According to the given example-the operation of the function f in the interface C is inherited from A.

3.4. Compiler Building

This method performs processing of the source code in PL (C++, Fortran, Go, Perl, PHP, Pyrhon, Java) with make, cmake on BSD systems, GNU Unix, and MS and Assembly interface in the API (Application Programming Interface) an executable file in native code (libraries of modules of Microsoft, Unix, etc.), as well as the integration of the integration of compiled modules with ABI (Application Binary Interface) to an intermediate language MSIL bytecode in the environment .NET (Fig.6).

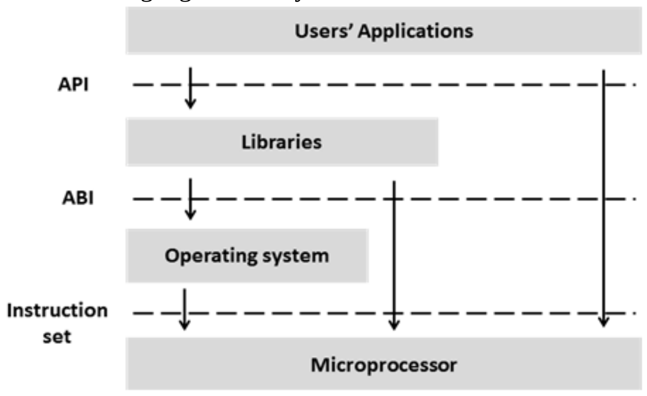


Fig. 6. Processing of the application interface ABI and API.

The process of building resources in PL with an interface in API and ABI through make executes the GNU Build system Assembly script generator in MSBuild (.NET), Apache Ant (Java), Apache Maven translators with Java, C#, Scala; Scons(C, C++, Java, Fortran,Tex), etc.:

- the instruction set of the processor to the register file, stack and memory;

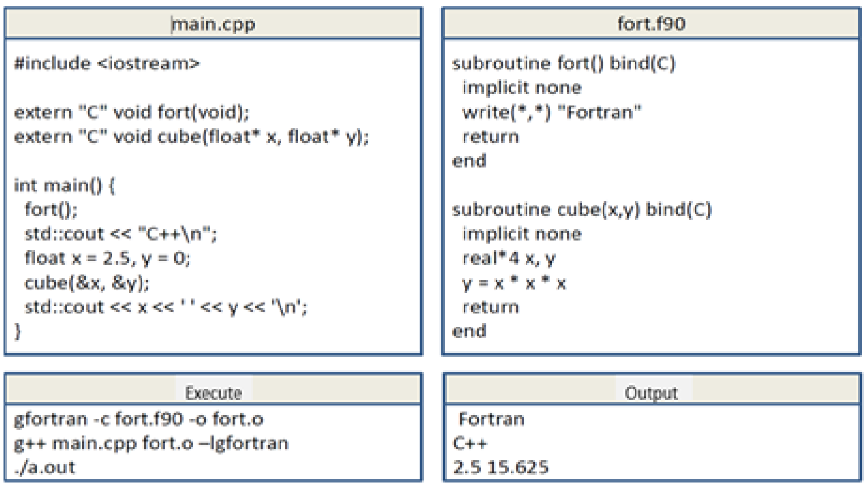
- the size and location of the base data that the computer processor works with; - binary file formats, libraries and executables.

Program communication operations in API and ABI for C++ and Fortran (Fig.Seven): add executable (exec_name source1 source2 ...) - create executable file from source code files source1, source2, etc.;

- add library (lib_name source1 source2 ...) # create lib_name library from source code files source1, source2, etc.

- target include_directories (target PUBLIC dir1 dir2 ...) # include dir1, dir2, ... di-rec-tories in the header search list when building an executable from the library;

- target_link_libraries(target lib1 lib2 ...) # link (include in Assembly) library elements lib1, lib2 for target.



main.cpp

```
#include <iostream>

extern "C" void fort(void);
extern "C" void cube(float* x, float* y);

int main() {
  fort();
  std::cout << "C++\n";
  float x = 2.5, y = 0;
  cube(&x, &y);
  std::cout << x << ' ' << y << '\n';
}
```

fort.f90

```
subroutine fort() bind(C)
  implicit none
  write(*,*) "Fortran"
  return
end

subroutine cube(x,y) bind(C)
  implicit none
  real*4 x, y
  y = x * x * x
  return
end
```

Execute

```
gfortran -c fort.f90 -o fort.o
g++ main.cpp fort.o –lgfortran
./a.out
```

Output

```
 Fortran
C++
2.5 15.625
```

Fig. 7. Example of linking of C++ and Fortran programs.

3.5. Build in GRID

Build in GRID (www.gloubos.org) provides interaction of resources through calls of RPC/RMI in programs on YAP by means of operations assembling, config, make and establish communication of resources among themselves at work with Cloud Computing and Big Data. A system GRID (Fig. 8) provides processing and management of software, service and other Web resources of the global network. The Assembly of heterogeneous network and system resources into Web systems, applications and packages that work with data of different volumes is carried out in the ETICS GRID system (Fig. 8). Resources are described in different modern PL. The basic entities of systems, services, and applications, as well as relationships, are described in the CIM (Common Information Model). ETICS uses a standardized description of the project for the main objects: Project, Subsystem, and Component. Project. A subsystem can contain only Components. They use the CIM data model to define relationships between different objects and to describe objects and relationships between them, and to pass the results of their execution to queries. Storing and maintaining data is based on a relational type model in MySQL.

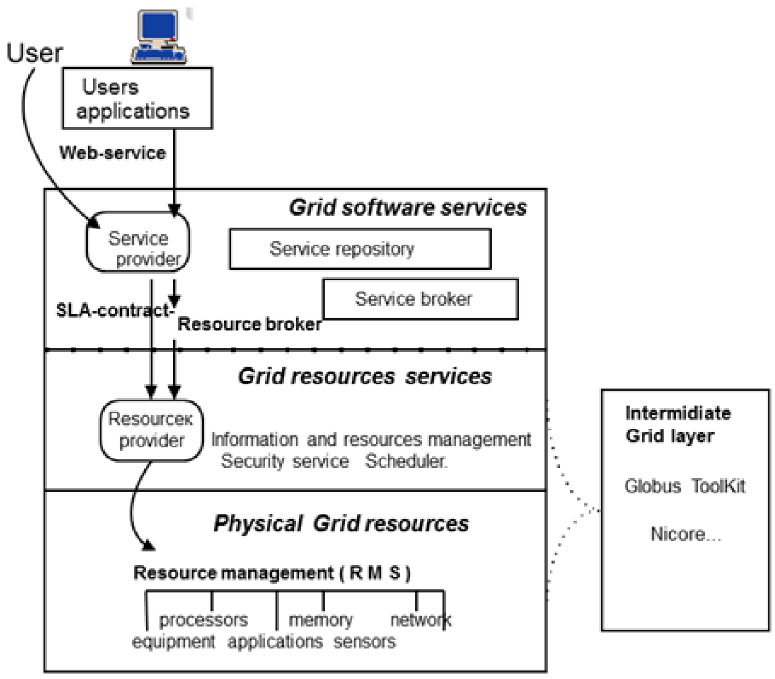


Fig. 8. Basic ETICS systems of Grids as Factory software.

The data model description is based on the following basic assumptions: 1) each component contains a description of the information (name, license, repository URL, etc.) and a global unique identifier-ID (GUID); 2) the configuration object contains version information, repository link, GUID, framework platform and system configuration link; 3) each component is checked (checkout) compiled element, test commands and GUIDs, as well as communication with the configuration; 4) when defining the configuration and platform, each object displays a GUID, its properties, runtime, and dependencies, which can be declared statically or dynamically. A static dependency is a relationship between two configurations, a dynamic dependency is a relationship between a configuration and a module.

ETICS by function is a modern software factory. It is based on a set of features, services, and package-building procedures that can be combined by plugins, provide job management, and provide access to the OS, CPU architecture, compilers in the YAP, and tools to define different packages and test them after resource Assembly and deployment. Many functional plug-ins provide contract verification, execution tests of different system elements, documentation generation, maintenance of ready-made CPIs in the operational or static repository.

The main task of the ETICS system is to convert some system components for an alternative platform of heterogeneous computer environment by means of links from 16 -, 32-bit platforms to 64-bit Grid environment platform. Unicore (Uniform Interface to Computingresources) deals with software interface support issues (www.unicore.org). the WSRF (Web Service Recurse Framework) tool provides security And protection of data resources and systems. The Grid environment includes system components that provide simulation of physical experiments, Cloud computing and processing of large amounts of data:

- Globus Toolkit (GGF) for SOA and SCA; Condor (www.cs.wisc.edu/condor); WebFlow (http://www.npac.syr.edu/users/haupt/WebFlow/); Nimrod/G
- (www.csse.monash.edu.au/~davida/nimrod/references.htm);
- Gridbus Data Grid Service Broker;GRACE (Grid Architecture for Computational Economy);
- Grid- Port and Grid Port Toolkit) for management creation of Web-systems.

3.6. Industrial Resource Assembly Systems

Build by type of software factories: IBM WebSphere, Microsoft Biz Talk, BEA WebLogic Oracle 10g, SAP NetWeaver and IVK "Jupiter". They contain CASE tools for assembling (link) resources (services, components and data through the data broker stub and skeleton of the CORBA system, overeating data using the Workflow Protocol with safety and quality control (Table 2) [24].

Table 2. CASE–tools of assemble resources

Platforms	Forms	The purpose of the platforms
IBM WebSphere	Corporation IBM	J2EE application server, data exchange brokers, KPI, portal, workflow/BPM, EII, SCA
Microsoft Biz Talk 2004 and .Net	Corporation Microsoft	COM application server, data brokers, RGB delivery, portal, workflow/BPMN
BEA WebLogic	Corporation "BEA Systems" in "Oracle" (from 2008)	J2EE application server, data exchange brokers, GORE, application server, portal, workflow/BPMN
Oracle 10g	Corporation "Oracle" http://www.oracle.com	J2EE application server, data brokers, GORE, portal, workflow/BPMN, EII tools
SAP NetWeaver	Corporation SAP http://www1.sap.com/www.sap.ru	J2EE/ABAP application server, data exchange brokers, portal, BPMN tools
IVK "Upiter"	Company IVK (Russian)http://www.ivk.ru	Data brokers, KPIs, runtime, certification, data protection

In the software factories of these systems, the architecture is modified according to the required service resource codes of the SOA model in Visual Studio Industry Partners (VSIP). This uses the Guidance Automation eXtensions (GAX) and Guidance Automation Toolkit (GAT) models in Visual Studio. GAX is an execution environment in VSIP using recommendation packages. There are two service options: web service ASP.NET (ASMX) for Windows Communication Foundation (WCF) in the

.NET Framework 3.0. Versions for the WCF web service are available from the developers of the GotDotN SOFTWARE factory. It includes enterprise processes and packages of documentation and recommendations for an application such as Global Bank. This is similar to the SCA model in IBM WebSphere.

Build (integrate) different programs on CASE tools: Make, Apache Ant, Apache Maven, Gradle, etc. Make is a cross-platform automation system for building systems from source code. It generates Assembly control files, such as a Makefile on Unix systems, for Assembly through the make operation (see 3.4.) Apache Ant-JAVA-utility to automate the process of building a software product. Ant is a platform-independent analogue of the UNIX make utility, but using the JAVA language adapted for JAVA projects. The most important difference between Ant and Make is that Ant uses XML to describe the build process and its dependencies, while Make has its own Makefile format. XML file (or build.xml) builds executable programs.

3.7. Configuring Resource and Component

Assembly in GFM is a way to assemble resources in an industry. K. Chernetsky created a generational multiassembly on Product Line / Product Family. (see Generative programming. Methods, tools, application, 2005.- 750s.). The GDM model introduces new concepts to represent the subject areas of knowledge: the problem space, the solution space, and the configuration base of the system family (SF). The task space displays the concepts of Software Systems / Family (SSP), members and their General characteristics in the MF (Feature Model), as well as functions and tasks that are described by the GPL (General-Purpose Language) or domain-oriented languages DSL, UML2. The solution space is the components, frameworks, templates, and CRUs of implementing the tasks of members FS of the SSF family.

The rules of description, generation of components and selection of CPIs for individual SSF tasks are included in the configuration database. In this case, the framework is equipped with variable model parameters, which can lead to excessive fragmentation and the appearance of "many small methods and classes". The framework provides dynamic linking of aspects and components in the process of implementing variability between different applications. Design patterns enable the creation of reusable solutions in different program systems (PS). ActiveX and JAVABeans component technologies, as well as new build mechanisms, are used to define aspects such as synchronization, remote interaction, data protection, etc.

The results of the description of the SSF model in these spaces are in the configuration knowledge base (BZ): connections and characteristics (functional or non-functional) specified in the corresponding MF model of family members and construction operations are combined into a common PS or SSF. Knowledge about system configuration in the form of abstractions of General and special purpose, CRU and Reuses c results of their testing, measurement and evaluation are displayed in BZ.

3.8. Build on the Model Transformation and Configuration

The transformation model is described in a DSL. The main mechanism of transition from the description of models to the initial result is the transformation of domain concept descriptions into an intermediate language of DSL-space of solutions, and then into a language of implementation of components taking into account the platform, where ready-made components and/or new tasks are located.

In the software factories of these systems, the architecture is modified according to the required service resource codes of the SOA model in Visual Studio Industry Partners (VSIP). This uses the Guidance Automation eXtensions (GAX) and Guidance Automation Toolkit (GAT) models in Visual Studio. GAX is an execution environment in VSIP using recommendation packages. There are two service options: web service ASP.NET (ASMX) for Windows Communication Foundation (WCF) in the .NET Framework 3.0. Versions for the WCF web service are available from the developers of the GotDotN SOFTWARE factory. It includes enterprise processes and packages of documentation and recommendations for an application such as Global Bank. This is similar to the SCA model in IBM WebSphere. Build (integrate) different programs on CASE tools: Make, Apache Ant, Apache Maven, Gradle, etc. Make is a cross-platform automation system for building systems from source code. It generates Assembly control files, such as a Makefile on Unix systems, for Assembly through the make operation (see 3.4.) Apache Ant-JAVA-utility to automate the process of building a software product. Ant is a platform-independent analogue of the UNIX make utility, but using the JAVA language adapted for JAVA projects. The most important difference between Ant and Make is that Ant uses XML to describe the build process and its dependencies, while Make has its own Makefile format. XML file (or build.xml) builds executable programs.

3.9. The Standard Configuration of the Service Resources

Under the configuration of the system is understood the structure of some of its versions, including functions, combined with each other communication operations with parameters that specify the modes of operation of the system [1, 2, 16–18, 31]. The system version or configuration according to IEEE Standard 828-2012 (Configuration) includes:

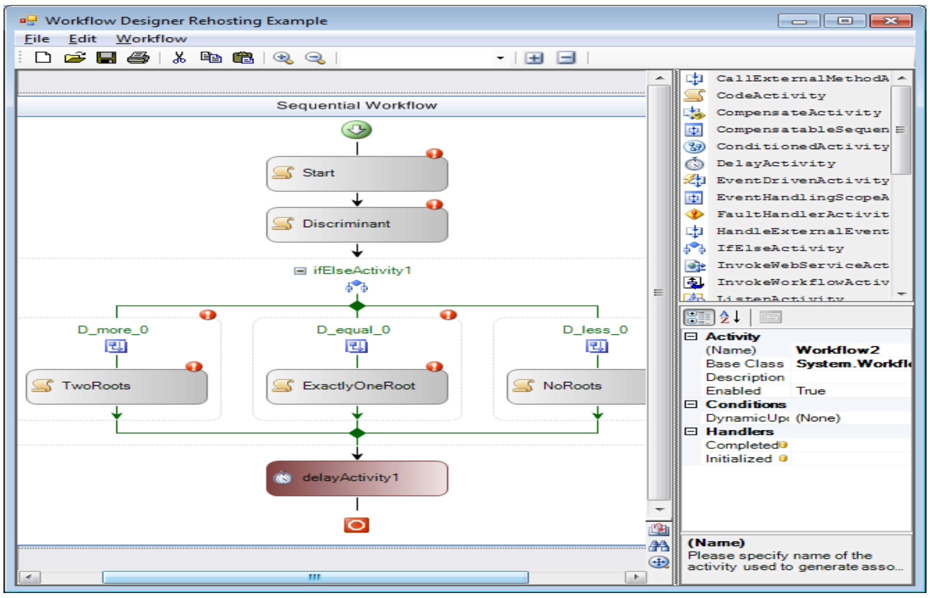
- Configuration Baseline-BC;
- Configuration Item;
- Components included in the description of Msys, Mwsys models.

Configuration Management is to monitor the modification of configuration parameters and components of the system, as well as to conduct systematic monitoring, accounting and auditing of changes, integrity and health of the system on the processes:

1. Configuration Identification;
2. Configuration Control;
3. Configuration Status Accounting;
4. Configuration Audit;
5. The tracing configuration, maintenance and operation of the system;

6. Verification of component services according to the model of the system or the web system.

The system model and MF (Model Feature) model are used in the configuration Assembly of the CRU. RUs are selected by their libraries, adapted and integrated into the system. The main role in these processes is performed by the Configurator, for example, Fig. 9 in (http://7dragons.ru/ru).



Activity
(Name) Workflow2
Base Class System.Workfl
Description
Enabled True
Conditions
DynamicUp (None)
Handlers
Completed
Initialized

(Name)
Please specify name of the activity used to generate asso...

Fig. 9. Configuration modeler.

The Configurator manages the creation of a variant of the finished product and stores it in the repository. The model of the environment of the Configurator includes: граф graph structure of the system from resources; – system options model; – configuration model and CRU Assembly operations; аудит system configuration audit; – verification of models and resources; – quality assessment of KRU and systems.

The Configurator builds the web system using the config operation (http://7dragons.ru/ru) on the model of Msys, MMF and specified domain resources and their interfaces. The configured file is then evaluated for quality and security.

Linux OS variant configuration build OS Linux contains more than 10 000 variables and a large number of functional system components that provide processing of various tasks for the functioning of any application systems running OS Linux is presented in the diagram (Fig.10) and performed in the master's work [34, 46].

At the top level is the user space, which houses user applications and the GNU C library (glibc) with system call interfaces to communicate with the kernel. The Linux OS kernel contains more than 11,000 program elements and is common to all processor architectures supported by Linux. The next level – architecture-dependent BSP (Board Support Package) kernel code defines a specific processor and platform architecture. Linux OS can be compiled for a huge number of different processors and platforms with different architectural constraints and needs. Linux OS can run on a processor with or without a memory management unit (MMU).

The main components of the Linux OS kernel include the following: - kernel function system call interface; - manage processes or threads using the application programming interface (API) via SCI to create a new process and scheduler algorithm whose running time is independent of the number of threads; - memory management with the participation of hardware that establishes the correspondence between virtual memory and physical memory; - definition of a virtual file system (VFS), which provides a General abstraction of the interface to file systems and serves for switching between SCI and kernel file systems; - the network stack has a multilevel structure of the protocols themselves.

Internet Protocol (IP) is the basic Protocol of the network layer and is located below the transport Protocol TCP (Transmission Control Protocol). Above TCP is the socket layer and is a standard API to the network subsystem and to hardware devices. From the OS kernel, the necessary components are selected and an MF model is created with the basic characteristics of the OS components and the Msys system model, including many functional MF, interface Mio and Md of working with the data of the Linux OS base kernel.

The types build different resources in Sections 3.1–3.9 (link, make, config, assembling, building, weaver, integration) for creating application, service, information and technical systems in the network to the Global Internet, is based on the theory of conversion of complex data types ISO/IEC 11404 GDT to a more simple FDT for calculations on the created systems. The General provisions of the data type conversion theory of this standard are discussed below.

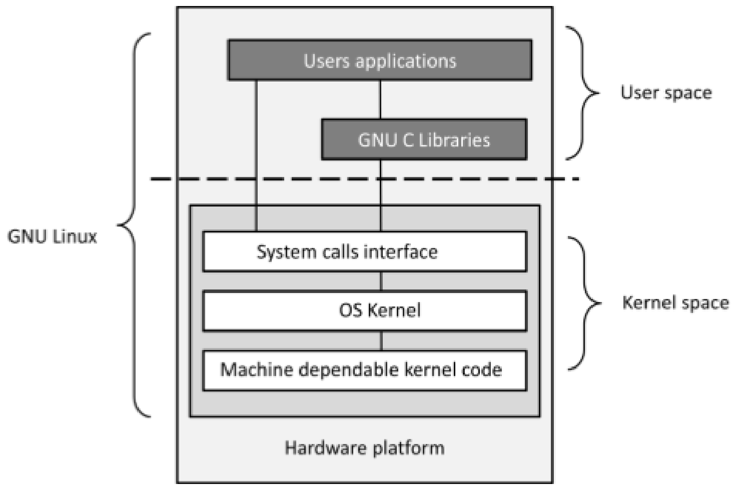


Fig. 10. The structure of kernel OS Linux.

3.10. Theoretical Basis of TD FDT and GDT Transformation

Heterogeneous CRUs, service components work with data that includes multiple values, operations on those values, and the interaction of performing operations on them. Initially, the module Assembly method used the axiomatic FDT (Fundamental Data Types) in the YAP class for the EU OS. This axiomatics was developed by well-known specialists E. Dijkstra, N. Wirth, V. Tursky, V. N. Agafonov and others in the 70s [5, 16]. Later in 1996, the axiomatics of General Data Types (GDT) was developed in ISO/IEC 11404 — GDT, 1996, 2007.

Fundamental data types (FDT) include: – simple types (integer, real, boolean, character, bite, etc.); complex (arrays, tables, files, records, sets, trees, etc.). These parameters set the basic values of data in the programs in the PL, which is checked at the compilation stage for type compliance. At runtime, they are implemented using polymorphic type predicates. CRUs exchange data, the values of which must correspond to each other. In case of nonconformity of TD (for example, integer is passed, and it is required for execution of the called module real), equivalent transformations of the exchanged data (integer <-> real) are carried out by means of primitive functions of APROP interface library for FDT [1, 16, 32].

To the problems of conversion of TD are different in of modules or components:

a) discrepancy of quantity (formal and actual) of parameters or their incorrect description;

b) inconsistency of types of transmitted parameters or their values for computer formats;

c) absence of direct and inverse transformations of parameters, etc.;

d) no transformation of complex to simple data types. The new GDT standard works in information and software systems with data such as containers, templates, patterns, protocols, pointers, sets, lists, sequences, non-structural, heterogeneous, etc.

This standard ISO/IEC 11404 GDT-1996 has passed many years of testing and released a new version in 2007. the standard ISO / IEC 11404 GDT-1996 includes aggregate, generative, extensible, etc. TD, which require their generation to the fundamental TD for subsequent use in the ready resources of the Global Internet to perform calculations. In this regard, it is necessary to develop new primitive conversion functions of non-equivalent TD for new YAP (C, C++, Python, Basic, Ruby, JAVA, etc.) and other types (Fig. 12).

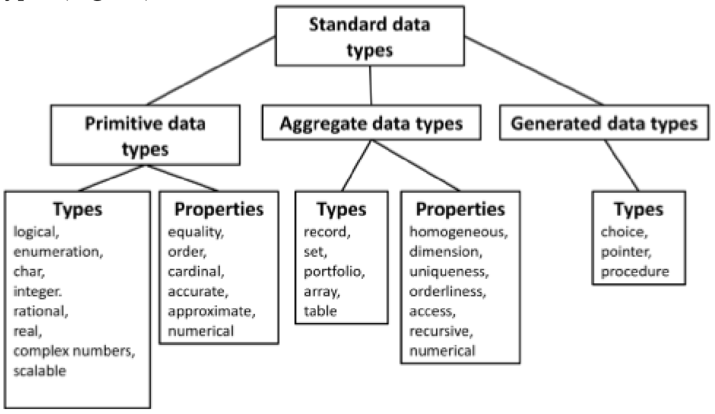

Standard data types
Primitive data types
Aggregate data types
Generated data types
Types
logical, enumeration, char, integer. rational, real, complex numbers, scalable
Properties
equality, order, cardinal, accurate, approximate, numerical
Types
record, set, portfolio, array, table
Properties
homogeneous, dimension, uniqueness, orderliness, access, recursive, numerical
Types
choice, pointer, procedure

Fig. 11. The data types in GDT standard.

To solve the problems of data transformation according to the GDT and FDT standard when linking (interacting) CRU and services specified in different modern NPS, an approach to the generation of GDT<=>FDT is proposed (Fig. 13), the Basis of this approach is the problem of converting common data types to simpler FDT.

To support resource Assembly, you must implement the following GDT<=>FDT data conversions: specifying external TD'S in WSDL, saving them in DB and in repositories; convert TD GDT to TD in new PL1,..., PL n ; implementation of TD GDT with special primitive functions and taking into account the format of framework platforms; equivalent mapping and generation of GDT<=>FDT data taking into account the platforms of modern computer and Internet cluster systems where resources will operate.

Semi-structured, unstructured and expandable TD appear in connection with the exploration of the bowels of the earth, space and ocean. Practical processing of these new data structures will be required in order to solve effectively applied tasks in different application areas, especially those working with Big Data

Approaches to processing unstructured Big Data – a set of large-volume data, as well as approaches, tools and methods of presenting unstructured huge amounts of data to obtain data and effectively use them on numerous nodes of the Internet when solving applied intelligence is and it systems using DBMS [17, 25, 26]. To work with large amounts of data, the ETL (Extract Transform Load) method was formed, with which the:

– extract data from external sources;

– transform and clean data to meet your requirements;

– load data into data stores; analyze data and transfer data from one application to another, etc.

The main properties of Big Data include:

– horizontal scalability of processed big data and a large number of clusters and servers;

– fault tolerance against the failure on processor clusters and nodes in the network. Thus, the Yahoo hadoop cluster tool has more than 42,000 computers, among which some of them may fail;

– data localization and processing on servers where big data is practically stored to solve the relevant tasks; change the number of workers on the cluster using MySQL Cluster tools;

– Big Data can be represented as tensors that control computation (e.g. multilinear subspace learning, Etc.).

System tools for processing Big Data include:

NoSQL is a database of non-relational and distributed data with open source and horizontal scalability, effectively support random read, write and versioning. MapReduce is a distributed computing model that is used in parallel computing with big data in computer clusters. Hadoop is a freely distributed set of utilities, libraries, and frameworks for developing and running distributed applications (including MapReduce programs) running on clusters of hundreds or thousands of nodes.

SMB-big data processing systems within Cloud computing. Big Data is analyzed by means of statistical and dynamic methods of analysis of artificial intelligence, neural networks, mathematical linguistics; 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, etc.

4. Service-component Processing of Internet Resources

To build services there is a Jopera tool for Eclipse (http://www.jopera.ethz.ch/), containing a set of Eclipse-plugins for communication of various software resources and implementation of iterative composition of services (via routers SOAP and RESTful Web-service, Grid-services, Java snipes, etc.) and modeling of processes on the Internet. To search for services by their semantic descriptions, Feta Client and Feta Engine are used, where Feta Client is a GUI plugin of the Internet Taverna system to describe the service, and Feta Engine to specify the Webservice.

4.1. Client Server Architecture

Internet to work with resources uses three-level Internet Architecture: client, application server and Database server (DB) [32]. At the client level, the resources of web systems are transferred: the interface, data operations, encryption algorithms and interaction with the application server, etc. the Client sends a request (request) in XML to the server using protocols (HTTP, SMTP). The application server receives a request from the client, processes it, and generates a response to the client by horizontally scaling the performance of web systems without making changes to the code of the running system.

The database server starts from the application server and performs database maintenance to ensure the integrity, safety of data and client access to database information. When working with Big Data on the server, the tasks of managing and analyzing large data, as well as manipulating data with a large load are solved. The front-end client part of the application is responsible for the interface to the hardware and software part of the web application. Front-end servers handle Internet applications that take up sufficient memory space. Back-end server runs a server application in PL (C, C++, Python, Ruby, Java, Basic, etc.), organizes their calculation and processing of emergencies.

4.2. Ensuring the Quality of Web systems

According to the ISO / IEC 12207 standard, the software LC standard regulates the planning, quality management and cost estimation for the creation of the system. In these processes, lifecycle analyses quality assurance; verification and validation (V&V) resources and evaluating the degree of achievement of individual quality indicators; test of the finished system; data collection about the failures, defects etc. and failures; assessment of the reliability on the respective reliability models based on the results of testing [33, 35, 36].

The ISO/IEC 9000 (1-4) Quality model defines six characteristics q_1– q_6 (q – quality):

q_1 — functionality,
q_2 — reliability,
q_3 — usability,
q_4 — efficiency,
q_5 — maintainability,
q_6 — portability.

Each q_i characteristic is calculated according to special formulas and metrics of the standard. Reliability is evaluated according to the errors, defects and failures in the SOFTWARE obtained during the testing process and according to the corresponding reliability models (evaluation, measurement, etc.). Data on all quality indicators q_1 — q_6 are evaluated by the formula:

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

where a_{ij}-attributes of each quality indicator (i = 1..6); m_i – metrics of each quality attribute; w_i – weight of each attribute of the system quality indicator. The obtained values for the indicators of the quality model are included in the product quality certificate. Options for assessing the quality of configured systems are given on the ITC website [18, 19].

5. Conclusions

The technology of assembling Web-systems from intellectual and service resources was formed in the framework of RFBR project No. 16-01-00352 "Theory and methods of development of variable software systems" [21–23]. In this paper, we consider the basic concepts of configuration Assembly of systems, web systems from ready-made resources-KPI, reuses, service-components, Web-services, which are described in modern PL (C, C++, JAVA, Python, Ruby, Basic, etc.), and their interfaces in the WSDL language. Operations of configuration Assembly of systems from ready network resources are given.

The analysis of intellectual and information resources in the environment of Semantics of Web and open models of SOA and SCA for representation of these elements by means of system services, servers, clients and means of data processing of Big Data in Cloud Computing is given. Assembly operations (link, make, cmake, config, weaver) are considered on the example of existing systems (APROP, CORBA, BSD, GNU, MSBuild; EuroGrid; Semantic Web; BEA WebLogic Oracle, SAP Net, etc.). The theoretical apparatus of resource Assembly using fundamental and General data types and mechanisms of transformation of non-equivalent unstructured data (including Big Data) transmitted through the Assembly operations in the Internet environment based on the ISO/IEEE 11404 GDT standard is presented.

The configuration Assembly of resources and processes of verification, testing and quality assessment using the ISO/IEC 9000 (1-4) Quality standard of the received product is described. The RFBR project 16-01-00209 (2019-2021) considers the client-server architecture of the Internet and the configuration Assembly of intellectualization and information ready-made resources with the provision of functional security, reliability and quality of Web systems for working with Big Data in the Cloud Computing environment. The graph theory, new programming paradigms and ontology of mathematical modeling of applied problems for vital areas of society (medicine, biology, physics, mathematics, economics, etc.) will become the main tools of smart machines of the 21st century.

References

1. Lavrishcheva, E.M., Grishchenko, V.N.: Sviaz raznoiazykovykh modulei v OS ES. Finansy i statistika, Moscow (1982).
2. Lavrischeva, E.M.: Formal fundamentals of component interoperability in programming. Cybernetics and Systems Analysis 46(4), 639–652 (2010), https://doi.org/10.1007/s10559-010-9240-z.
3. Lavrishcheva, E.M., Grishchenko, V.N.: Sborochnoe programmirovanie. Nauk. dumka, Kiev (1991).
4. Lipaev, V.V., Pozin, B.A., Shtrik, A.A.: Tekhnologiia sborochnogo programmirovaniia. Radio i sviaz, Moscow (1992).
5. Lavrishcheva, E.M., Grishchenko, V.N.: Sborochnoe programmirovanie. Osnovy industrii programmnykh produktov. Nauk. Dumka, Kiev (2009).
6. Ershov, A.P.: Opyt integralnogo podkhoda k aktualnoi probleme PO. Kibernetika, 11–21 (1984). Nauchnye osnovy dokazatelnogo programmirovaniia. Vestnik AN SSSR 10, 9–19 (1984).
7. Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl, last accessed 2019/11/21.
8. Semantic Web, http://www.w3.org/2001/sw/, last accessed 2019/11/21.
9. Reference Model for Service Oriented Architecture 1.0, http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html, last accessed 2019/11/21.
10. Web Services Resource 1.2 (WS-Resource), http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf, last accessed 2019/11/21.
11. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) // W3C Recommendation. 27 April 2007, http://www.w3.org/TR/SOAP12-part 1, last accessed 2019/11/21.
12. Hebeler, S., Fisher, M., Blace, R., Peter-Lopes, A.: Semantic Web programming. Willey Publishing Inc. (2008).
13. Lavrishcheva, E.M.: Teoriia i praktika fabrik program. Kibernetika i sistemnyi analiz 6, 145–158 (2011); Theory and practice of software factories. Cybernetics and Systems Analysis 47(6), 961–972(2011).
14. Lavrishcheva, E.M.: Classification of software engineering disciplines. Cybernetics and Systems Analysis, 44(6), 791–796 (2008).
15. Lavrishcheva, E.M.: Teoriia obieektno-komponentnogo modelirovaniia programmnykh system. Preprint ISP RAN 29, 1–52 (2016).

16. Ivannikov, V.P., Dyshlevyi, K.V., Mazhelei, S.G.: Raspredelennye obieektno-orientirovannye sredy. Trudy Instituta sistemnogo programmirovaniia RAN, vol. 1, pp. 100–121 (2000).
17. Lavrishcheva, E.M., Ryzhov, A.G.: Primenenie teoriia obshchikh tipov dannykh standarta ISO/IEC 12207 GDT primenitelno k Big Data. In: Conference "Actual problems in science and ways their development", 27 December 2016, pp. 99–110 (2016). http://euroasia-science.ru/, last accessed 2019/11/21.
18. Lavrishcheva, E.M., Karpov, L.E., Tomilin A.N.: Sistemnaia podderzhki resheniia biznes zadach v globalnoi informatsionnoi seti. In: Nauchnyi servis v seti Internet: Trudy XVII Vserossiiskoi nauchnoi konferentsii (19–24 sentiabria 2016, Novorossiisk), pp. 101–127. IPM im. M.V. Keldysha (2015).
19. Lavrishcheva, E.M., Karpov, L.E., Tomilin, A.N.: Semanticheskie resursy dlia razrabotki ontologii nauchnoi i inzhenernoi predmetnykh oblastei. In: Nauchnyi servis v seti Internet: Trudy XVIII Vserossiiskoi nauchnoi konferentsii (19–24 sentiabria 2016 g., g. Novorossi-isk), pp. 126–138. IPM im. M.V. Keldysha (2016), https://doi.org/10.20948/abrau-2016-8.
20. Lavrishcheva, E.M., Karpov, L.E., Tomilin, A.N.: Podkhody k predstavleniiu nauchnykh znanii v Internet nauke. In: XIX Vserossiiskii nauchnoi konferentsii «Nauchnyi servis v seti Internet (Novorossiisk, 18–23 sentiabria 2017), pp. 310–326. IPM im. M.V. Keldysha (2017).
21. Lavrishcheva, E.M.: Komponentnaia teoriia i kollektsiia tekhnologii dlia razrabotki industrialnykh prilozhenii iz gotovykh resursov. In: Trudy 4-oi nauchno-prakticheskoi konferentsii «Aktualnye problemy sistemnoi i programmnoi inzhenerii», APSPI-2015, 20–21 maia 2015, pp. 101–119 (2015).
22. Lavrishcheva, E.M.: Programmnaia inzheneriia. Teoriia Programmirovaniia. MFTI, Moscow (2016).
23. Lavrishcheva, E.M., Petrenko, A.K.: Modelirovanie sistem i ikh semeistv. Trudy ISP RAN 28(6), 180–190 (2016).
24. Kuliamin, V.V., Lavrishcheva, E.M., Mutilin, V.S., Petrenko, A.K.: Verifikatsiia i analiz variabelnykh operatsionnykh system. Trudy ISP RAN 28(3), 189–209 (2017).
25. Lavrishcheva, E.M.: Programmnaia inzheneriia. Paradigmy, Tekhnologii, CASE-sredstva programmirovaniia. 2nd edn. Iurait, Moscow (2016).
26. Lavrishcheva, E.M.: Programmnaia inzheneriia i tekhnologiia programmirovaniia slozhnykh system. Iurait, Moscow (2017).
27. Lavrishcheva, E.M., Kolesnik, A.L., Steniashin, A.Iu.: Obieektno-komponentnoe proektirovanie programmnykh system. Teoreticheskie i prikladnye voprosy. Vesnik KNU, seriia fiz.-mat. Nauk 4, 150–164 (2013); Object-Component Development of Application and Systems. Theory and Practice. Journal of Software Engineering and Applications 7(9) (2014), https://doi.org/10.4236/jsea.2014.79070.
28. Lavrischeva, E.M.: Assembling Paradigms of Programming in Software Engineering. Journal of Software Engineering and Applications 9(6), 296–317 (2016), https://doi.org/10.4236/jsea.2016.96021.
29. Lavrischeva, E.M.: Ontological Approach to the Formal Specification of the Standard Life Cycle. In: Science and Information Conference-2015, July 28–30, London, UK, pp. 965–972, www.conference.thesai.org, last accessed 2019/11/21.
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. In: IEEE/ACM 28-th International Conference on Auto-

mated Software Engineering (ASE 2013). IEEE, pp. 465–474 (2013), https://doi.org/10.1109/ASE.2013.6693104.

32. Lavrishcheva, E.M., Ryzhov, A.V.: Approach to the Modeling of Systems and Sites From Ready Resources. In: CEUR Workshop Proceedings, vol. 2260, pp. 321–345 (2018).
33. Lavrishcheva, E.M., Petrenko, A.K.: Informatics-70. Computerization aspects of programming software and informatic systems technologies.Proc. ISP RAS 1(2), 3–4 (2018).
34. Lavrishcheva, E.M., Avetisian, A.I., Petrenko, A.K.: Informatika. EVM-70. Analiz i aspekty razvitiia. In: ISPRAS-2018, http://0x1.tv/20181122AF, last accessed 2019/11/21.
35. Lavrishcheva, E.M.: The Scientific Basis of Software Engineering. International Journal of Applied and Natural Sciences (IJANS) 7(5), 15–32 (2018).
36. Lavrishcheva, E.M., Pakulin, N.V., Ryzhov, A.G., Zelenov, S.V.: Analiz metodov otsenki nadezhnosti oborudovaniia i sistem. Praktika primeneniia metodov. Trudy ISP RAN 30 (3), 99–120 (2018), https://doi.org/10.15514/ISPRAS-2018-30(3)-8, http://0x1.tv/20180517F, last accessed 2019/11/21.
37. Lavrishcheva, E.M.: Sovremennye sistemy iskusstvennogo intellekta. In: Simpozium «Iskusstvennyi intellekt: razlichnye podkhody k ego voploshcheniiu (kompiuterno-setevye, neirosetevye, kvantovye tekhnologii i drugie)», Moskva, 13 oktiabria 2018. Moscow (2018), https://videonauka.ru/stati/30-metodika-prepodavaniya-tekhnicheskikh-distsiplin/237-informatika-i-evm-70-analiz-i-aspekty-razvitiya, last accessed 2019/11/21.
38. Lavrishcheva, E.M.: Teoriia grafovogo modelirovaniia slozhnykh sistem iz modulei dlia prikladnykh oblastei. Nauchno-prakticheskii zhurnal «Vysshaia shkola» 14, 27–46 (2019).
39. Lavrishcheva, E.M., Mutiltin, V.S., Kozin, V.S., Ryzhov, A.G.: Modelirovanie prikladnykh i informatsionnykh sistem iz gotovykh resursov Internet. Trudy IspRAN 31(1), 7–24 (2019).
40. Lavrischeva, E.M.: The theory graph modeling systems from quality modules of the application areas. In: APSE-2019, pp. 235-247. Moscow (2019).