Architecture of a Flexible Web-Based Framework for Building Models and Solving Decision Optimization Problems

B. Staykov[1], F. Andonov[2],D. Vatov[1], K. Genova[1], L. Kirilov[1], V. Guliashki[1]

Abstract – **A flexible web-based framework for multiple criteria decision support is presented. The system is targeted at different types of users – researchers, educators and business people and should facilitate the problem solving process of different types of optimization problems, mainly single and multi-objective linear programming problems with continuous and/or integer variables.**

Keywords – **Web-based systems, decision support, single and multiple objective optimization.**

I. INTRODUCTION

The evolution of decision support systems can be traced back to the dawn of computers. "First-generation" systems were *single computer - single user* type and implemented only one method or a few similar methods of one type (see [17, 18] for example). With the spreading of the Internet some network-based systems appeared – both web-based and traditional client-server(A good survey about such systems is given in [12]). Yet most of the systems were problem-oriented and they could only solve one specific problem(see [15, 16] for example). Many of them were based on the java applet technology, but this approach has many limitations. This architecture is still basically a client-side technology, and is limited by the resources of the client machine and hence the data volume, memory, CPU power and problem complexity. Later on group decision support systems gained popularity, where not a single decision maker is responsible for the decision but a group of decision makers.

Web architecture by itself does not allow persistent connections, and because of that there are limits on the response time and the amount of data exchanged between the server and client, but the solving time for this kind of decision problems is highly undetermined. As a result, the usability of such systems is limited to problems whose solving time does not exceed the browser time-out interval.

The development of the Internet and its total penetration in all social areas combined with globalization brought a new breed of systems – less monolithic, more versatile, incorporating cutting-edge software technologies [12, 14].

The modern Internet technologies provide platform-independent, remote computation, as well as exchange of complex multimedia information. The end users of decision support systems can focus their efforts on problem analysis and decision making.

The use of Decision support systems in general, requires specific knowledge background about the methodology of mathematical optimization and its applicability to the user's professional area.

As a result of the above, it is a challenge to create a universal system, which is method-, user-, solver-agnostic and applicable to a wide range of example, research and business problems. In this paper, we describe a software system which represents our understanding of what the architecture of such a system should look like.It is under the provisional name WebOptim.

II. IMPORTANT ADVANTAGES OF WEBOPTIM

Web-accessible: The researchers [11, 12] in the area of operational research (OR) consider that the development of Web-access technologies is of key importance for the usability of optimization, so as the name suggests, the system should be accessible from the Web via a browser, and users should be able to define, solve, save, load and share their decision problems.

User-agnostic: WebOptim should allow to be used by different types of users – educators, who demonstrate the large variety of single and multiple criteria methods on different size and type of problems, researchers, who will test their own methods or/and solvers and business people, who will solve their real-world decision making problems with a method of their choice. The diversity of users and goals translates into the requirements of a highly-customizable user interface and an extendible framework. Users should be able to define their own problems, to solve them with any applicable method, save them for later evaluation, andbe able to see examples or similar problems. Another way for broadening the user target group is by offering well-formed and solved examples of typical optimization problems. Less-experienced users will be able to browse these examples and hopefully find one similar to their own problem.

Solver-agnostic: WebOptim should incorporate several solvers with metadata about their applications and methods they are used by. System administrators will be able to add new solvers, written in different programming languages.

[1]Boris Staykov, E-mail:bstaykov@iinf.bas.bg
Daniel Vatov, E-mail: daniel.vatov@gmail.com
Krasimira Genova, E-mail: kgenova@iinf.bas.bg
Leoneed Kirilov, E-mail: lkirilov@iinf.bas.bg
Vassil Guliashki, E-mail:vggul@yahoo.com
are with the Institute of Information and Communications Technologies- BAS, 1113 Sofia, "Acad. G. Bonchev,bl. 2, Bulgaria.

[2]Filip Andonov is with the NewBulgarianUniversity, Sofia 1618, z.k. Ovtcha Kupel,.Montevideo str. № 21, Bulgaria, E-mail: vonodna@yahoo.com

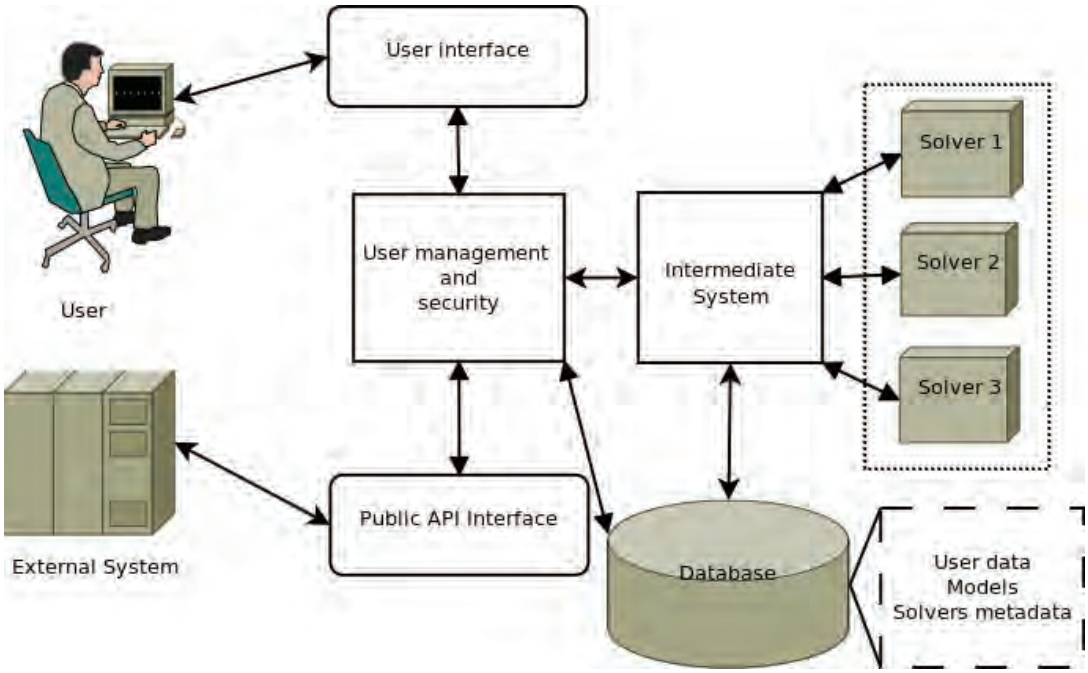

Fig. 1. Architecture of WebOptim

Method-agnostic: WebOptim should have metadata for all the included methods, their applications and the solvers they use. System administrators will be able to add new single or multiobjective solvers.

Heterogeneous: The system should allow to be continuously developed by several teams, using different software technologies. That is why a reliable and flexible environment for data and command interchange is needed to provide the common ground. The authors think that web-services are the answer to this requirement. Using web-services allows developers to choose a programming language, operating system and software design of their own taste and to glue their piece of the puzzle with as little effort as possible.

III. Structural Description

For maximum modularity the system is composed of the following main components: user management and security, intermediate module, database and solvers. The principle schema of this intermediate system is given in Fig. 1.

A. User management and security

In order to use the system, each user must create its own personal profile. The profile containing login credentials, user type, personal details, defined problems, their solutions and additional information about errors and specific solving details will be stored in a common database. The registration of users will be done through standard web page forms. After registering and logging in their profile, the users will be provided with interface for defining and solving problems. Those problems must be defined by one of the below described standard syntaxes. After syntax verification, each problem will be stored in the database with unique ID parameter. The user will be able to see a list of these defined problems and their status. There will be four possible states of this problem status:

- "Not solved". When the problem is defined but still not sent to the solver;
- "Waiting for solution". When the problem is submitted to the solver queue and solution is expected when ready. In this state the user should not be able to edit the problem definition;
- "Solution obtained". When a solution is obtained from the solver and is ready to be previewed;
- "Solver error". The solver returns some error.

This part of the module will provide tools for the user to preview problems and solutions; start or interrupt solution process for certain problem; preview error messages received from the solvers and to obtain specific detailed debug information about the solving process (if the chosen solver provides this information).

B. Database

The main database will store all the data for the users, problems they have saved, solutions and metadata about the methods and solvers for use in the intermediate system. The metadata should contain at least the solver's name, description, type (integer, mixed integer, multicriterial, etc.) and type(s) of accepted input. Because of the network environment the system will be running in, if the solvers need a database for temporary data during the solving process, they can use their DBMS and databases to minimize the communication volumes and thus reduce solving time. To

overcome the lack of connection persistence between the client and the server in web-based systems all intermediate and final results are stored in the database, which is possible due to the nature of these types of optimization problems, where every intermediate result can be used as a reference point for the next(every method can be used as interactive).This also allows unbindingof the calculation process from the user interface. The user is no longer required to wait for the solution and to keep the connection alive because all the results will be stored in database and associated with his/her personal profile and problem.

C. Solvers

All solvers will implement a common XML SOAP[2, 3] protocol for communication with the intermediate system. This allows diversity of the solvers, the machines they are running on, the language they are written in, etc. The limitation of this approach is that due the asynchronous nature of this type of communication, the control will be harder to implement, because in a simplified way the communication looks like this: intermediate system sends problem for solving and when ready, the solver gives an answer or an error (if occurred). Because there are many problem definition formats, it is certain that at some point in time there will be a set of solvers that support a non-intercepting set of syntax formats. An intermediate module will translate the problem definition to a format understandable by the solver. Another way of interaction will be optimization problem data to be delivered directly to the solver. This will be the case of already existing solvers or solvers that are considered useful for the system's users. They will have their own syntax for the input. Besides AMPL as a modeling language there are many other modeling languages used by quality solvers that is worth integrating into the system. Possible modeling formats are listed bellow [9]:

- MPS file format - The MPS format is supported by most lp solvers and thus very universal. The model is provided to the solver via an ASCII file. This format is very old and difficult to read by humans. See [8] for a complete description of the format.

- lp file format - The lp format is the native lpsolve format for providing LP models via an ASCII file to the solver. It is very readable and its syntax is very similar to the Mathematical formulation. See [6] for a complete description of the format

- CPLEX lp file format - The CPLEX lp format is another format for providing LP models via an ASCII file to the solver. It is very readable and its syntax is very similar to the Mathematical formulation. It is a format used by the CPLEX solver. See [4] for a complete description about the format

- LINDO lp file format - The LINDO FILE format is another format for providing LP models via an ASCII file to the solver. It is very readable and its syntax is very similar to the Mathematical formulation. It is a format used by the LINDO solver. See [7] for a complete description about the format.

- GNU MathProg file format - The GNU MathProg format is another format to provide LP models via an ASCII file to the solver. It is very readable and its syntax is very similar to the Mathematical formulation. It is a format used by the GLPK solver and a subset of AMPL. It has also the possibility to use loops. See [9].

- LPFML/OSIL XML file format - The LPFML XML format, currently named OSIL after becoming part of COIN-OR project (https://www.coin-or.org/), is another format to provide LP models via an ASCII file to the solver. This format is very recent and uses XML layout. It is not very readable by humans, but because of the XML structure is very flexible. For more informationsee [10].

Solvers will be exposed to external systems following Web Services [1] paradigm.

D. Intermediate core system module

The fact that the system will be used as an educational and research tool means that it has to be as open and extensible as possible, so new methods, solvers, problem types, and basically every kind of modules can be added later. The development team is heterogeneous and not static, especially for the projected lifetime of the product, so different software libraries and components and different program languages will be used in it, which put a lot of stress on the system architectural design. To make these different parts glued together several modern technologies and principles will be applied to the development process – extensive use of XML as communication standard and web services.

There is one major problem that concerns the intercommunication between the different modules in the system. The problem appears because this intercommunication needs to be done asynchronous due to its nature - mainly HTTP and SOAP protocols, which have very tight response timeouts. Depending on its size, each solving problem will take different time to be solved by a certain solver and sent back to the sender module. This makes it impossible for the communication to be done in an uninterrupted cycle of request-response.

This problem is solved by bringing in an intermediate system module, which will take care of all the communication between end user interfaces, database storage and solvers. This system will perform two main tasks – user management and solvers communication.

E. Solvers communication

On the other hand, this intermediate system module will handle the bidirectional communication with solvers, based on the XML SOAP protocol.

There are two subtasks in this part of the module – sending requests to the solvers and providing a web service for handling solver responses for the solutions of the problems.

Sending a solver request will be done by posting data to the unified solver web service through XML SOAP protocol and will contain the following mandatory fields:

- Problem ID (an integer number field);
- Problem syntax (a string field (different syntaxes are described in another part of the article));
- Problem definition (a string field);

- Debug (a Boolean field) – defines if additional debug information is requested.

Additional fields may be implemented at later stage of the project. After posting this information, a confirmation response is expected from the solver and the problem state is changed to “Waiting for solution”. The solver response web service will expect information posting from solvers (again in a standard XML SOAP protocol) and will look for the following fields:

- Problem ID (an integer number field);
- Problem solution (a string field) - in case of a successful solution;
- Solver error message (a string field) - in case of a solver error;
- Debug information (a string field) – in case it is requested.

When such a request is received, the module will perform search in the database for the certain problem ID, store the solution (or the error message) and change the problem status to “solution obtained” or “solver error”, according to the current case. The web service must response to the solver request; otherwise the solver must continue to try to post the solution after a certain amount of time, until a confirmatory answer is received!

IV. Conclusion

There exists a variety of stand-alone, network and web-based decision support systems. Despite the fact that with the increased demand for such systems in global economy, the development of semantic technologies and many R&D teams working on less fragmented, more universal systems, there is still room for improvement. The key features of the proposed web-based system WebOptimare:

• Useful to a wide variety of users from different professional backgrounds with different level of optimization competence.

• A user friendly customizable interface, reflecting the needs of different users and accessible worldwide via the Web;

• A set of solvers which covers the most popular optimization and decision making problems;

• Designed to be easily extended by adding new solvers;

• Providing an API interface for external use by third party developers.

By implementing our view of such architecture, we intend to overcome the inherited limitations of web-based systems in general and target a larger user group. Several other similar systems exists, but they are not widely used, mainly because specialists from different areas do not recognize their problems as optimizationproblems and they are not familiar with the available software instruments for decision support. In that respect, WebOptim is an attempt to promote and encourage the use of decision support optimization systems in all areas where such problems occur.

One future direction for the development of WebOptim is to make it able to solve single and multi-objective nonlinear optimization problems by implementing the necessary solvers, methods and user interfaces.

Acknowledgement

This research is supported in part by the Bulgarian National Science Fund, Grant No DTK02/71 and IICT-BAS research project Modeling, Optimization and Multiple Criteria Decision Making.

References

[1] Web Services Architecture. Technical report, World Wide Web Consortium, February 2004.
[2] Paul V. Biron and Ashok Malhotra, editors. XML Schema Part 2: Datatypes. W3C Recommendation. W3C, second edition, October 2004.
[3] http://www.xml.com/pub/a/2000/02/09/feature/index.html
[4] P. Notebaert K. Eikland. Cplex file format. http://lpsolve.sourceforge.net/5.5/CPLEX-format.htm.
[5] P. Notebaert K. Eikland. Lindo file format. http://lpsolve.sourceforge.net/5.5/LINDO-format.htm.
[6] P. Notebaert K. Eikland. Lp file format. http://lpsolve.sourceforge.net/5.5/lp-format.htm.
[7] P. Notebaert K. Eikland. lp_solve reference guide. http://lpsolve.sourceforge.net/5.5.
[8] P. Notebaert K. Eikland. Mps file format. http://lpsolve.sourceforge.net/5.5/mps-format.htm.
[9] A. Makhorin. Gnu linear programming kit, modeling language gnu mathprog.http://plato.asu.edu/gnu_mp.pdf.
[10] A. Makhorin. Optimization services linear language https://www.coin-or.org/OS/OSlL.html
[11] P. Valente, G. Mitra. The evolution of web-based optimization: From ASP to e-Services. Decision Support Systems, Volume 43, Issue 4, (2007) 1096–1116.
[12] H. K. Bhargava, D. J. Power and D. Sun, “Progress in Web-based decision support technologies”, Decision Support Systems, Volume 43, Issue 4, August 2007, pp. 1083 – 1095
[13] A.M. Geoffrion, R. Krishnan, Prospects for operations research in the E-business era, Interfaces 31 (2) (2001)], 6-36.
[14] M. Andersson, H. Grimm, A, Persson, Amos Ng, A Web-basedsimulationoptimizationsystemforindustrialscheduling, In: Proceedings ofWSC '07, (Henderson, S. G. et al, eds.), IEEE Press Piscataway, NJ, USA 2007.
[15] Plácido Rogério Pinheiro1 and José Auriço Oliveira,WEB - Based Optimization System Applied to High School Schedule Building, http://www.asap.cs.nott.ac.uk/patat/patat04/553.pdf.
[16] P. Korhonen, A Visual Interactive Support System for Multiple Criteria Decision Making, Belgian Journal of Operations Research, Statistics and Computer Science, 27 (1), 1987, 3-15.
[17] V. Vassilev, B. Staykov, F. Andonov, K. Genova, M. Vassileva, Multicriteria Decision Support System MOLIP, Cybernetics and Information Technologies, 2 (1), 2002, 3-15.