Wiki-Based Stochastic Programming and Statistical Modeling System for the Cloud

Vaidas Giedrimas
Department of Informatics
Siauliai University
Siauliai, Lithuania

Kestutis Žilinskas
Department of Informatics
Siauliai University
Siauliai, Lithuania

Leonidas Sakalauskas
Department of Informatics
Siauliai University
Siauliai, Lithuania

Nerijus Barauskas
Imagine Communications Group
Dublin, Ireland

Marius Neimantas
Department of Informatics
Siauliai University
Siauliai, Lithuania

Remigijus Valčiukas
Department of Informatics
Siauliai University
Siauliai, Lithuania

***Abstract*—Scientific software is a special type of software because its quality has a huge impact on the quality of scientific conclusions and scientific progress. However, it is hard to ensure required quality of the software because of the misunderstandings between the scientists and the software engineers. In this paper, we present a system for improving the quality of scientific software using elements of wikinomics and cloud computing and its implementation details. The system enables scientists to collaborate and make direct evolution of the models, algorithms, and programs. WikiSPSM expands the limits of mathematical software.**

Keywords—Wikinomics; open source; mathematical programming; software modeling; online computing

I. INTRODUCTION

Scientific software is a special type of software because its quality depends not only on the financial results but also that of the quality of appropriate scientific conclusions and the speed of scientific progress. However, the success ratio for the projects of scientific software development is close to average. This means that part of the projects fail, over budget or give inadequate product.

The misunderstandings between final users (scientists) and contractors (software engineers) are even more frequent as usual. This is because of the fact that software engineers sometimes are unable to get deep knowledge of user's domain (e.g. high energy physics or life sciences). In order to avoid possible problems, scientists sometimes try to develop software indecently. However, such projects can also fail due to the lack of knowledge of software engineering domain. For example, scientists may not know (or just do not care about) good software engineering practices, common processes, etc. They may even lack knowledge about good practices or good ,artifacts of the software, made by their colleagues.

We believe that this problem must and can be solved using the idea of *Wikinomics*, which is introduced by Tapscott and Williams [7]. Wikinomics (or Wiki economics) is a spatial activity, which helps to achieve results using the available resources only. The idea of Wiki-based systems is very simple: the project leaders collect critical mass of volunteer who are willing and who can to contribute small parts. Sum of such small parts makes a huge contribution to project goals and makes this form of mass-collaboration very attractive. Such systems as Wikipedia or Wikitravel are good advocates of the advantages of the Wiki technologies.

On the other hand mass-collaboration in the software modelling or software coding states is not enough. In order to use all (or at least part of) computational power of the distributed infrastructures, we need software developing solutions, oriented to clouds and grids.

Software synthesis for grid, cloud, and other distributed systems is one of the three main distributed computing-related research areas of Siauliai University [1, 2]. On the umbrella of this research area, a new project at Siauliai University was started from October 1, 2010. The main goal of this project was to develop the environment for scientific software synthesis using grid, cloud, web, and wiki-oriented technologies.

The goal of this paper is to present results of the empirical research related to this project, namely - the system for the Stochastic Programming and Statistical Modelling System empowering the members of scientific community with the abilities to make, edit, and reuse models, algorithms and the software.

The rest of the paper is organized as follows: Section 2 outlines the overall architecture of our system including its main components. Section 3 presents the result of testing.

Section 4 describes the comparison of similar systems. Finally, the conclusions are made and future work is discussed.

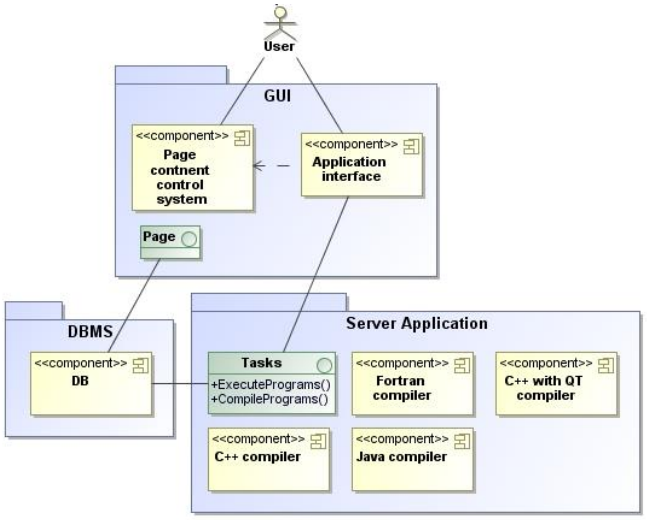

Fig. 1. Main components of the Wiki-based Stochastic Programming and Statistical Modelling System

II. The Architecture of Stochastic Programming and Modelling System

We have started from the hypothesis that using together wiki-based technologies, software synthesis methods, and the power of the grid/cloud infrastructure, scientific software can be developed more rapidly and the quality of the software will increase. The project consists of three main stages:

1) The development of the portal for the wiki-based mass-collaboration. This portal will be used as the UI enabling scientists to specify the problems for software development, to rewrite/refine the specifications and software artefacts given by other researchers, to contribute all software developing for particular domain process. As the target domain for software development we have chosen the set of the statistical simulation and optimization problems. In the future the created environment can be applied to other domains.

2) The development of the model of the interoperability bridge between wiki-based portal and the Lithuanian National Grid Infrastructure or other distributed infrastructures. For this purpose currently the private cloud is created at Siauliai University, based on Ubuntu One.

3) To refine existing methods for software synthesis using the power of distributed computing infrastructures.

This paper mainly covers the results of the first two stages.

The system for Stochastic Programming and Statistical Modelling based on Wiki technologies (WikiSPSM) is built using the Model-View-Control architecture. It consists of the following parts (Figure 1):

- Web portal with the content management system and Wiki-capabilities as the Graphical User Interface (see Subsection B for more details).
- Server application for jobs processing (Subsection C).
- Database and its management system for the storage and processing of existing software artifacts including programs, subroutines, and models (Subsection A).

The importance of each part and its role in overall functionality of the system is detailed in the remaining subsections.

A. Database

The database is based on MySQL. It consists of 34 tables, which can be divided to 5 groups: user data, logging data, job data, website data, and program data.

The tables of the user data consist of information about user access rights, groups, etc. Tables for logging data consist of the portal usage history data, portal user actions, and so on. Job data tables carry information about the submitted jobs and their states and location details.

Website data is the basis for CMS and user interface of the Wiki part of the portal. Program data (PD) is probably the most important (and most distinctive) part of all the systems. PD tables are for storage and processing of the programs, written by users and APIs. In other words, PD is the basis for the Wiki part of the portal.

B. Wiki-based Portal

The user interface portal is the central entry-point for the users in our system. It consists of four main components:

- *Template-based generator of web pages*. This component empowers a user to make web page content using its structure from the template. The same component is used for the storage of generated web-pages including version control.
- *WYSIWYG text editor*. The functionality of this component is beyond simple text editor on the web page. The component is dedicated to describing mathematical models and numerical algorithms. For safety reasons, it is enriched with the text pre-processing algorithms, which prevents code injection and hijacking attacks.
- *Integrated developing environment* component is implemented for the modeling and the codding of the software. It is built according to theoretical background published in [10].
- *Mathematical Functions Repository*. This component enables a user to retrieve, rewrite, and add repository of mathematical functions with new artifacts, for example, new APIs. Almost all WikiSPSM is built from scratch. However, for efforts optimization it was decided to include “standart” library of mathematical functions in it. Our system is based on NetLib repository LAPACK API. However, it can be changed on demand to work with other libraries, for example, ESSL (Engineering Scientific Subroutine Library) or Parallel ESSL [6].

The architectural decision to store all the mathematical models, algorithms, programs, and APIs in central database makes WikiSPSM easy extensible and evolvable.

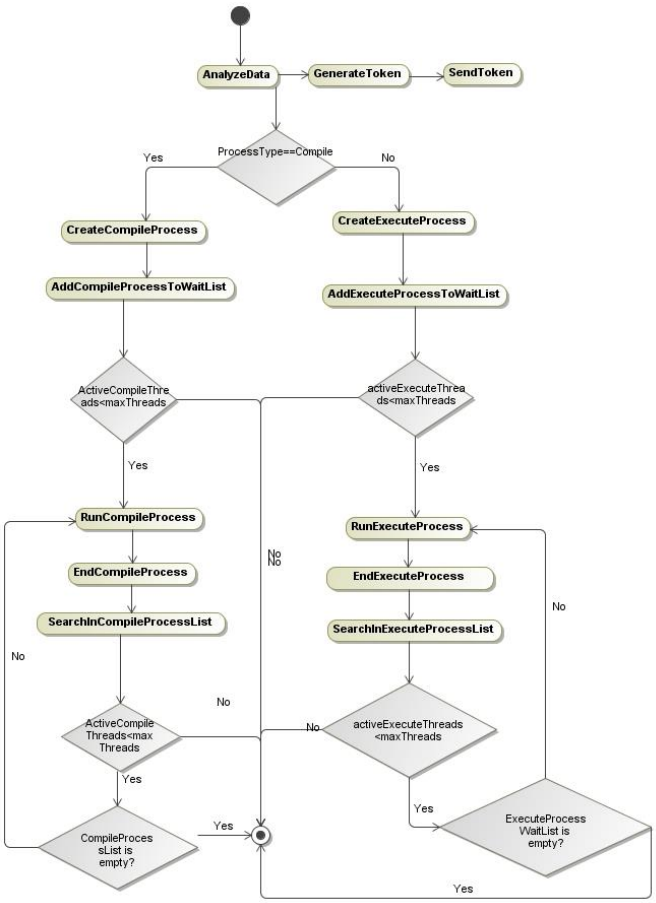

Fig. 2. The algorithm of data analysis and job processing

C. Software Generation Process

The architectural decision to enable users to write their software in C/C++, Java, Fortran 90, and QT programming languages was made in the design stage of WikiSPSM. In order to implement this idea, command-line interfaces were chosen as the architecture of communication between the UI and software generator part. Software generator must perform the following actions:

- Compilation,
- Job submission (to cloud or to single server),
- Job monitoring,
- Job and its results control.

External command-line compilers were chosen for the compilation of the programs. Theoretically, our system can be configured to work with any programming language if only its command-line compiler (or compilation service) is available. The requirement to have command-line user interface is necessary for user's programs also. In the current version, WikiSPSM is not able to work with the external programs using GUI.

It is planned that compilation and execution must be made on the server side. Note that "server side" here means not only one server, but can be considered as a cloud also (see Subsection D). The server application consists of three classes: Server, Task, and vrMath. At the application initialization, time object vrMath is created, and then it creates the object Server. These objects help to monitor jobs and to manage them interactively, including abort operation.

The algorithm of the program data analysis and job processing is shown in Figure 2. Object Server creates an object Task for each submitted data array received from the portal. Task object parses the data and sends a Token back to the user (via portal). Token is necessary to identify the job, monitor it, and get its results. When the process Task is finished, it passes the data to Server object and then the process of compilation of execution begins. All the jobs are scheduled and queued. If number of resources (e.g. number of CPU) is less than requested, job waits the end of some other processes. After job completion, its result is stored in the database.

D. Bridge to Distributed Systems

In its early stage, WikiSPSM was implemented as a server-based distributed program [1]. However, it was observed that increased number of users and submitted tasks have negative impact on the performance of the system. For example, while testing the instability of the system, it was observed when a job tried to use more than 3 CPUs on one server. In order to solve this issue. the architecture of the system has been changed.

In revised architecture of the system. Wiki-based portal and other components remain more or less the same. However, the software generation component is changed dramatically. The adoption of this legacy component is made in two stages:

- *Transformation between different operating systems*. Initially, the server-side application was hard coupled with the Windows operating system, because of used before command-line compilers and Qt library. All this part was redesigned and new Linux-based implementation is made. The changes not only improved the system but also made it completely multiplatform.
- *Transformation between the paradigms*. For better throughput of computing application, server was redesigned to schedule jobs in wide-distributed environments, namely Ubuntu One and Open Stack private clouds (Figure 3). Besides the central DB distributed file system with NFS file system is used for the communication of working nodes. During the development and for the testing a set of VirtualBox virtual machines have been used.

Early test of redesigned component shows very good results (see Section 3).

III. Test Results

WikiSPSM is tested both in technical and scientific levels. The set of programs was made using Fortran 90, Java, C, C++, and NetLib repository LAPACK API.

The set of scientific tests consisted of a) the comparison of the results of well-known problems with the results given by the system; b) the comparison of the results of sequential and parallel implementations of the same Monte Carlo method-

based algorithm. All the tasks finished and show expected results only. The participants of the pilot project named the possibility to change and extend programs from the repository as a very useful and efforts-sparing option.

The set of technical tests consisted of the tests of program compilation, execution, data exchange, and user-computer interface support processes. For the system load testing, a program for Monte Carlo algorithm implementation was chosen. On its basis, 500 requests for the server side were generated. The server with 4 CPUs was chosen as an abstraction of the distributed system. The results of this test show (Table 1) that the system acts as a common task distribution system and the given job submission and job execution times fit the Amdahl's law very well.

For 150 jobs, Monte-Carlo problem using new (bridged to distributed systems) execution component was solved two times faster than initial server-based application component. The "toy example" (calculation of the factorial of big numbers) – was solved eight times faster.

IV. Related Work

Our system (in short WikiSPSM) has been compared to other commercial and open-source products. Here the comparison with two most distinguished products (each present the set of the tools) is presented.

A. Mathematica

Wolfram Research Mathematica is a multiplatform tool with high functionality. It consists of the core and user interface components. The core is able to parse Mathematica programming language, to call external command-line software, process with files using default programs, commutate via e-mail, and perform dynamic linking of the functions, in such a way as to extend the functionality of the core. The core with UI communicates using MathLink protocol. For the communication with external software .NET/Link and Java/Link are offered, which enables to call Mathematica's functions from e.g. Java program and vice versa [3, 8].

TABLE I. The Results of System Load Test

Number of processes	Time for all job submission	Time for all jobs completion
1	12 s	14 min. 21 s
2	16 s	9 min. 41 s
3	22 s	9 min. 51 s
4	NA	NA

TABLE II. The Criteria of Comparison

Criteria	Weight coefficient
Reusability of Mathematic functions, models and programs	0.3
Possibility to extend system repository by new functions, models and programs	0.2
User interface	0.2
Programming languages support	0.1
Programs' execution	0.1
The possibility to integrate system in other systems	0.1

There are two tools under Mathematica umbrella, which can be used as components for a stochastic programming and Statistical Modelling System: *Grid Mathematica* and *Web Mathematica*.

Grid Mathematica is able to perform calculations using distributed infrastructures. This tool is the subject of additional licensing and supports up to 16 cores per task.

Web Mathematica is based on Java applets and servlets. It can perform asynchronous computations only and is useful as a different type of UI only.

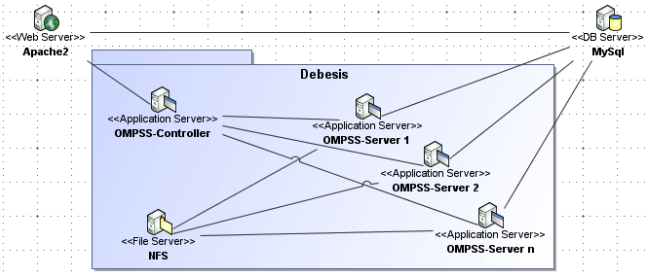

Fig. 3. The architecture of WikiSPSM with the cloud computing component

Grid Mathematica and Web Mathematica cover only two aspects of our System: distributed computation and web-based interface. However, the users of Mathematica aren't able to make new module dynamically, to add functionality to Mathematica Core dynamically, to reuse programs of other users. In Web Mathematica case, only the site administrator has rights do refining of the core. Therefor the use of Mathematica in Wikinomics context is problematical.

B. Scilab

Scilab Scientific is chosen as an example of open-source tool because of the functionality. The tool is oriented to solve problems of linear algebra, matrixes, polynomials, statistics, differential equations etc. Because of this, by the functionality Scilab can be named as a good competitor of commercial systems. Scilab is multiplatform tool; it supports all recent operating systems. Scilab provides links between it and C, C++ and Java. Scilab native programming language subroutines can be called for C, C++, and Java programs. There is a possibility to import Matlab code, because of the similarity of Scilab native language and Matlab language. However, in contrast to Mathematica, Scilab does not have task parallelization component or web UI component [4, 5].

C. Comparison

Our system and the two competitors described above have been compared using the list of criteria (see Table 2). Most important criterion is the reusability of artifacts, because on it depend the complexity of scientific software and the amount of time required to make new solutions. Second criterion is the possibility to extend the system with new artifacts: functions, models, and programs. The possibility of using the system artifacts from external sources (programs) is no less important.

These three criteria together have direct relation with the idea of wikinomics in scientific software domain.

All three compared products (see Table 3) have rich set of supported mathematical functions and API's. Mathematica has the most rich list of functions and algorithms for the problems of mathematical programming. WikiSPSM uses NetLib repository LAPACK [9] for C++ and FORTRAN and as a consequence provides more functionality as Scilab.

Users of Mathematica or Scilab can reuse their functions directly. In contrast, WikiSPSM does not offer such possibility, because it is web based and all the programs are executed on the server side, not locally.

As could be predicted, WikiSPSM shows best result on second criterion – the possibility to extend system repository. All artifacts are stored in public database, can be reviewed, reused, and appended. 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.

Comparing the systems by the user interface criterion it's easy to see that WikiSPSM has UI with a reach not as large as Mathematica or Scilab in terms of common functions. (e.g. possibility to show 2D and 3D graphical objects). However, WikiSPSM in its UI has a lot of Wiki-oriented functions (in contrast to other systems) and web-orientation (in contrast to Scilab).

TABLE III. THE ASSESSMENT OF THE SYSTEMS

Criteria	M	S	W
Reusability of Mathematic functions, models and programs	10	7	8
Possibility to extend system repository by new functions, models and programs	3	3	9
User interface	9	5	7
Programming languages support	6	6	8
Programs' execution	9	5	7
The possibility to integrate system in other systems	10	6	2
Total	7.9	5.4	7.3

M – Mathematica, S – Scilab, W – WikiSPSM (our system). The total value is calculated as a sum of values multiplied by the weight coficients (Table 2)

By the criterion of programming languages, WikiSPSM is clear winner also because it supports C, C++, Java, QT, and FORTRAN in contrast to others, which support only two to three external languages.

By the last criterion (the possibility to integrate the system itself with other IS), WikiSPSM can be assessed by the lower mark than Mathematica and Scilab. Its integration process and possibility of interoperability is the same as of some other application server-based systems. However, this is not a big disadvantage because our goal was to build this system as a portal, as a central entry-point, not as a component.

V. CONCLUSIONS AND FUTURE WORK

The following conclusions can be made:

1) The system of Wiki-based Stochastic Programming and Statistical Modelling is developed and it is adopted to cloud computing infrastructure. WikiSPSM can solve various optimization problems using C, C++, Java, Fortran, programming languages and Qt and Netlib Repository LAPACK libraries. The system already used to solve two stage stochastic programming problems.

2) The system enable scientist to collaborate and make direct evolution of the models, algorithms and programs. WikiSPSM expands the limits of common mathematical software.

3) WikiSPSM can be considered as the job submission and management system for private academic cloud or other distributed infrastructures. It has all the necessary components despite the fact that the system is build from scratch using open source technologies only.

The results of the project will have direct positive impact on development of scientific software. Since the gap between the two technologies is bridged, each of them promises good performance. The power of the wiki-technologies will ensure the ability of the interactive collaboration on software developing using the terms of particular domain.

Despite the success of the project, some future work is still required:

1) To increase the level of the security, automatically inspect what files and data are stored (or going to execute) in the server side and prevent hazardous actions.

2) To facilitate the configuration of the system and in such way to expand the list of supported programing languages.

3) To refine job scheduling, distribution and parallel execution algorithms by adding stochastic elements.

4) To make a bridge between WikiSPSM and European Grid Infrastructure or commercial infrastructures.

Continued testing of WikiSPSM is planned by solving the following problems: power plant investment planning by stochastic programming; two-stage stochastic programing problem for short term financial planning.

REFERENCES

[1] V. Giedrimas, L. Sakalauskas, K. Žilinskas. Towards the environment for mass-collaboration for software synthesis, 2011. EGI User Forum 2011. pp. 1-2 [online] URL: https://indico.egi.eu/indico/event/207/session/15/contribution/117/material/slides/0.pdf

[2] V. Giedrimas, A. Varoneckas, A. Juozapavicius. "The grid and cloud computing facilities in lithuania." In *Scalable Computing: Practice and Experience*. vol. 12(4). pp. 417–421, 2011.

[3] S. Steinhaus. Comparison of mathematical programs for data analysis. Munich, 2008.

[4] M. Baudin. Introduction to Scilab. The Scilab Consortium, 2010.

[5] C. Bunks, J.-P. Chancelier, F. Delebecque. C. Gomez, M. Goursat, R. Nikoukhah and S. Steer. Engineering and scientific computing with Scilab. Birkhauser Boston, 1999.

[6] ESSL and Parallel ESSL library [online] URL: http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp.

[7] D. Tapscott, A. D. Williams. Wikinomics – How mass collaboration changes everything. Atlantic Books, 2011.

[8] Mathematica Documentation Center [online] URL: <http://reference.wolfram.com/mathematica/guide/Mathematica.html>.

[9] V.A. Barker et al. LAPACK User's Guide: Software, Envinroments and Tools. Society for Industrial and Applied Mathematics. 2001

[10] L. Sakalauskas. Application of the Monte-Carlo method to nonlinear stochastic optimization with linear constraints. Informatica, Vol 15, No 2, pp. 271-282, 2004.

AUTHOR PROFILE

Vaidas Giedrimas is a member of the IEEE and the IEEE Cloud computing SIG, a member of Lithuanian Computer Society. He holds PhD from Vytautas Magnus University (Lithuania) in Informatics. He is associate professor of Department of Software Systems at Siauliai University, Lithuania. His research interests include Distributed software systems (grid and cloud computing, component-based software engineering, service-oriented architecture) and methodologies of automated software development. He is the member of the Management committee of COST Action IC1201 Behavioural Types For Reliable Large-Scale Software Systems (BETTY) and Lithuanian representative at NorduGrid and Baltic-HPC associations.

Kestutis Zilinksas is a member of the Council of the Lithuanian Operational Research Society. He holds PhD from Vytautas Magnus University (Lithuania) in Informatics. He is associate professor of Department of Informatics at Siauliai University, Lithuania. His research interests include stochastic programming, Monte Carlo method, and numerical methods in optimization.

Leonidas Sakalauskas is the main author of the presented project idea. He holds the professor position at Department of informatics, Siauliai University and principal researcher position at Operational research sector at Systems analysis department. He also is the president of the council of the lithuanian operational research society. His research interests consist of operational research, artificial intelligence, data mining and information systems (business informatics).

Nerijus Barauskas holds MS from Siauliai University (Lithuania) in Informatics. He has rich experience as Web Application Developer and Cloud computing infrastructures builder in Lithuania and Ireland. Currently he is working as System Developer at Imagine Communications Group (Ireland)

Marius Neimantas holds MS from Siauliai University (Lithuania) in Informatics. Currently he is working as Chief specialist at Informatics department, National Low Administration (Lithuania). Security is the main focus of his reseach interests.

Remigijus Valčiukas holds MS from Siauliai University (Lithuania) in Informatics. Currently he is working as Software Developer at Šiaulių bank (Lithuania). His reseach interests consist of software development with SQL, C++, QT, PHP, JavaScript and jQuery programming languages, Oracle and other databases.