Bolshoi - A Modeling Spreadsheet

(Improving Usability of Complex Analytical Modeling Tools)

——— Project Summary ———

Technical Report: CS-TR-4110

Department of Computer Science, University of Maryland at College Park

William C. Cheng
william@cs.umd.edu

Leana Golubchik
leana@cs.umd.edu

Abstract

Spreadsheet programs are very popular financial modeling tools because they allow users to juggle numbers and formulas with a powerful yet intuitive and easy to understand user interface; also, they often are equipped with sophisticated numerical analysis packages for data analysis and powerful presentation utilities for visualizing results. Computer systems performance and reliability modeling tools of today, on the other hand, have un-intuitive user interfaces and are difficult to learn and use. In this work, we propose to design, build, and evaluate *βolshoi*, a *modeling spreadsheet*, with the goal of putting modeling tools comfortably in the hands of non-expert users.

In this proposal, we address management of complexity that exists in performance and reliability analysis of real computer and communication systems. Specifically, we propose to do so through the design and development of an advanced modeling tool. Our tool will provide two important functions: (1) a proper interface for building models that will allow system designers not just to define their models, but visualize them in various ways and (2) easy plug-in of existing and future advanced solution techniques. We call this tool *βolshoi*, a *Modeling Spreadsheet*, because it has a spreadsheet-type interface as detailed below.

Performance evaluation of real systems is complex, suffers from scalability problems (or the so-called "state explosion" problem) and in many cases requires advanced computational techniques. Often, advanced computational techniques are based on exploitation of "special structure" in the models (the primary way to deal with state explosion besides getting a bigger machine). With large and complex models, these special structures are very expensive to expose automatically as it involves searching through a combinatorial number of permutations. Proper visualization of models can greatly assist in the discovery of these special structures so that state space reduction techniques can be applied. Discovery of special structure regularly contributes to many orders of magnitude in computational efficiency. Furthermore, models are often defined over infinite state spaces. We believe that a spreadsheet paradigm is ideal for visualizing such models.

Without proper modeling tools, much effort and money is wasted by the computer industry, and moreover, the probability of a successful outcome is low. Thus, a good tool is crucial to advances

in the state of the art in performance modeling as well as to successful design of systems in the industry. Every system designer should be able to integrate the use of a performance modeling tool into his/her design process. He/she should be able to easily ask "what-if" type questions, explore possible design choices, and make decisions based on quantitative results rather than "gut feeling". We believe that a modeling *spreadsheet* is the right abstraction for such tasks, and furthermore, to the best of our knowledge this abstraction has not been exploited for performance evaluation tool purposes.

We believe that the approach proposed here will have a significant impact on future performance tool designs as well as make significant strides in wide-spread use of performance evaluation techniques among computer and communication system designers.

Furthermore, a modeling tool that does not require expert-level methodology knowledge is also an excellent undergraduate-level and graduate-level *educational* tool. Opportunities for hands-on experience with modeling and performance evaluation as well as the ability to add new techniques to the tool greatly improve the educational experience of students and their future ability to apply what they have learned in class to design of real computer and communication systems.

1 Introduction

Spreadsheet programs are very popular financial modeling tools because they allow users to juggle numbers and formulas with a powerful yet intuitive and easy to understand user interface; also, they often are equipped with sophisticated numerical analysis packages for data analysis and powerful presentation utilities for visualizing results. Computer systems performance and reliability modeling tools of today, on the other hand, have un-intuitive user interfaces and are difficult to learn and use. In this work, we propose to design, build, and evaluate *βolshoi*, a *modeling spreadsheet*, with the goal of putting modeling tools comfortably in the hands of non-expert users.

1.1 The Need for a New Modeling Tool

Modeling is an established and accepted part of the design process of many large and complex systems, for instance, architectural designs, aircraft designs, and many more. Modeling has also become an integral part of some aspects of computer systems design, for instance, telecommunication systems. However, many areas of computer systems design still lack proper use of modeling techniques and tools. If we are to build large complex computer and communication systems which exhibit the necessary performance characteristics, for instance, such as large scale clusters needed for web servers and search engines, then we must make modeling and performance evaluation an integral part of the entire computer and communication systems' design and development process. Part of the difficulty in realizing this goal is a lack of "easy to use" performance evaluation tools, which in turn is partly due to the fact that most performance evaluation tools in existence today effectively require the user to first become an expert in the methodology on which the tool is based. Furthermore, many of the existing tools simply do not scale up when used to model and evaluate large systems — the scalability problem also results in the need for the tool user to be a methodology expert, i.e., in order to devise models that this particular tool is able to handle.

1.2 Our Goals

In this proposal, we address management of complexity that exists in performance and reliability analysis of real computer and communication systems. Specifically, we propose to do so through the design and development of an advanced modeling tool. Our tool will provide two important functions: (1) a proper interface for building models that will allow system designers not just to define their models, but visualize them in various ways and (2) easy plug-in of existing and future advanced solution techniques. We call this tool *βolshoi*, a *Modeling Spreadsheet*, because it has a spreadsheet-type interface as detailed below.

Performance evaluation of real systems is complex, suffers from scalability problems (or the so-called "state explosion" problem) and in many cases requires advanced computational techniques. Often, advanced computational techniques are based on exploitation of "special structure" in the models (the primary way to deal with state explosion besides getting a bigger machine). With large and complex models, these special structures are very expensive to expose automatically as it involves searching through a combinatorial number of permutations. Proper visualization of models can greatly assist in the discovery of these special structures so that state space reduction techniques can be applied. Discovery of special structure regularly contributes to many orders of magnitude in computational efficiency. Furthermore, models are often defined over infinite state spaces. We believe that a spreadsheet paradigm is ideal for visualizing such models.

Using the spreadsheet abstraction, existing model solution techniques can be packaged as "operators" in a modeling spreadsheet where a model can be transformed (or solved) using an operator to produce one or more models (which may be of different types). Therefore, we also view the Modeling Spreadsheet as a tool for integrating existing and future solution techniques.

Without proper modeling tools, much effort and money is wasted by the computer industry, and moreover, the probability of a successful outcome is low. Thus, a good tool is crucial to advances in the state of the art in performance modeling as well as to successful design of systems in the industry.

1.3 Our Vision

Every system designer should be able to integrate the use of a performance modeling tool into his/her design process. He/she should be able to easily ask "what-if" type questions, explore possible design choices, and make decisions based on quantitative results rather than "gut feeling". We believe that a modeling *spreadsheet* is the right abstraction for such tasks, and furthermore, to the best of our knowledge this abstraction has not been exploited for performance evaluation tool purposes.

We believe that the approach proposed here will have a significant impact on future performance tool designs as well as make significant strides in wide-spread use of performance evaluation techniques among computer and communication system designers.

Furthermore, a modeling tool that does not require expert-level methodology knowledge is also an excellent undergraduate-level and graduate-level *educational* tool. Opportunities for hands-on experience with modeling and performance evaluation as well as the ability to add new techniques to the tool greatly improve the educational experience of students and their future ability to apply what they have learned in class to design of real computer and communication systems.

2 Challenges

In this section we first discuss the need for and appropriate use of analytical models in the design and development process of computer and communication systems, from the point of view of system performance. We then give a brief overview of problems and challenges that a system designer and performance analyst face, as well as shortcomings of existing tools intended for system performance evaluation purposes.

2.1 A Case for Analytical Models

Design choices in complex systems are non-obvious and mistakes in the design process can be costly — as stated in [8]:

> "... it is easy to cite an ongoing litany of multimillion-dollar computer systems that have failed as a consequence of a short-sighted approach to system design (e.g., see [18, 20])."

Hence, we believe that it is crucial to employ modeling tools and techniques at design-time, to facilitate the asking of "what-if" type questions and use the answers to guide the design process and making of design choices.

There are two basic types of model evaluation techniques - analytical and simulation-based. Ideally, the "what-if" process should be *interactive*; hence, it would be inappropriate to have a simulation back-end for tools intended for use at design-time, since simulation is an inherently time consuming process. Consequently, we focus on analytical techniques. The key to a successful analytical modeling tool is to make it easy to use, yet powerful, flexible, and fast.

2.2 Problems with Existing Tools

Existing commercial and academic tools suffer from the following shortcomings: (1) lack of applicability to real systems, due to the so called "state explosion" problem, and (2) lack of ease of use, due to the need to be an expert in the tool's methodology.

The "state explosion" problem is due to the need to model *real* systems which often result in extremely large models and sometimes infinite ones (as explained below). Numerous research efforts have focused on development of methodologies (including our own work) [3, 17, 6] that address this "state explosion" problem. However, what is needed (and currently lacking) in order to apply existing methodologies as well as facilitate development of new ones, is a tool that is able to easily incorporate such approaches. For reasons explained below, we believe that a tool that employs a *spreadsheet-style* abstraction, as proposed here, is the right one for these goals.

The lack of ease of use is due to the lack of proper specification, manipulation, and visualization of *models*, i.e., there is a need to visualize and manipulate models, not just the results of using them to evaluate performance of systems. For example, one motivation often cited for use of Petri Net based tools [16] is that they provide a small set of primitives for describing system behavior and therefore are easy to understand. However, even Petri Nets become too complex for a designer to keep track of when modeling real systems [15]. Furthermore, it is difficult to *predict*, based on specification alone, which complex models of real systems will be too "large" for the tool to handle and which will not be. If a petri net is too large it is up to the designer to go through a trial and error process until he/she is able to manipulate the model into a "manageable" state.

The crux of the problem is that existing tools are *too rigid*. By that we mean that only experts in (a) that tool and (b) the modeling methodology implemented by that tool are able to use it successfully. The rest have to be satisfied with toy examples. And, when things do not work as expected (such as the model space is too large for the tool to handle), the designer using the tool is forced to redevelop the model. Ideally we would just like the designer to "tweek" it. That is these tools are not built to support model manipulation and changes properly, and so the designer is forced to "redo" the model rather than work interactively with the tool to come to a point where the tool can "handle" the intended model (with appropriate tweeks).

For instance, in our past work [6] we have had success with developing solution methodologies for models that did not have known tractable solutions, where the "trick" was to tweek the model, in a controlled manner, until (a) it did have a tractable solution and (b) we could *prove* that the new model produced *bounded* errors on performance results of the original model. Having *bounded* errors is key in methodology development. And, the reason we were able to produce such bounds, is because we were able to modify the original model in a systematic manner and keep track of the corresponding changes. Of course, we had to do it all *by hand*. Thus, having a tool that can do this would greatly simplify the process and facilitate development of new methodologies.

In summary, our challenge in this work is to *build a flexible modeling tool that can be easily used by system designers who do not have expertise in modeling methodologies* to guide their design process in an *interactive* manner.

2.3 Infinite Models

We now discuss the importance of being able to specify, manipulate, and visualize *infinite* models, which to the best of our knowledge no existing commercial or academic tool can do in general. This is partly due to the fact that many existing tools must first generate the model's state space, in order to compute the corresponding performance metrics, and of course, actual generation of infinite state space is not possible. Hence, most of these tools require the user to "truncate" the model at specification time (rather than at solution time). Truncation often results in unbounded errors as well as in a loss of the model's original structure.

Although real systems are not infinite, the need for infinite models is real. Such needs are due to:

1. utility of computing asymptotes when evaluating system performance;

2. utility of exploring system bounds, e.g., in order to expose bottlenecks;

3. the need for system sizing, for example for answering questions such as "what is the *right* buffer size for a network switch such that the probability of dropping packets is *sufficiently* low?";

4. the fact that is is sometimes cheaper and easier to solve an infinite version of a model than a very large version of it; this is due to the fact that often there is structure in infinite models which can be exploited (and which is not there in the very large versions of it), e.g., matrix-analytic techniques are often exploited for more efficient solutions of system performance models; however, very few finite forms of such models are known [10]

Constructing tractable solutions for such models often requires searching for special structure. Thus, we next discuss the need for ease of model manipulation and visualization.

2.4 Model Manipulation and Visualization

We have a great deal of expertise in constructing methodologies for efficient computation of performance metrics for large scale models. For instance, we have done work on bounding techniques, as described in the example above. Based on our experience, we know that finding appropriate model structure to exploit for the purposes of an efficient computation can be an "art" — sometimes one might see it right away (if one is lucky) and sometimes not. In either case, such work requires much manipulation of the model (because often it is the ordering of the states that makes all the difference) to massage it into the "right" form. Given existing tools, e.g., Matlab and Mathematica, such complex manipulation must be done by hand along with a "mental" visualization of what would be the product of each manipulation.

Having a flexible mechanism for visualization and manipulation of models would make this "methodology development" process a more structured process than just "dumb luck".

2.5 Model Solution Management

Given the above stated need for model manipulation, part of the tool's essential functionality must include *model solution management* support.

A system designer should not have to come up with the right (i.e., solvable) model on a first try, and neither should he/she have to "redo" the model when it is not solvable by the tool (due to scalability and complexity problems). Tools, such as Matlab and Mathematica, do allow one to manipulate the model, however this is accomplished through essentially general "code development". The ability to do such manipulations is, to the best of our knowledge, completely missing from the higher abstraction based tools, such as Petri Net tools.

Ideally, the designer should not be forced to describe the model based on what is "solvable". That is one reason why designers feel more comfortable with simulation models rather than analytical models, i.e., however they describe the simulation model it is "solvable". We believe that designers should model their systems based on how it should work and what is important about its behavior rather than what is "solvable" by the tool. Of course, solutions of these models are still needed.

Hence, what is really needed in a tool is a way for a designer to "associate" what he/she would like to be the "real" model of their system with what the tool is capable of solving. This capability will facilitate (a) successful use of the modeling tool by designers and (b) successful use of the tool by methodology developers, since in both cases support must exist for model manipulation at solution time.

3 Our Approach — *βolshoi*, A Modeling Spreadsheet

In this section we describe our approach and the corresponding proposed research as well as discuss advantages of such an approach.

3.1 Spreadsheets

The first important part of our approach is a *spreadsheet* style interface which has the advantages listed below.

Note that these advantages and flexibility (as listed below) are due to the fact that in some sense spreadsheets are "structure-free" because everything boils down to the manipulation of a single primitive, a *range of cells*, i.e., we can go from a single cell to a semi-infinite structure in 2-D. Hence, we believe that the spreadsheet abstraction is the appropriate "foundation" for our modeling tool. For instance, we can construct, visualize, and maintain infinite models naturally (as described in the previous section).

This useful interface and abstraction thus gives us the following advantages.

1. Spreadsheets were originally designed as "fancy calculators" for data evaluation and analysis. Today, spreadsheets are an essential business tool, used for a multitude of purposes such as presentation and manipulation of data (not just its evaluation and analysis) by millions of people from a variety of backgrounds. Therefore, as a user interface and as a useful abstraction, spreadsheets are a "proven technology" and hence will be an appropriate interface and abstraction for a modeling tool intended for non-experts.

2. Spreadsheets have also "proven" useful for answering "what-if" type questions easily. The capability to explore "what-if" type questions with ease is essential for system design, capacity planning, and so on.

3. Spreadsheets are also appropriate for model visualization purposes, particularly because the spreadsheet interface makes it easy to visualize only parts of a model, i.e., we do not have to bother with the whole, which is also useful in the case of infinite models.

4. Spreadsheets can be used as a "canonical" intermediate form for model and solution representation, i.e., we can build both front-ends (e.g., petri nets, queueing networks) and back-ends (e.g., various solvers) for it. Thus, it will aid with model manipulation as well.

Hence, part of this research effort will be to explore how far we can push the "range of cells" abstraction in achieving the essential features and goals of a modeling tool (as described in the previous section). We will do this by integrating, into a basic spreadsheet, a collection of solvers (i.e., solution techniques) and by building a collection of applications on top of this, as described in the remainder of this section.

3.2 Models and Operators

The two main "entities" that exist in *βolshoi* are: models and operators. A model is simply a range of cells. An operator is an entity that takes one or more models as input and produces one or more models as output, after some manipulation. We view *βolshoi* as a repository of methodologies and solution techniques, implemented as operators, where the goal is to insure that the tool itself can evolve as new and better modeling techniques become available for improving efficiency and scalability of the modeling process.

There are different classes of operators: (1) model solvers (i.e., traditional solvers which, for instance, compute performance measures, state probabilities, etc.), and (2) model transformers, which are explained in more detail below. Note that both models and operators can be hierarchical, i.e., made up of sub-models and sub-operators.

All operators will be implemented using software components methodology, such as DCOM or CORBA. The component methodology will allow us to provide interoperability between generic solvers (refer to

Section 3.2.1) and specialized solvers that will be created by the modeler using our tool. Furthermore, we will add wrappers to existing solvers (these solvers will include both "home grown" and contributed software, as described in Section 4), in order to make them all interoperate.

We next describe the different classes of operators, and specifically, traditional solvers and model transformation. We also briefly discuss integration issues, including benefits of hierarchical model manipulation.

3.2.1 Traditional Solvers

We will provide the following initial set of traditional solvers, where many of these solvers already exist, and our main efforts here will be to provide wrappers for these solvers in order to integrate them into *βolshoi*:

- *Markov chain solvers* — including direct as well as iterative methods (e.g., power method, SOR, GTH, etc.) [19] as well as structure-based solvers such as matrix-analytic solution methods [10]. The code base for some of these solvers already exists and will be provided by our collaborators (refer to Section 4.
- *Queueing network solvers* — including product-form (open, closed, mixed) solvers (e.g., MVA, AMVA, etc.) and non-product-form extensions [2, 5].
- *Commercial solvers* — including Matlab, Mathematica, and so on.

3.2.2 Model Transformation

As mentioned earlier, discovery of special structure facilitates efficient model solution. However, when such structure is not present in the system model, we can still produce efficient solution techniques by "transforming" the original model into one that does have structure. Such transformations are at the core of the art and science of system modeling.

Thus, we will also develop methodology that will facilitate model "transformation" and keep track of the relationships between the transformed models. This capability will result in significant improvements in efficiency and scalability of performance evaluation techniques that can be used to analyze the original model. This is due to the fact that different analysis techniques can take advantage of different structure that maybe present in the system model. For instance, a Markov chain representation of a system's model might have matrix-geometric form [12] or a queueing networks representation of the same model might have product form [1] — in either case, an efficient solution of the model is possible but undetectable without proper model representation. With proper represenation of a system model, detection of such special structure in the model, or in a sub-component of a model, can be semi-automated.

3.2.3 Integration and Implementation Issues

Integration of performance evaluation techniques is another feature that is key to achieving a high degree of scalability. And, scalability is essential to the success of performance evaluation of large and complex systems. We will design and develop mechanisms for specification of multiple analysis and evaluation techniques of a model. This will be useful in hierarchical performance evaluation and analysis of models. These declarations will be used in algorithms for automatic discovery of analysis "capabilities" of model sub-components, prior to deciding on a plan of analysis and evaluation.

Note that, integration of evaluation techniques is not, in general, a straightforward task. However, such integration can have significant advantages in computing solutions to performance evaluation models. For instance, certain combinations of analytic solution techniques, e.g., as in [11], have the advantage of being easily parallelizable as compared to simulation.

To this end, we will proceed as follows.

- We will start our implementation efforts with a publically available spreadsheet, `gnumeric`.
- We will also begin with a set of available traditional solvers (as described above).
- We will also implement an initial set of applications (as described in Section 3.4).
- Due to the use of component software technology, it will be simple to contribute new solvers as well as other operators, and new applications to *βolshoi*. For instance, this will be done through our collaborating research groups (as described in Section 4) as well as through educational efforts (as described in Section 5).
- Current spreadsheets provide macro languages. However, they are insufficient for our purposes, for instance, there are no provisions for manipulation of infinite models. Hence, part of the effort here will be to extend the existing macro language capabilities for our purposes. This will include, making it sufficiently efficient for large-scale models, providing some form of "lazy evaluation" in order to handle manipulation of (parts of) infinite models, and so on.
- Generic spreadsheets, in general, are inefficient, and can not be used to solve real-size models, at least not when applied to large-scale computer and communication systems. Hence, part of the effort here will be to focus on the spreadsheet's efficiency as well.
- Existing spreadsheet plug-in technology is fairly limited. Hence, part of the effort here will be to extend it and customize it for system performance evaluation needs. Again, extensions that will allow manipulation of infinite models will be one of the significant efforts here.

3.3 Visualization

Generic spreadsheets already have fairly extensive data visualization and presentation capabilities, in the form of graphs, charts, histograms, and so on. However, for our purposes we still need to extend the existing capabilities for such customized support for visualization of infinite models, visualization of scientific data with a set of given semantics (e.g., sparse matrices, block-structured matrices, and so on). Thus, part of this effort will be to provide such extensions.

3.4 Applications

We will also customize *βolshoi* for several applications as a proof of concept. These applications include:

1. *capacity planning and system sizing tool* — for instance, for web hosting applications, where, e.g., the ISP providing the web hosting service would ideally like to determine how to size each part of the system (memory, disks, CPU, etc.) so as to provide a level of quality-of-service commensurate with the customer service agreements but at the lowest cost (to the ISP) possible.

2. *cluster-based computing design tool* — for instance, for "porting" a federated (or an embedded) system to a more generic environment, such as a network of commodity PCs; this is no small task, given that the federated system is likely to have been developed using customized software and hardware, i.e., essentially this is not a port but a re-design of the entire system. Without a fairy extensive performance evaluation process, it would be difficult to say what type of architecture should be targeted for such a system. The specific application we target in this context is a radar application, as we already have expertise in that area [9].

3. *corporate network planning tool* — planning of a corporate network requires answering a multitude of "what-if" type questions; since we believe (as described above) that spreadsheets are well suited for such tasks, we will build this application as a test of the "what-if" capabilities of *βolshoi*.

4. *Markov chain modeling tool* — since we believe (as described above) that spreadsheets are well suited for infinite models and facilitate detection of special structure for efficient solution purposes, we will build this application as a test of *βolshoi*'s capabilities in this area. Furthermore, given our expertise and interest in research in methodology for solution of Markov chain models [6], this is a good choice of applications. Lastly, as described in Section 4, our existing collaborations with Prof. de Souza e Silva's group will also contribute to expanding the capabilities of this application (in the form of traditional solvers contributions) as well provide a user base for it (with Profs. Lui's and de Souza e Silva's research groups).

5. *Petri net modeling tool* — we believe that higher level tools (i.e., tools using higher level abstractions, such as petri nets, queueing networks, etc.) can be easily built on top of *βolshoi* (i.e., we can easily build front-ends for such applications); we will test this concept by building a Petri Net front-end for *βolshoi*. Given our existing collaborations (see Section 4) with Prof. Franceschinis' group we will have the needed input from experts and a solid user base for this purpose.

3.5 Past Experience

Our previous efforts in the area of performance evaluation tools include the Tangram [7, 13, 14] project, which was partly an effort to provide access to a multitude of existing analytic performance evaluation tools as well as simulation tools through a graphical user interface.

Although successful in its efforts, part of the lessons learned was a need for a simple yet powerful and general abstraction for model specification which was independent of the graphical UI used, the application being developed, the methodology used to solve the models, and so on. We believe that the "range of cells" abstraction is just such an abstraction.

Furthermore, the object-oriented technology has significantly matured since the Tangram project, and thus we will build a modern tool with modern technology, i.e., the component software technology.

Lastly, the past experience of designing and building a performance evaluation tool, the lessons learned, the feedback received from the users (such as the difficulty with task-graph-based model specification used in the past) will contribute to the success of *βolshoi*.

4 Collaborations: expanding the tool's usability range

The following major (USA and foreign) collaborators, as well as their students and the research groups they lead, will participate in this project:

1. Dr. Mark S. Squillante, IBM T.J. Watson Research Center, USA
2. Prof. Edmundo de Souza e Silva, Federal University of Rio de Janeiro, Brazil
3. Prof. John Chi Shing Lui, The Chinese University of Hong Kong, Hong Kong
4. Prof. Giuliana Franceschinis, Università del Piemonte, Orientale "Amedeo Avogadro", Italy

Letters of support from three of the above collaborators (Dr. Squillante, Prof. de Souza e Silva, and Prof. Lui) are included in this proposal.

We anticipate significant benefits from the collaboration with the above researchers and institutions. These benefits are as follows.

Dr. Squillante's group does extensive work on modeling and performance evaluation of computer systems, including extensive studies of web servers. Furthermore, they have expertise in a variety of modeling methodologies, for instance, Dr. Squillante is an expert on matrix-analytic techniques (as stated in his letter). Thus, he and his group will be provide us with user input, for *βolshoi*, in general, as well as specifically for the capacity planning tool and Markov chain modeling tool applications (as described in Section 3.4). Having user input, as well as a user base, in the industry will be of significant benefit to our efforts.

Prof. de Souza e Silva and Prof. Franceschinis have for many years been involved in tool development. Prof. de Souza e Silva is an expert on Markov chains, queueing networks, and reliability models as well as on tools related to these methodologies. Prof. Franceschinis is an expert on Petri Nets as well as tools related to this methodology (she is a member of the core team that developed the widely used GreatSPN tool). Prof. Lui is an expert on methodology for Markov chain models and is interested in customizing our tool for Internet and other network related applications. Thus, these collaborators will contribute to expanding the range of solution methodologies and applications (refer to Section 3.4) that will be provided by our tool. These contributions and support will be done in the form of (1) the respective researcher's time, (2) their graduate students' time, and (3) contribution of their existing source code (this includes TANGRAM-II and GreatSPN). This is also stated in the support letters.

Furthermore, from an educational point of view, they will serve as other educational sites (in addition to University of Maryland) that will use our tool in their courses and provide feedback on its improvement for educational purposes.

Thus these USA and foreign collaborations will bring benefits through expanding the range of methodologies and applications that can be included in the tool, i.e., in addition to the expertise of the PI's, as well as through expanding the user base, for research and educational purposes.

5 Educational Benefits

System performance evaluation should be an important part of an educational experience of both undergraduate and graduate students in Computer Science and Computer Engineering who have an interest in systems related topics, such as operating systems, communication networks, database systems, and so on. Even students that do not go on to become "performance specialists" can greatly benefit from such an education, as "good performance" is a goal in most computer and communication systems built today.

Performance evaluation and prediction of today's complex computer systems is no easy task. The analytical methodologies required to do that are complex. Even if only simulation is used, proper simulation methodology and result interpretation is also critical to the success of the performance studies.

One of the best ways to educate students about performance evaluation is to have them do performance-oriented design and studies of realistic systems, i.e., in the form of class projects. Having a tool that is able to handle evaluation of realistic systems and that will allow students to (a) have a hands-on experience with existing evaluation techniques and (b) customize or add their own techniques and applications, will be invaluable to their educational experience and have a significant effect on their future work and careers.

Thus, our educational goal in this project is to use *βolshoi* in performance evaluation courses at the University of Maryland (as well as at foreign institutions as described above), so that the students can use it to (1) learn about performance evaluation methods, (2) add new techniques to it, which greatly contributes to their understanding of the performance field, and (3) customize it for new applications, which greatly contributes to their understanding of systems that are of interest to them.

6 Previous Work

In this section we briefly discuss existing analytical tools using the classification given in [16]. For a listing of tools and details on each tool category refer to an excellent tutorial in [16]. (Note that, a variety of simulation and measurement tools also exist, but are outside the scope of this proposal — as stated previously, our interests here are mostly in an *interactive* (and therefore analytic) performance evaluation process.)

- *Single Formalism Tools* — these tools provide a single model specification abstraction, often with a variety of solution techniques. These include, queueing networks (product-form and non-product-form), matrix-analytic, stochastic petri nets and extensions, stochastic process algebras, and a variety of application-tailored tools (but also based on a single formalism).
- *Combination of Multiple Tools in a Single Software Environment* — these tools are user-interface-driven, i.e., they integrate multiple specification, analysis, and presentation tools through a common graphical user interface.
- *Integrated Modeling Framework* — these tools integrate multiple modeling formalisms and solution techniques and are motivated by the belief that no single formalism is best for representation of all parts of a complex computer or communication system as well as that no single solution technique is appropriate and/or efficient for all models. These include recent work on the Mobius project [4].

To the best of our knowledge, none of these tools deal with the issues of infinite models in a general manner; specifically, except for tools that include open queueing networks and matrix-analytic techniques, none deal with infinite models at all. The open queueing networks and matrix-analytic tools are based on a specific single formalism, and it is not at all clear how to carry over their approaches (to infinite models) to other formalisms.

Furthermore, we believe that all of the above tools suffer from the problem that they are intended for expert users only. As stated earlier, we also believe that the spreadsheet abstraction is the most natural one available to date and the one suited for experts and non-experts alike. None of the tools that fall into categories given above are based on the spreadsheet abstraction.

Thus, we believe that, if successful, our efforts in this proposal will have a significant impact on future performance tool designs as well as make significant strides in wide-spread use of performance evaluation techniques among computer and communication system designers.

References

[1] F. Baskett, K. M. Chandy, R. R. Muntz, and F. Palacios. Open, closed and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, April 1975.

[2] A. E. Conway and N. D. Georganas. *Queueing Networks - Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation*. MIT Press, 1989.

[3] P. J. Courtois and P. Semal. Bounds for the positive eigenvectors of non-negative matrices and for their approximations. *Journal of the ACM*, 31(4):804–825, October 1984.

[4] D. Daly, D. D. Deavours, J. M. Doyle, A. J. Stillman, P. G. Webster, and W. H. Sanders. Mobius: An extensible framework for performance and dependability modeling. In *Eighth International Workshop on Petri Nets and Performance Models*, September 1999.

[5] E. de Souza e Silva. Algorithms for queueing network analysis of distributed systems. *UCLA Technical Report no. 840040*, 1984.

[6] L. Golubchik and J. C.S. Lui. Bounding of performance measures for a threshold-based queueing system with hysteresis. In *Proceedings of 1997 ACM SIGMETRICS Conf.*, Seattle, WA, June 1997.

[7] L. Golubchik, G. D. Rozenblat, W. C. Cheng, and R. R. Muntz. The Tangram modeling environment. In *Fifth International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, pages 421–435, Turin, Italy, February 1991.

[8] N. J. Gunther. *The Practical Performance Analyst*. McGraw-Hill, 1998.

[9] D.I. Kang, R. Gerber, L. Golubchik, and J. K. Hollingsworth. Techniques for automating distributed real-time applications design. In *High Performance Distributed Computing Conference*, Redondo Beach, CA, August 1999.

[10] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA-SIAM, 1999.

[11] I. Matta and A. U. Shankar. Z-Iteration Efficient Estimation of Instantaneous Measures in Time-Dependent Multi-Class Systems. In *Proceedings of 1995 ACM SIGMETRICS Conf.*, Ottawa, Canada, May 1995.

[12] M. F. Neuts. *Matrix-geometric Solutions in Stochastic Models – an Algorithmic Approach*. John Hopkins University Press, Baltimore, MD, 1960.

[13] T. Page, S. Berson, W. Cheng, and R. Muntz. An object oriented modeling environment. In *Proc. of the OOPSLA '89 Conf.*, pages 287–296, New Orleans, October 1989.

[14] G. D. Rozenblat, , and R. R. Muntz. The Tangram Simulation Animation System. In *Second Eurographics Workshop on Animation and Simulation*, Vienna, Austria, September 1991.

[15] E. de Souza e Silva S. E. Berson and R. R. Muntz. An object oriented methodology for the specification of Markov models. Technical report, UCLA Computer Science Department, June 1987 (revised February 1988).

[16] W. H. Sanders. Integrated frameworks for multi-level and multi-formalism modeling. In *Workshop on Modeling of Heterogeneous Networks*, October 1999.

[17] H. A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29:111–138, 1961.

[18] C. U. Smith. *Performance Engineering of Software Systems.* Addison-Wesley, Boston, 1991.

[19] W. J. Stewart. *Introduction to Numerical Solution of Markov Chains.* Princeton University Press, 1994.

[20] C. White. Dmv hits pothole. *BYTE*, 1996.