In-doc controls

DISTRIBUTED SOFTWARE DEVELOPMENT TOOLS FOR DISTRIBUTED SCIENTIFIC APPLICATIONS

Vaidas GIEDRIMAS

Siauliai University, Siauliai, Lithuania, vaigie@mi.su.lt

Leonidas SAKALAUSKAS
Vilnius University, Vilnius, Lithuania, sakal@mii.vu.lt

Anatoly PETRENKO
National Technical University of Ukraine "Kyiv Polytechnic Institute", Kiyv, Ukraine, tolja.petrenko@gmail.com

Abstract: This chapter provides a new methodology and two tools for user-driven wikinomics-oriented scientific applications' development. Service-oriented architecture for such applications is used, where the entire research supporting computing or simulating process is broken down into a set of loosely-coupled stages in form of interoperating replaceable Web services that can be distributed over different Clouds. Any piece of the code and any application component deployed on a system can be reused and transformed into a service. The combination of service-oriented and cloud computing will indeed begin to challenge the way of research supporting computing development, the facilities of which are considered in this chapter.

Keywords: service computing; engineering tools, wikinomics; mathematical programming; software modelling

1. Introduction

One of the factors on which the financial results of the business company depend, is the quality of software which company is using. Scientific software plays even more special role. On its quality depends the reliability of the scientific conclusions and the speed of scientific progress. However, the ratio of successful scientific software projects is close to average: some part of the projects fails, some exceed the budget, some makes inadequate product.

The misunderstandings between scientists as end users and software engineers are even more frequent as usual. Software engineers have a lack of deep knowledge of user's domain (e.g. high energy physics, chemistry, life sciences). In order to avoid possible problems scientists sometimes try to develop "home-made" software. However the probability of failure in such projects are even higher, because of the lack of the knowledge of software engineering domain. For example, scientists in common cases do not know good software engineering practices, processes etc. They even can have a lack of knowledge about good practices or good artefacts of the software, made by its colleagues.

We stand among the believers that this problem can be solved using the Wikinomics. The idea of Wikinomics (or Wiki economics) is introduced by Tapscott and Williams [15]. Wikinomics is the spatial activity, which helps to achieve the result having available resources only. Wiki technologies are laid on very simple procedures: the project leaders collect critical mass of volunteers, who have a willing and possibilities to contribute in small scale. The sum of such small contributions gives huge contribution to the project result. The Wikipedia or Wikitravel portals can be presented as a success stories of mass-collaboration.

In other hand we believe that the mass-collaboration can help to improve only part of the scientific development process. We need a software developing solutions, oriented to services and clouds in order to use all available computational power of the distributed infrastructures.

Service-Oriented Computing (SOC) is an extremely powerful in terms of the help for developer. The key point of modern scientific applications is a very quick transition from hypothesis generation stage to evaluating mathematical experiment, which is important for evidence and optimization of the result and its possible practical use. SOC technologies provide an important platform to make the resource-intensive scientific and engineering applications more significant [1-4]. So any community, regardless of its working area, should be supplied with the technological approach to build their own distributed compute-intensive multidisciplinary applications rapidly.

Service-oriented software developers works either as an application builders (or services clients), service brokers or service providers. Usually the Services Repository is created which contains as Platform Environment Supporting Services, so Application supporting services). The Environment Supporting Services offer the standard operations for service management and hosting (e.g. cloud hosting, event processing and management, mediation

and data services, services composition and workflow, security, connectivity, messaging, storage etc.). They are correlated with generic services, provided by other producers (for example, EGI (http://www.egi.eu/), Flatworld (http://www.flatworldsolutions. com/), FI-WARE (http://catalogue.fi-ware.org/enablers), SAP (http://www.sap.com/pc/tech/enterprise-information- management/), ESRC (http://ukdataservice. ac.uk/) , etc). Two dimensions of service interoperability, namely horizontal (communication protocol and data flow between services) and vertical matching (correspondence between an abstract user task and concrete service capabilities) should be supported in the composition process.

Modern scientific and engineering applications are built as a complex network of services offered by different providers, based on heterogeneous resources of different organizational structures. There are two ways to combine services: either through orchestration or choreography. In orchestration, the involved Web services are under control of a single endpoint central process (another Web service). The choreography, in contrast, does not depend on a central orchestrator. The choreography is based on collaboration and is mainly used to exchange messages in public business processes. As SOC developed, a number of languages for service orchestration and choreography has been introduced: BPEL4WS, BPML, WSFL, XLANG, BPSS, WSCI, and WSCL [5].

Our proposal has the following innovative features:

- Implementation of novel service-oriented design paradigm in Distributed Scientific Applications Development area according to which all levels of research or design are divided into separate loosely coupled stages and procedures for their subsequent transfer to the form of standardized Web services.
- Creation of the repository of research Application Web services which support collective research computing, simulating and globalization of R&D activities.
- Adaption of the Wiki-technologies for creation of the repository of scientific applications' source code, reusing existing software assets at the code level as well as at the web services level.
- Personalization and customization of Distributed Scientific Applications because users can build and adjust their research or design scenario and workflow by selecting the necessary Web services (as computing procedures) to be executed on Cloud resources.

The rest of the paper is organized as follows: Section 2 presents overall idea of the platform for research collaborative computing (PRCC). Section 3 presents web-enabled Engineering Design Platform as one of the possible implementations of PRCC. Section 4 outlines the architecture, and main components of other our system based on wiki-technologies. Section 5 describes the comparison of similar systems. Finally, the conclusions are made and future work discussed.

2. The Platform for Research Collaborative Computing

At the heart of the service-oriented computing there are services that provide autonomous, platform-independent, computational elements that can be described, published, discovered, orchestrated and programmed using standard protocols to build networks of collaborating applications distributed within and across organizational boundaries. In order to describe the service collections and their interconnections, a wide range of Web service composition languages have been developed [5]. The goal of the service composition languages is to glue Web services together and describe coordination logic of service invocations.

In order to support design, development, and execution of distributed applications in Internet environment we have developed the end-user development framework called the Platform for research collaborative computing (PRCC). PRCC is an emerging interdisciplinary field and it embraces physical sciences like chemistry, physics, biology, environmental sciences, hydrometeorology, engineering and even art and humanities. All these fields need powerful tools that meet the needs of a quite wide range of customers in the means of mathematical modelling and collective computing research support, enabling collaboration of distributed group of partners – the providers and consumers of computing resources and data processing solutions. Providing effective ways for the distributed user groups to compose distributed workflows representing the sequence of data processing procedures needed to solve their problems – this is what the Platform for research collaborative computing is about. PRCC has the potential to benefit research in all disciplines at all stages of research. A well-constructed SOC can empower a research environment with a flexible infrastructure and processing environment by provisioning independent, reusable automated simulating processes (as services) and providing a robust foundation for leveraging these services.

PRCC concept is 24/7-available online intelligent multidisciplinary gateway for researchers supporting the following main users' activities: login, new project creation, creation of workflow, provision of input data such as computational tasks description and constrains, specification of additional parameters, workflow execution and collection of data for further analysis.

User authorization is performed at two levels: for the virtual workplace access (login and password) and for grid/cloud resources access (grid certificate).

Application creating. Each customer has a possibility to create some projects, with services stored in the Repository. Each application consists of a set of the files containing information about the computing workflow, the solved tasks, execution results etc.

Solved task description is allowed whether with the problem-oriented languages of the respective services or by the graphic editor.

Constructing of a computational route consists of choosing the computing services needed and connecting them in the execution order required. The workflow editor checks the compatibility of numerical procedures to be connected.

Parameters for different computational tasks are provided by means of the respective web-interface elements or set up by default (except the control parameters, for instance, desirable value for time response, border frequencies for frequency analysis etc.). It can be also required to provide type and parameters of output information (arguments of output characteristics, scale type used for plot construction and others).

Launch for execution initiates a procedure of the application description generation in the internal format and its transferring to the Task execution system. Web and grid service orchestrator is responsible for automatic route execution composed of the remote service invocation. Grid/cloud services invoked by the orchestrator during execution are responsible for preparing input data for a grid/cloud task, its launch, inquiring the execution state, unloading grid/cloud task results and their transferring to the orchestrator.

Execution results consist of a set of files containing information on the results of computing fulfilled (according to the parameters set by a user) including plots and histograms, logs of the errors resulting in a stop of separate route's branches, ancillary data regarding grid/cloud resources used and grid/cloud task executing. Based on the analysis of the received results, a customer could make a decision to repeat computational workflow execution with changed workflow's fragments, input data, and parameters of the computing procedures.

Service-oriented applications governance involves knowledge about services, providers and customers. Development is divided between a service provider and an application builder. This separation enables application builders to focus on research logic of his application while leaving the technical details to service providers. If a large multidisciplinary and multinational Repository of services is created, the customers can tailor the services to their own personal requirements and expectations by incorporating functionalities of available services into large-scale Internet-based distributed application software (figure 1).
Services can be clustered to two main groups: Application Supporting Services (including subgroups: Data Processing Services, Modelling and Simulating Services) and Environment Supporting (Generic) Services (including subgroups: Cloud Hosting for computational, network and software resources provision, Applications/Services Ecosystems and Delivery Framework, Security, Work-flow engine for calculating purposes, Digital Science services).

[Placeholder for fig. 1 Please, do not alter]

Figure 1. General structure of PRCC

As far as authors knows, today there are no well-recognized user-driven applied platforms with support of arbitrary mathematical experiments during scientific and applied research that can offer all of mentioned above. PRCC stands for a new technology and methodology for planning and modelling of mathematical experiments, and it can enhances Europe's future competitiveness by strengthening its scientific and technological base in the area of Experimenting and Data Processing, makes public service infrastructures and simulation processes smarter i.e. more intelligent, more efficient, more adaptive and sustainable.

2.1. Possible content of Services' Repository

Providing the ability to store ever-increasing amounts of data, making them available for sharing and provide scientists and engineers with efficient means of data processing – that is the problem today. In the PRCC this problem is solving by using the service Repository which are described here. From the beginning it includes Application Supporting Services (AS) for the typical scheme of a computational modelling experiment, been already considered.

Web services can contain program codes for implementation of concrete tasks from mathematical modelling and data processing, and also represent results of calculations in Grid/Cloud e-infrastructures. They provide mathematical models equations solving procedures in depending on their type (differential, algebraic- nonlinear, linear) and selected science and engineering analysis. Web services are representing the basic building blocks of system's functionality: input data pre-processing; mathematical model development and its dimension reduction; DC, AC, TR, STA, FOUR and sensitivities analysis; parametrical optimization, tolerances assignment; statistical analysis and yield maximization; data mining, processing results, etc. More detailed description of typical scheme of a computational modelling experiment in many fields of science and technology which has an invariant character is given in [3,10]. The offered list of calculation types covers considerable part of possible needs in computational solving scientifically-applied research tasks in many fields of science and technology.

Services are registered in the network service UDDI (Universal Description, Discovery and Integration) which facilitate the access to them from different clients. Needed functionality is exposed via the web service interface. Each web service is capable to launch computations, to start and cancel jobs, to monitor their status, to retrieve the results etc.

Beside modelling tasks there are other type of computational experiments in which distributed Web service technologies for science data analysis solutions can be used. They include in user scenarios procedures of Curve fitting and Approximation for estimating the relationships among variables, Classification Techniques for categorizing different data into various folders, Clustering Techniques for grouping a set of objects in such a way that objects in the same group (cluster) are more similar to each other than to those in other groups, Pattern Recognition Utilities, Image processing, Filtering and Optimization Techniques.

Above computational Web services for data proceeding are used in different science and technology branches during data collection, data management, data analytics and data visualization, where there are very large datasets: Earth Observation Data from satellites; Data in Meteorology, Oceanography and Hydrology; Experimental Data in Physics of high energy; Observing Data in Astrophysics; Seismograms, earthquake monitoring data, etc.

Services may be offered by different enterprises and communicate over the PRCC, that why they provide a distributed computing infrastructure for both intra- and cross-enterprise

application integration and collaboration. For semantic service discovery in the Repository a set of ontologies was developed which include *resource ontology* (hardware and software grid and cloud resources used for workflow execution), *data ontology* (for annotation of large data files and databases), and *workflow ontology* (for annotating past workflows and enabling their reuse in the future). The ontologies will be separated into two levels: generic ontologies and domain-specific ontologies. Services will be annotated in terms of their functional aspects such as IOPE, internal states (an activity could be executed in a loop and it will keep track of its internal state), data transformation (e.g. unit or format conversion between input and output) and internal processes (which can describe in detail how to interact with a service, e.g. a service which takes partial sets of data on each call, and performs some operation on the full set after last call).

2.2. Web services Management

Implementation of the SOC concept means generating end-user applications based on dynamic composition and orchestration of web services workflows. A workflow describes how tasks are orchestrated, what components performs them, what their relative order is, how they are synchronized, how information flows to support the tasks and how tasks are being tracked. The workflow management may be based on standard web-service orchestration description language WS-BPEL 2.0 (Business Process Execution Language). The initial XML-based description of the abstract workflow containing the task description parameters (prepared by user via the editor) is transformed to the WS-BPEL 2.0 description. Then the orchestration engine invokes web-services passing this task description to them for execution.

The workflow management engine provides seamless and transparent execution of concrete workflows generated at the composition service. This engine leverages existing solutions to allow execution of user defined workflows on the right combination of resources and services available through clusters, grids, clouds or web services. Furthermore, the project plans to work on the development of New scheduling strategies for workflow execution can be implemented that will take into account multi-criteria expressions defined by the user as a set of preferences and requirements. In this way, workflow execution could be directed, for instance, to minimize execution time, to reduce total fee cost, or any combination of both.

The configuration and coordination of services in applications, based on the services, and the composition of services are equally important in the modern service systems [6]. The services interact with each other via messages. Message can be accomplished by using a template "request-response", when at a given time only one of the specific services caused by one user (the connection between "one-to-one" or synchronous model); using a template "publish / subscribe" when on one particular event many services can respond (communications “ one-to-many " or asynchronous model); using intelligent agents that determine the coordination of services, because each agent has at its disposal some of the

knowledge of the business process and can share this knowledge with other agents. Such a system can combine the quality of SOS, such as interoperability and openness, with MAS properties such as flexibility and autonomy.

3. Prototyping optimal design platform for engineering

The analysis of a current state of Engineering simulation and design software proves the urgent need of these systems re-engineering to enable their operation in distributed computing environments. It requires reorganizing such systems in the form of a set of separate interacting modules or services. In that case, the designer's work is to compose a scenario for service interacting (a computational experiment route).

Based on PRCC facilities, the Institute of Applied System Analysis (IASA) of NTUU "Kiev Polytechnic Institute" (Ukraine) has developed the user case WebALLTED[1] as the web-enabled Engineering Design Platform, intended, in particular, for modelling and optimization of Nonlinear Dynamic Systems, which consist of the components of different physical nature and which are widely spread in different scientific and engineering fields. It is the cross-disciplinary application for distributed computing.

Developed engineering service-oriented simulation platform consists of the following layers (Figure 2). This architecture characterized in that: its web-accessible, its functionality is distributed across the ecosystem of both Web services from the PRCC's Repository and grid/cloud services (from e-infrastructure); it is compatible with adopted standards and protocols; it supports custom user analysis scenario development and execution; it hides the complexity of web-service interaction from user with abstract workflow concept and graphical workflow editor.

[Placeholder for fig. 2 Please, do not alter]

Figure 2. Main elements of SOA in the engineering simulation system

User interface provides the following functionality: authorization, graphical workflow editor, project artefacts browsing (input and output files management, simulation results visualizers etc.), task execution monitoring and others. The server-side part of the architecture has several layers to reflect the abstract workflow concept described above. First access tier is the portal which organizes user environment: holds user data and preferences, controls user access, provides information support, organizes user interface. Its modules are also responsible for: abstract workflow description generation according to user inputs, passing this task description to lower architecture layers for execution, retrieving finished task results and storing all the project artefacts in the database.

The next execution tier is the workflow manager running on the execution server. It is responsible for mapping (with the help of service registry) the abstract workflow

[1] The abbreviation ALLTED means ALL TEchnologies Designer [7,8].

description to the concrete web services orchestration scenario expressed in the orchestrator-specific input language (like WS-BPEL for BPEL engines). It also initiates the execution of the concrete workflow with the external orchestrator, monitors its state and fetches the results.

Concrete workflow operates with functional web services on the services tier representing the basic building blocks of system's functionality: data preparation and adaptation, simulation, optimization, results processing etc. Compute-intensive steps are implemented as grid/cloud services interacting with grid/cloud resources to run computations as grid/cloud tasks. Modification or introduction of the new functionality to the system is done by the user by selection or registration of another Web or grid/cloud services.

The execution phase is initiated by user. User task description is passed to workflow management service on execution server, where this abstract workflow is translated to the concrete one. Workflow manager parses the description and checks for errors, requests metadata from service Repository and performs mapping from activity sequence to web service invocations sequence, described in selected standard orchestration languages. Mapper unit of the workflow manager should arrange web services in correct invocation order according to abstract workflow, organize XML messages and variables initializations and assignments between calls, and provide the ways for run-time control (workflow monitoring, cancelling, intermediate results retrieving etc.). Then this concrete scenario is executed by orchestrator.

User is informed about the progress of the workflow execution by monitoring unit communicating with workflow manager. When execution is finished the user can retrieve the results, browse and analyse them and repeat this sequence if needed.

The architecture hides the complexity of web-service interaction from user with abstract workflow concept and simple graphical workflow editor (Figure 3).

[Placeholder for fig. 3 Please, do not alter]

Figure 3. WebALLTED graphical workflow editor

Web services are representing the basic building blocks of simulation system's functionality and they enable customers to build and adjust scenarios and workflows of their design procedures or mathematical experiments via the Internet by selecting the necessary web-services, including automatic creation of equations of a mathematical model (an object or a process) based on a description of its structure and properties of the used components, operations with large-scale mathematical models, steady state analysis, transient and frequency domain analysis, sensitivity and statistical analysis, parametric optimization and optimal tolerances assignment, solution centring, Yield maximization, etc. [3].

Computational supporting services are based mostly on innovative numeric methods can be composed by an end-user for workflow execution on evaluable grid/cloud nodes [3]. They are oriented, first of all, on Design Automation domain, where simulation, analysis

and design can be done for different electronic circuits, control systems and dynamic systems composed of electronic, hydraulic, pneumatic, mechanical, electrical, electromagnetic, and other physical phenomena elements.

The developed methodology and modelling toolkit support collective design of various micro electro-mechanical systems (MEMS) and different microsystems in the form of chips.

4. Distributed Wiki-based system for stochastic programming and modelling

As is mentioned above, even empowered by huge computing power accessible to via web services and clouds user (scientist) have still not exhausted possibilities, because of the lack of communication. Only communication and legal reuse of existing software assets in addition to available computing power can ensure high speed of scientific activities. In this section is described another distributed scientific software development system, which is developed in parallel and independently from the system described in Section 4. However both of these are sharing similar ideas.

4.1. The Architecture of stochastic programming and modelling system

We are started from the following hypothesis: the duration of development of scientific software can be decreased, the quality of such software can be improved using together the power of the grid/cloud infrastructure, wiki-based technologies and software synthesis methods. The project was executed via three main stages:

1. The development of the portal for the wiki-based mass-collaboration. This portal is used as the user interface in which scientists can specify software development problems, can rewrite/refine the specifications and software artefacts given by its (remote) colleagues, can contribute all the process of software development for particular domain. The set of the statistical simulation and optimization problems was selected as the target domain for pilot project. In the future the created environment can be applied to other domains as well.
2. The development of the interoperability model in order to bridge wiki-based portal and the Lithuanian National Grid Infrastructure (NGI-LT) or other (European) distributed infrastructures. A private cloud based on Ubuntu One is created at Siauliai University within the framework of this pilot project.
3. To refine existing methods for software synthesis using the power of distributed computing infrastructures. This stage is under development yet so it is not covered by this chapter. More details and early results is exposed in [22].

[Placeholder for fig. 4 Please, do not alter]

Figure 4. Main components of the Wiki-based Stochastic Programming and Statistical Modelling System

The system for Stochastic Programming and Statistical Modelling based on Wiki technologies (WikiSPSM) consists of the following parts (Figure 1):

- Web portal with the content management system as the Graphical User Interface.
- Server-side backed for tasks scheduling and execution.
- Software artefacts (programs, subroutines, and models etc.) storage and management system.

The user interface portal consists of four main components:

- Template-based generator of web pages. This component helps user to create web page content using template-based structure. The same component is used for the storage and version control of generated web-pages.
- WYSIWYG text editor. This editor provides more functionality than simple text editor on the web page. It is dedicated to describe mathematical models and numerical algorithms. This component is enriched with the text pre-processing algorithms, which prevents from the hijacking attacks and code injection.
- The component of IDE (Integrated developing environment) is implemented for the software modelling and code writing.
- The Repository of Mathematical Functions. This component helps user to retrieve, rewrite and append the repository of mathematical functions with new artefacts. WikiSPSM system is using NetLib repository LAPACK API, however it can be improved on demand and can use other libraries, e.g. ESSL (Engineering Scientific Subroutine Library) or Parallel ESSL [18].

WikiSPSM is easy extensible and evolvable because of the architectural decision to store all the mathematical models, algorithms, programs and libraries in central database.

Initially it was planned that WikiSPSM will enable scientific users to write their software in C/C++, Java, Fortran 90 and QT programming languages. Because of this the command-line interface is chosen as the architecture of communication between the UI and software generator part. Software generator performs the following functions: compilation, task submission (to distributed infrastructure or to single server), task monitoring, control of the tasks and their results. For the compilation of the programs we have chosen external command-line compilers. The architecture of the system lets to change its configuration and to work with another programming language having related SDK with command-line compiler. The users also are encouraged to use command-line interfaces instead of GUI. Latest version of WikiSPSM do not supports application with GUI interfaces. This is done because of two factors: a) many scientific applications are command-line based and the graphical representation of the data is performed with other tools; b) the support of GUI gives more constraints for scientific application.

In early versions of WikiSPSM the compilation and execution actions were made in server side[2]. Object Server creates an object Task for each submitted data array received from the portal. Task object parses the data and send back to user (via portal). For tasks monitoring, results getting the Token is used. When the process Task is finished, it passes the data to Server object and then the process of compilation of execution begins. All the tasks are queued and scheduled. If number of resources (e.g. number of working nodes) is less than requested, task waits the end of some other processes. After task completion its result is stored in the database.

[Placeholder for fig. 5 Please, do not alter]

Figure 5. The architecture of WikiSPSM with the cloud computing component

4.2. Bridge to Distributed Systems

Soon after first test of WikiSPSM was observed, that client-server architecture do not fits the demands on computational resources. Increased number of users and tasks have negative impact on the performance of overall system. The architecture of the system has been changed in order to solve this issue.

In current architecture the software generation component is changed dramatically. The adoption of this legacy component is made in two stages:

- Transformation between different operating systems. First server-side application was hardly coupled with Windows, because of chosen command-line compilers and Qt library. This part was redesigned completely. New Linux-based implementation is made, so now WikiSPSM can be considered as multiplatform tool.
- Transformation between the paradigms. In order to ensure better throughput of computing application server was redesigned to schedule tasks in distributed infrastructures. Ubuntu One and Open Stack private clouds were chosen for the pilot project (Figure 5). Distributed file system NFS is used for the communication of working nodes.

Tests of redesigned component show very good results. For example for 150 tasks Monte-Carlo problem using new (bridged to distributed systems) execution component was solved in 2 times faster than initial server-based application component. The "toy example" (calculation of the factorial of big numbers) – was solved 8 times faster.
More comprehensive information about WikiSPSM could be found in [19].

[2] server side" here means general backend including cloud also.

5. Related work

As far as authors of the chapter knows the conception of Engineering SOC with Design procedures as Web services has almost no complete competitors worldwide [3]. However partial comparison to other systems is possible.

WebALLTED is based on the original numerical algorithms for all the stages of design [3,7,8]: starting from steady state, frequency and transient analyses till parametrical optimization of a designed device output characteristics, optimal component tolerances assignment, cantering of solution, and Yield maximization. The proposed approach to application design is completely different from present attempts to use the whole indivisible applied software in the grid / cloud infrastructure as it is done in Cloud SME, TINACloud , PartSim , RT-LAB, FineSimPro and CloudSME .

In comparison with SPICE-like programs WebALLTED offers:

- Faster simulation speed and improved numerical convergence;
- Sensitivity analysis for frequency and transient analyses;
- Comprehensive optimization procedure and optimal tolerances assignment;
- Alternative approach to the secondary response parameters determination (delays, rise and fall times, etc.);
- Powerful user-defined modelling capability;
- Original way of generating a system-level model of MEMS from FEM component equations (being received, for example, by means of ANSYS) when these equations with boundary conditions are transformed into the equivalent equations of a schematic model, which consists of L, C and G components, and then are simplified by means of Y-Δ transformation [7];
- Dynamically configurable software architecture due to composited services and executing in grid nodes.

For evaluation the possibilities of WikiSPSM it has been compared to other commercial (Mathematica) and open-source (Scilab) products. All compared products supports rich set of mathematical functions, however Mathematica's list of functions is most distinguishing for the problems of mathematical programming. WikiSPSM uses NetLib repository LAPACK [33] for C++ and FORTRAN, so they provides more functionality as Scilab. In contrast to Mathematica and Scilab, WikiSPSM cannot reuse its functions directly, because it is web-based and all the programs are executed on the server side, not locally. However WikiSPSM shows best result by the possibility to extend system repository. Other systems have different – single user oriented architecture. Moreover they have only a little possibility to change system functions or extend the core of the system by user subroutines.

6. CONCLUSIONS

The following conclusion can be made:

- The analysis of a current state of scientific software development tools proves the urgent need of existing tools re-engineering to enable their operation in distributed computing environments.
- The original concept of the service-oriented distributed scientific applications development (with computing procedures as Web services) has the following innovative features:
    - Division of the entire computational process into separate loosely coupled stages and procedures for their subsequent transfer to the form of unified software services;
    - Creation of a Repository of computational Web services which contains components developed by different producers that support collective research applications development and globalization of R&D activities;
    - Separation services into Environment Supporting (Generic) Services and Application Supporting services;
    - Unique Web services to enable automatic formation of mathematical models for the solution tasks in the form of equation descriptions or equivalent substituting schemes;
    - Personalized and customized user-centric application design enabling users to build and adjust their design scenario and workflow by selecting the necessary Web services to be executed on grid/cloud resources.
    - Re-composition of multidisciplinary applications can at runtime because Web services can be further discovered after the application has been deployed;
    - Service metadata creation to allow meaningful definition of information in cloud environments for many service providers which may reside within the same infrastructure by agreement on linked ontology.
    - The possibility to collaborate using Wiki-technologies and reuse software at code level as well as at service level.
- The prototype of the service-oriented Engineering Design Platform was developed on the base of the proposed architecture for Electronic Design Automation domain. Beside EDA the simulation, analysis and design can be done using WebALLTED for different control systems and dynamic systems
- The prototype of collaboration-oriented stochastic programming and modelling system WikiSPMS was developed on the base of Wiki technologies and open source software.

We believe that the results of the projects will have direct positive impact in the scientific software development, because of the bridging two technologies, each of them promises good performance. The power of the wiki-technologies, software services and clouds will ensure the ability of the interactive collaboration on software developing using the terms of particular domain.

7. References

[1] Yinong Chen, Wei-Tek Tsai. Distributed Service-Oriented Software Development. Kendall Hunt Publishing; 2008. 467 p.
[2] M P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann. Service-Oriented Computing: A Research Roadmap. International Journal of Cooperative Information Systems. 2008;17(2):223–255. .
[3] A.I.Petrenko. Service-oriented computing (SOC) in a cloud computing environment. Computer Science and Applications. 2014;1(6):349-358..
[4] J. Kress, B. Maier, H. Normann, D. Schmeidel, G. Schmutz, B. Trops, C. Utschig-Utschig, T. Winterberg. Industrial SOA [Internet]. . Available from: http://www.oracle.com/ technetwork/articles/soa/ind-soa-toc-1934143.html
[5] OASIS, "OASIS Web Services Business Process Execution Language [Internet]. 2008 . Available from: http://www. oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
[6] A.A. Petrenko. Comparing the types of service systems architectures" (in Ukrainian). System Research & Information Technologies. 2015;4:48-62..
[7] M. Zgurovsky, A. Petrenko, V. Ladogubets, O. Finogenov, B.Bulakh. WebALLTED: Interdisciplinary Simulation in Grid and Cloud. Computer Science (Cracow). 2013;**14**(2):295-306 .
[8] A.Petrenko, V. Ladogubets, V.Tchkalov, Z. Pudlowski. ALLTED - a Computer-Aided System for Electronic Circuit Design. In: UICEE (UNESCO); 1997; Melbourne. 1997. p. - 204.
[9] A.I. Petrenko. Macromodels of Micro-Electro-Mechanical Systems. In: Microelectro-mechanical Systems and Devices, Nazmul Islam , editor. Microelectro-mechanical Systems and Devices. InTech; 2012. p. 155-190.
[10] A.I. Petrenko. Collaborative Complex Computing Environment (Com-Com). J Comput Sci Syst Biol . **8**:278-284. DOI: 10.4172/jcsb.1000201
[11] V. Giedrimas, L. Sakalauskas, K. Žilinskas. Towards the environment for mass-collaboration for software synthesis, EGI User Forum [Internet]. 2011 . Available from: https://indico.egi.eu/indico/event/207/session/15/contribution/117/material/slides/0.pdf
[12] V. Giedrimas, A. Varoneckas, A. Juozapavicius. The grid and cloud computing facilities in Lithuania. Scalable Computing: Practice and Experience. 2011;**12**(4):417–421.
[13] S. Steinhaus. Comparison of mathematical programs for data analysis. Munich:2008.
[14] M. Baudin . Introduction to Scilab. The Scilab Consortium; 2010.
[15] C. Bunks, J.-P. Chancelier, F. Delebecque. C. Gomez, M. Goursat, R. Nikoukhah and S. Steer. . Engineering and scientific computing with Scilab. Boston: Birkhauser; 1999.
[16] ESSL and Parallel ESSL library [Internet]. . Available from: http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp
[17] D. Tapscott, A. D. Williams. Wikinomics – How mass collaboration changes everything. Atlantic Books; 2011.
[18] Mathematica Documentation Center [Internet]. Available from: http://reference.wolfram.com/mathematica/guide/Mathematica.html
[19] V.A. Barker et al.. LAPACK User's Guide: Software, Envinroments and Tools. . Society for Industrial and Applied Mathematics; 2001.

[20] L. Sakalauskas. Application of the Monte-Carlo method to nonlinear stochastic optimization with linear constraints. Informatica. 2004;**15**(2):271-282.
[21] V. Giedrimas , L. Sakalauskas, K. Žilinskas, N. Barauskas, M. Neimantas, R. Valčiukas. Wiki-based stochastic programming and statistical modelling system for the cloud. International Journal of Advanced Computer Science & Applications. 2016;**7**(3):218-223.
[22] V. Giedrimas. Distributed Systems for Software Engineering: Non-traditional Approach. In Proc. of 7th International Conference on Application of Information and Communication Technologies (AICT 2013). 2013, 31-34.

[Double click to insert bibliographysource here]