Chapter 5

Distributed Software Development Tools for Distributed Scientific Applications

Vaidas Giedrimas, Leonidas Sakalauskas and Anatoly Petrenko

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.68334

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 the 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 modeling

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 depend 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 exceeds the budget, and 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, and life sciences). In order to avoid possible problems,

scientists sometimes try to develop "home-made" software. However, the probability of failure in such projects is 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, and so on. 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 story of mass collaboration [17].

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 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 work either as application builders (or services clients), service brokers, or service providers. Usually, the service repository is created which contains platform environment supporting services and 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, service composition and workflow, security, connectivity, messaging, storage, and so on). They are correlated with generic services, provided by other producers (e.g., 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/), and so on). 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. The services can be composed using orchestration or using choreography. If the orchestration is used, all corresponding Web services are controlled by one central web service. On the other hand, if the choreography is used and central orchestrator is absent, the

services are independent in some extent. 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 have 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 application 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 our other systems 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

The service-oriented computing is based on the software services, which are platform-independent, autonomous, and computational elements. The services can be specified, published, discovered, and composed with other services using standard protocols. Such composition of services can be threat as wide-distributed software system. Many languages for software service composition are developed [5]. The goal of such languages is to provide formal way for specifying the connections and the coordination logic between services.

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 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 are demanding for potent tools for mathematical modeling and collaborative computing research support. These tools should implement the idea of virtual organization including the possibility to combine distributed workflows, sequences of data processing functions, and so on. The platform for

research collaborative computing is the answer for this demand. 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 task 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, and so on.

Solved task description is allowed whether with the problem-oriented languages of the respective services or with 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, and so on). 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 executioninitiates a procedure of the application description generation in the internal format and its transferring to the task execution system. Web and grid service orchestrator are 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.

It is a need to know more details on services, its providers, and the customers, in order to manage service-oriented applications. There are two roles in development process: the role of service provider and the role of application builder. This separation of concerns empowers

application architects to concentrate more on the business logic (in this case research). The technical details are left to service providers. Comprehensive repository of various services would ensure the possibility to use the services for the personal/institutional requirements of the scientific users via incorporation of existing services into widely distributed system (**Figure 1**).

Services can be clustered to two main groups: application supporting services (including subgroups: data processing services, modeling, 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).

As far as authors know, there are no similar user-oriented platforms supporting experiments in mathematics and applied sciences. PRCC unveils new methodology for mathematical experiments planning and modeling. It can improve future competitiveness of the science by strengthening its scientific and technological base in the area of experimenting and data processing, which 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 providing scientists and engineers with efficient means of data processing are the problems today. In the PRCC, this problem is solving by using the service repository

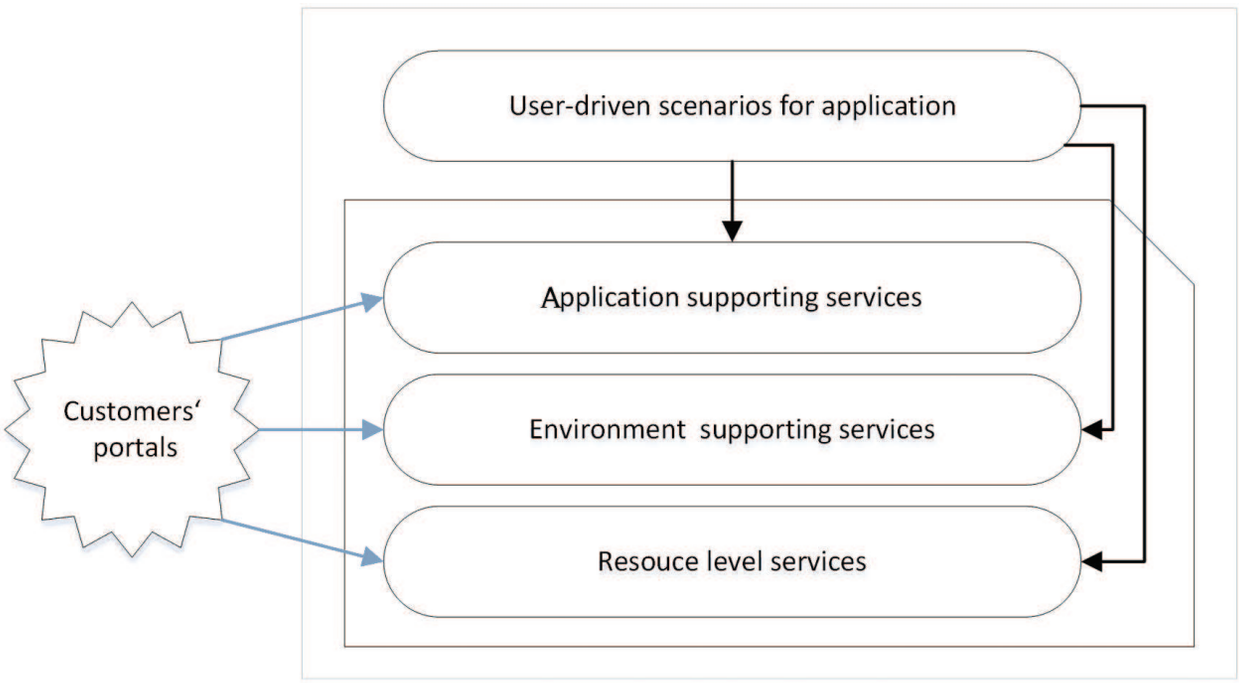

Figure 1. General structure of PRCC.

which is described here. From the beginning, it includes application supporting services (AS) for the typical scheme of a computational modeling experiment, been already considered.

Web services can contain program codes for implementation of concrete tasks from mathematical modeling and data processing and also represent results of calculations in grid/cloud e-infrastructures. They provide mathematical model equations solving procedures in depending on their type (differential, algebraic-nonlinear, and linear) and selected science and engineering analysis. Software services are main building blocks for the following functionality: data preprocessing and results postprocessing, mathematical modeling, DC, AC, TR, STA, FOUR and sensitivities analysis, optimization, statistical analysis and yield maximization, tolerance assignment, data mining, and so on. More detailed description of typical scheme of a computational modeling 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, and so on.

Besides modeling tasks, there are other types of computational experiments in which distributed Web service technologies for science data analysis solutions can be used. They include in user scenario 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, and 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 data sets: 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, and so on.

Services may be offered by different enterprises and communicate over the PRCC, that is 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. Management of Web services

Service-oriented paradigm implies automated composition and orchestration of software services using workflows. Each workflow defines how tasks should be orchestrated and what components in what execution sequence should be. The workflow also includes the details of the synchronization and data flows. 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 multicriteria 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); and 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 state-of-art scientific platforms shows that there is a need of distributed computing-oriented platform. This obliges to redesign similar environments in the terms of separate interacting software services. So the designers should specify a workflow of the interaction of services.

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 modeling 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**). The most important features of this architecture are the following: Web accessibility, the distribution of the functionality across the software services in e-infrastructure, the compatibility with existing protocols and standards, the support of user-made scenarios in

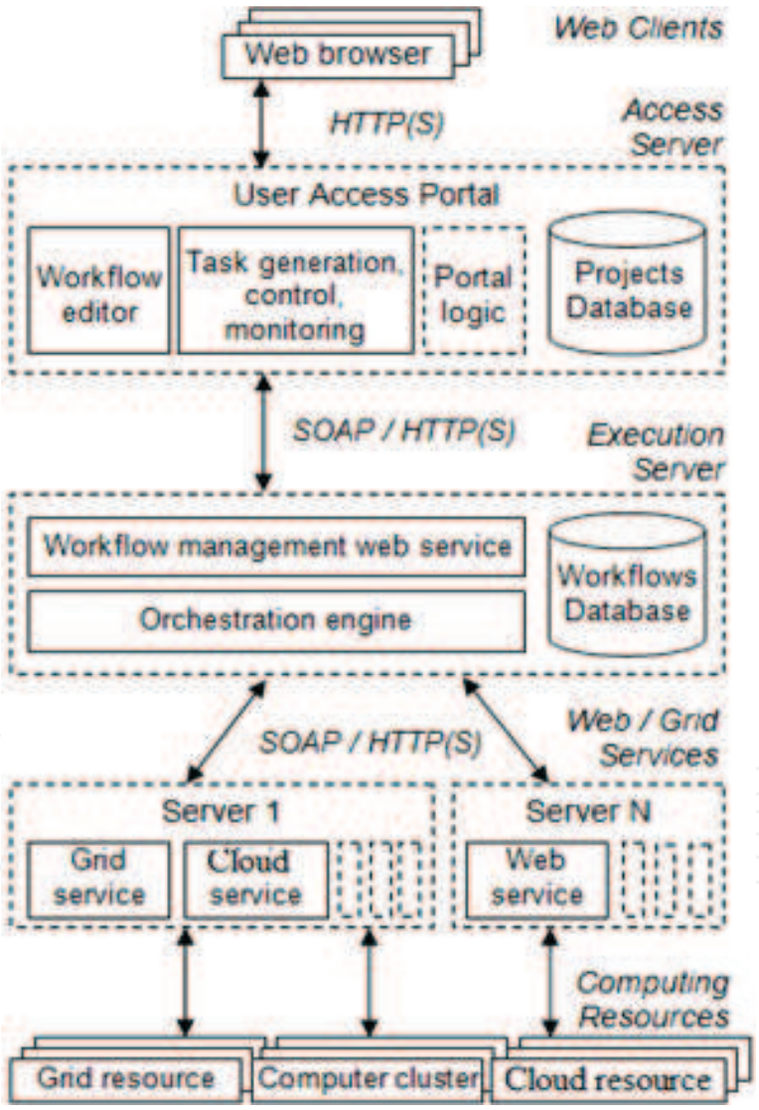

Figure 2. Main elements of SOA in the engineering simulation system.

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

http://dx.doi.org/10.5772/intechopen.68334

development-time and in run-time, and the encapsulation of the software services interaction complexity.

The following functions are accessible via user interface: authentication, workflow editor, artefacts repository management environment, task monitoring, and more. The server side of the system is designed as multitier one in order to implement the workflow concept described early. First-access tier is the portal supporting user environment. The purpose of its modules is the following: the user-input-based generation of abstract workflow specification; the transition of task specification to lower tiers, where the task will be executed; and the postprocessing of results including saving the artefacts in DB.

The workflow manager works as second-execution tier. It is deployed in the execution server. The purpose of this tier is the mapping of the abstract workflow specification to particular software services. The orchestration is done using the specific language similar to WS-BPEL for BPEL instruments. The workflow manager starts executing particular workflow with the external orchestrator as well as observes the state of workflow execution and procures its results.

Particular workflow is working with functional software services and performs the following actions: data preprocessing and postprocessing, simulation, optimization, and so on. If high demand for resources is forecasted, only one node could be loaded to heavy. So the computation is planned on separate nodes and hosting grid/cloud services. These services give possibility to use widespread infrastructure (such as grid or cloud). It is possible to modify and to introduce of new functions to the system. This is done by the user by selection or registration of another Web or grid/cloud services.

The user is able to start the task in an execution tier. Task specification is transient to the service of workflow management. This abstract workflow is transformed to the particular implementation on execution server. Then, the workflow manager analyses the specification, corrects its possible errors (in some extent), demands the data about the services from the repository, and performs binding of activity sequence and software services calls. For the arrangement of software services in correct invocation order, the Mapper unit is used in the workflow. It initializes XML messages, variables, etc., and provides the means for the control during a run-time including the observing of workflow execution, its cancelling, early results monitoring, and so on. Finally, the orchestrator executes this particular "script."

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 analyze 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**).

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)

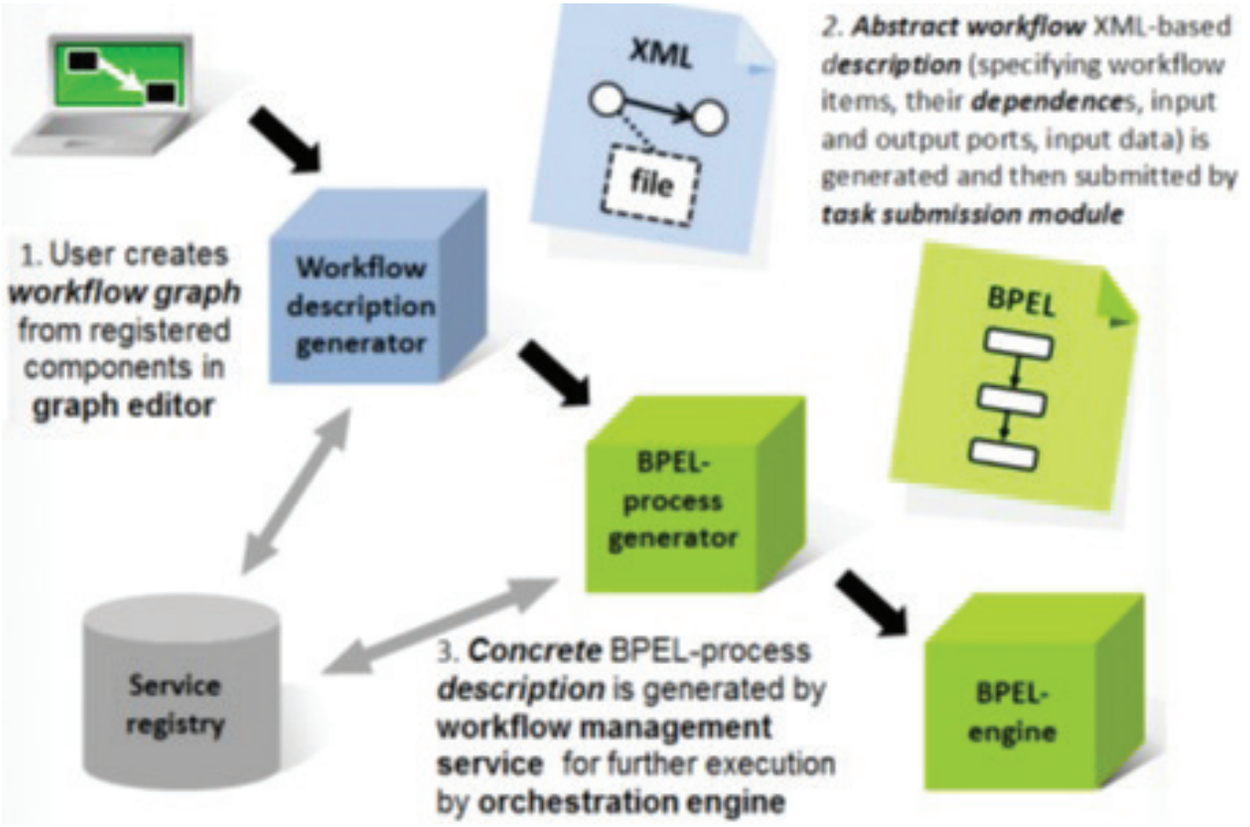

Figure 3. WebALLTED graphical workflow editor.

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 tolerance assignment, solution centering, yield maximization, and so on [3].

Computational supporting services are based mostly on innovative numeric methods and 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 modeling 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 modeling

As is mentioned above, even empowered by huge computing power accessible to via Web services and clouds, users 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 modeling 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 with the power of the grid/cloud infrastructure, Wiki-based technologies, and software synthesis methods. The project was executed via three main stages:

- 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, and 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.
- 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.
- 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 are exposed in [22] (**Figure 4**).

The system for stochastic programming and statistical modeling 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, 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 preprocessing algorithms, which prevents from the hijacking attacks and code injection.

- The component of IDE (integrated developing environment) is implemented for the software modeling 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 [16].

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, and control of the tasks and their results. For the compilation of the programs, we have chosen external command-

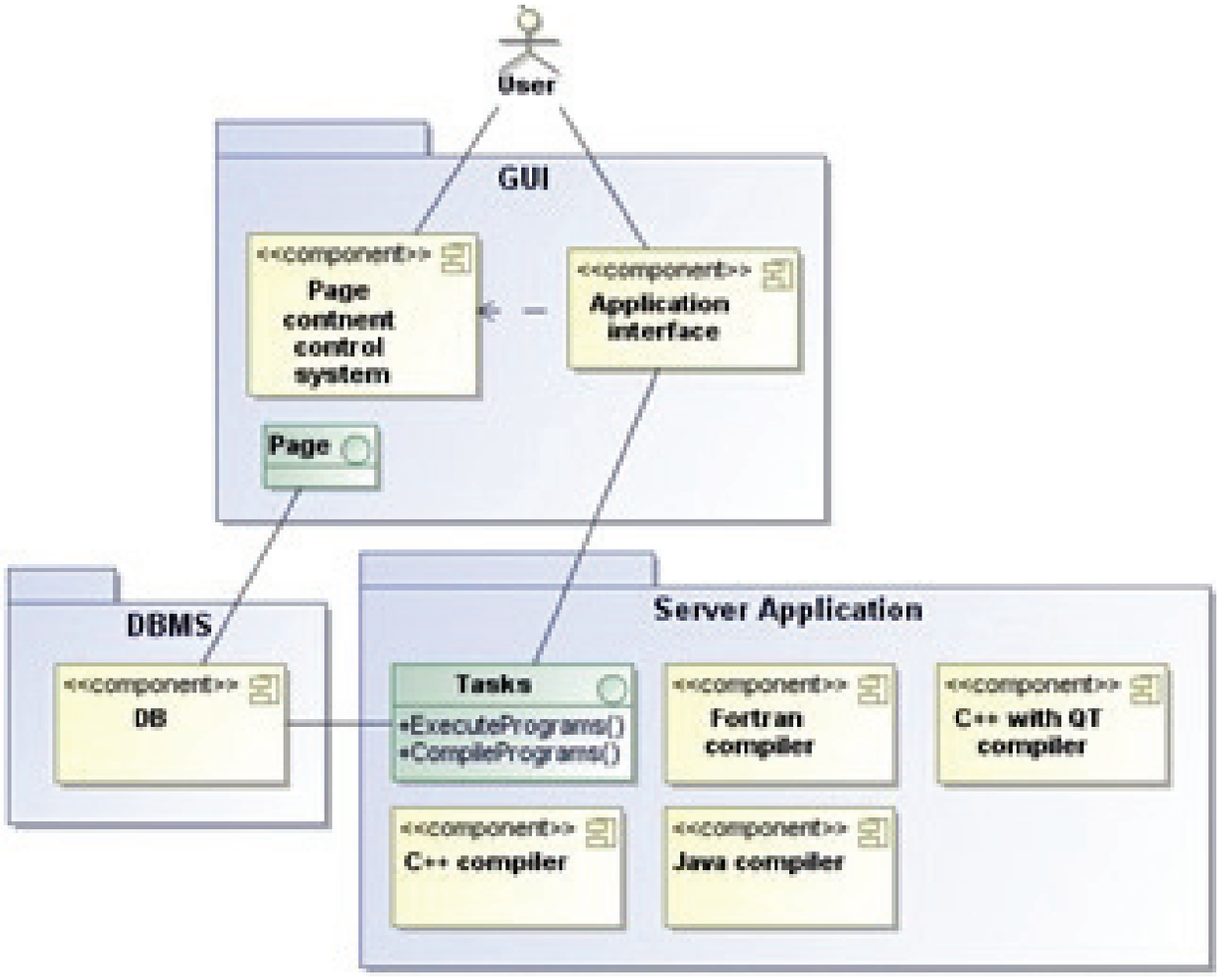

Figure 4. Main components of the Wiki-based stochastic programming and statistical modeling system.

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 does not support 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 sends back to user (via portal). For tasks monitoring, results getting the token are used. After the finishing of task, it transfers the data to server object. After that it is time for the compilation and execution. Each task is queued and scheduled. If it is not sufficient amount of resources (e.g., working nodes), task is laid out to the waiting queue. When the task is finished, its results are stored in DB (**Figure 4**).

4.2. Bridge to distributed systems

Soon after first test of WikiSPSM was observed, that client-server architecture does not fit 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 component for software generation was changed. This change was performed via two stages:

- Transformation between different OSs.[3] The server side of previous version was hardly coupled with OS (in particular, Windows). It was based on Qt API and command-line compilers. This fragment was reshaped completely. New implementation is Linux oriented, 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 two times faster than initial server-based application component. The "toy example" (calculation of the factorial of big numbers) was solved eight times faster.

More comprehensive information about WikiSPSM could be found in Ref. [11, 12, 20, 21].

[2]Server side" here means general backend including cloud also.
[3]OS—the operating system.

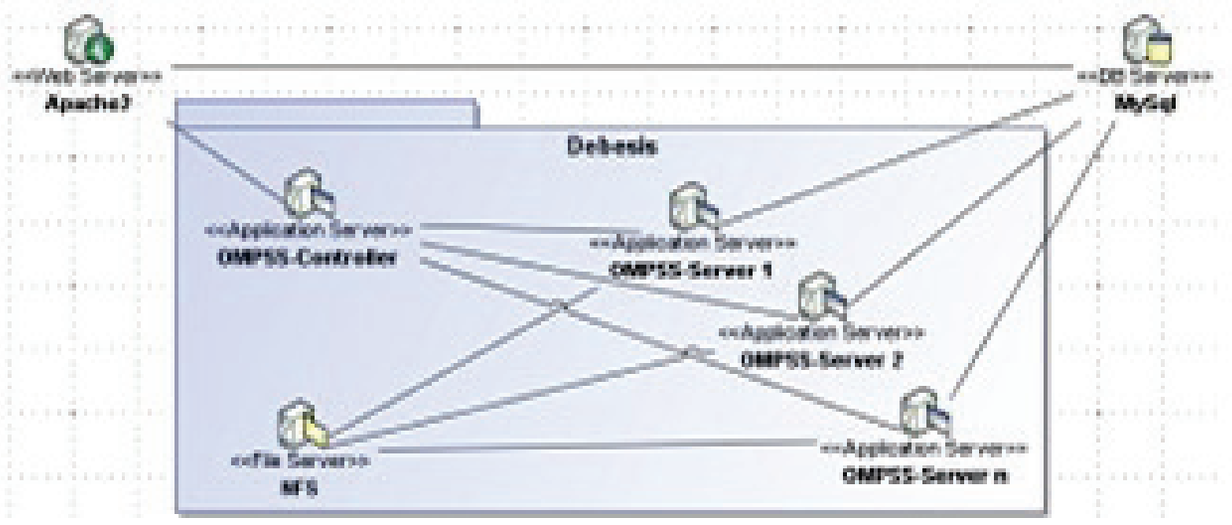

Figure 5. The architecture of WikiSPSM with the cloud computing component.

5. Related work

As far as authors of the chapter know 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.

The original numerical algorithms are in the background of WebALLTED [3, 7, 8, 9], e.g., algorithms for analysis of steady or transient state, frequency, algorithms for parametrical optimization, yield maximization, and so on. 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.

WebALLTED was compared to SPICE. The following positive features of WebALLTED were observed:

- Improvements on simulation rapidity and numerical convergence;
- Inclusive procedure of optimization and tolerance threshold setting;
- Sensitivity of analysis tools;
- Different approach of the determination of secondary response parameters (e.g., delays);
- Richer possibilities to perform user-defined modeling;
- Novel way of generate a system-level model of MEMS from FEM component equations (e.g., being received by means of ANSYS) [7];
- Dynamicity of the software architecture configuration in the terms of composed services and working nodes.

For evaluation the possibilities of WikiSPSM, it has been compared to other commercial (Mathematica) and open-source (Scilab) products. All compared products support rich set

of mathematical functions; however, Mathematica's list of functions [13, 18] is most distinguishing for the problems of mathematical programming. WikiSPSM uses NetLib repository LAPACK [19] for C++ and FORTRAN, so they provide more functionality as Scilab [14]. 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 application 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 modeling 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.

Author details

Vaidas Giedrimas[1]*, Leonidas Sakalauskas[1,2] and Anatoly Petrenko[3]

*Address all correspondence to: vaigie@mi.su.lt

1 Siauliai University, Siauliai, Lithuania

2 Vilnius University, Vilnius, Lithuania

3 National Technical University of Ukraine "Kyiv Polytechnic Institute", Kiyv, Ukraine

References

[1] Chen Y, Tsai W-T. Distributed Service-Oriented Software Development. Kendall Hunt Publishing; Iowa, USA. 2008. p. 467

[2] Papazoglou MP, Traverso P, Dustdar S, Leymann F. Service-oriented computing: A research roadmap. International Journal of Cooperative Information Systems. 2008;**17**(2):223-255

[3] Petrenko AI. Service-oriented computing (SOC) in a cloud computing environment. Computer Science and Applications. 2014;**1**(6):349-358

[4] Kress J, Maier B, Normann H, Schmeidel D, Schmutz G, Trops B, Utschig-Utschig C, Winterberg T. Industrial SOA [Internet]. 2013. Available from: http://www.oracle.com/technetwork/articles/soa/ind-soa-preface-1934606.html [Accesed: 2017-01-30]

[5] OASIS. OASIS Web Services Business Process Execution Language [Internet]. 2008. Available from: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel [Accessed 2017-06-01]

[6] Petrenko AA. Comparing the types of service systems architectures [in Ukrainian]. System Research & Information Technologies. 2015;**4**:48-62

[7] Zgurovsky M, Petrenko A, Ladogubets V, Finogenov O, Bulakh B. WebALLTED: Interdisciplinary simulation in grid and cloud. Computer Science (Cracow). 2013; **14**(2):295-306

[8] Petrenko A, Ladogubets V, Tchkalov V, Pudlowski Z. ALLTED—A Computer-Aided System for Electronic Circuit Design. Melbourne: UICEE (UNESCO); 1997. p. 204

[9] Petrenko AI. Macromodels of micro-electro-mechanical systems. In: Islam N, editor. Microelectro-Mechanical Systems and Devices. InTech Rijeka, Croatia; 2012. pp. 155-190

[10] Petrenko AI. Collaborative complex computing environment (Com-Com). Journal of Computer Science and Systems Biology. 2015;**8**:278-284. DOI: 10.4172/jcsb.1000201

[11] Giedrimas V, Sakalauskas L, Žilinskas K. 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 [Accessed 2016-09-15]

[12] Giedrimas V, Varoneckas A, Juozapavicius A. The grid and cloud computing facilities in Lithuania. Scalable Computing: Practice and Experience. 2011;**12**(4):417-421

[13] Steinhaus S. Comparison of mathematical programs for data analysis. Munich; 2008. http://www.newsciencecore.com/attach/201504/09/173347uyziem4evkr0i405.pdf last accesed : 2017-06-11

[14] Baudin M. Introduction to Scilab. The Scilab Consortium; 2010

[15] Bunks C, Chancelier J-P, Delebecque F, Gomez C, Goursat M, Nikoukhah R, Steer S. Engineering and Scientific Computing with Scilab. Boston: Birkhauser; 1999

[16] ESSL and Parallel ESSL Library [Internet]. IBM, 2016. Available from: https://www-03.ibm.com/systems/power/software/essl/ [Accessed 2017-06-01]

[17] Tapscott D, Williams AD. Wikinomics: How Mass Collaboration Changes Everything. Atlantic Books, London; 2011

[18] Wolfram Research. Wolfram gridMathematica: Multiplying the power of Mathematica over the grid. [Internet]. 2006. Available from: http://www.wolfram.com/gridmathematica/ [Accessed 2017-06-11]

[19] Barker VA, et al. LAPACK User's Guide: Software, Environments and Tools. Society for Industrial and Applied Mathematics; Philadelphia, USA. 2001

[20] Sakalauskas L. Application of the Monte-Carlo method to nonlinear stochastic optimization with linear constraints. Informatica. 2004;**15**(2):271-282

[21] Giedrimas V, Sakalauskas L, Žilinskas K, Barauskas N, Neimantas M, Valčiukas R. Wiki-based stochastic programming and statistical modelling system for the cloud. International Journal of Advanced Computer Science & Applications. 2016;**7**(3): 218-223

[22] Giedrimas V. Distributed systems for software engineering: Non-traditional approach. In: Proceedings of the 7th International Conference on Application of Information and Communication Technologies (AICT 2013). Baku (Azerbaijan), 23-25 October. Published by Institute of Electrical and Electronics Engineers (IEEE); 2013. pp. 31-34

IntechOpen