



Bell Laboratories

subject: **Providing System Analysis
Tools over the Internet
Work Project Nos.
311405-5607**

date: **February 10, 1997**
from: **Aad van Moorsel
Dept. BL011256
MH 2B-121
908-582-5183
aad@research.bell-labs.com
BL011256-970801-TM**

TECHNICAL MEMORANDUM

1 Introduction

A continuously and rapidly growing number of services is becoming available through the Internet. Examples are banking services, web servers that determine shortest travel routes, commercial services for ordering books CDs or other products, etc. Potential clients/users of these services only need to have access to an Internet-connected machine, with a web browser.

The mentioned services are based on relatively light-weight software solutions. However, more generic software applications are being thought of, for instance for Intranet-wide shared administrative tools. Furthermore, with respect to the global Internet, ideas about renting instead of buying and installing software are being ventilated. Realizing this seemingly irreversible trend towards Internet/Intranet applications, we discuss in this report how to offer system analysis software, such as tools for performance and reliability evaluation, over the web. Offering such services over the web is important for the tool developer, since a successful integration of analysis tools in future user environments is very important for the continuing acceptance of such tools.

Traditionally, system analysis software has been provided as stand-alone applications. Stand-alone applications need to be down loaded (or otherwise shipped) and installed, and the individual user environment variables have to be properly set. Moreover, the tool often only works on particular computer architectures, and for every supported architecture the tool developer has to do the porting. For every new version of the software the procedure of installing has to be repeated, and software updates and bug fixes can often not be distributed until a new release. Issues such as these have a severe impact on whether software is actually being used to its fullest potential. An Internet-based approach to providing software tools can help alleviate these problems, as we will illustrate in this paper.

Performance evaluation tools enjoy some particular characteristics important in their design as Internet service. The user interface is usually (and preferably) dominantly graphical, allowing for intuitive and high-level model building by system engineers. Typical solution algorithms, either of analytical or simulative nature, are potentially computationally very expensive. For analytic solutions, substantial disk space is often important as well (for instance for state space generation). In other words, while the user interface puts minor requirements on CPU and I/O, the solvers typically require high-end equipment (fast CPU and sufficient disk space), or even special equipment such as parallel machines.

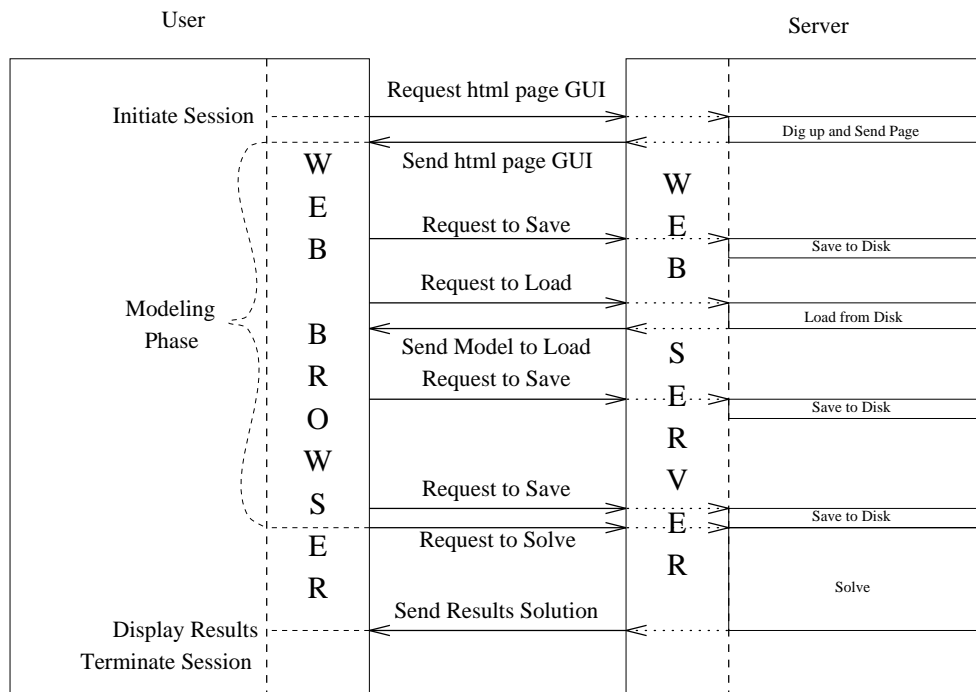
The solution we propose only uses standard and readily available Internet technology. We base our design on the premise that most computer users nowadays have access to the Internet and know how to use a Java-enabled web browser. No further requirements will be posed on the user and user equipment. The tool developer programs the graphical user interface (GUI) as a Java applet, and installs the tool on the local web server. Computationally light weight functionality is part of the Java applet, but computationally intensive parts, such as model solution algorithms, are run on the server side for performance reasons. Also, model saving and loading is done on the server side, since most web browsers restrict the access to local file systems. As a consequence, the server must provide appropriate disk space and CPU time; the client machine, however, only needs web-browser functionality.

Providing analysis tools on the web has advantages for the user since accessing the tool is much simplified and direct, and has advantages for the tool builder, since tool maintenance and upgrading is kept local. Furthermore, since solution algorithms are being run on the server side, all users will experience improved performance if high-end or special-purpose equipment is used on the server side. However, the proposed solution should not be considered a panacea. It is the logical first step in investigating Internet-based solutions, since it puts minimal requirements on the user and provider. Depending on the specific user and user equipment, as well as on the particular tool, enhancements of this approach are, however, in order, and can be worked out based on the provided basic solution.

2 Design Choices

In the design of Internet-based performance and reliability modeling tools, there are at least three basic aspects to consider: GUI, saving/loading of models, and solving models. Our decisions regarding these elements have as underlying goal to simplify user access and use of tools, and to simplify the development, release and maintenance of the software. Additionally, improvements in these elements should not be at the cost of the tool's performance, reliability and other characteristics.

In Figure 1 the proposed design is depicted. The figure is intended to be read from top to bottom, ordering the subsequent actions initiated by the user. The user interacts solely via a web browser, and the server runs web server software. All communication takes place using the HTTP protocol, delivering web pages to the user: first to download the GUI, then for saving and loading models, then to solve the model, and finally to display the results. The activities the client performs are denoted on the left side of the user box, and the boxes on the server side represent the CPU or I/O activities of the server.



HTTP protocol

Figure 1: Design Choices–Potential Actions of Users Ordered from Top to Bottom.

With respect to the GUI, we take advantage of the opportunities offered by any competitive web browser. In particular, we implement the GUI as a Java applet, which can be accessed through a web browser, and we use Common Gateway Interface (CGI) output in the form of HTML pages to display the results. The server site requires a CGI supporting web server and the analysis tool, but no additional system software.

Clearly, the opportunity to access and use the software in a web browser is a very important step towards improving the acceptance of a tool. It is, however, not practical to carry out all computation and file I/O within the applet. First, solvers are computationally intensive, and should not be implemented in Java for performance reasons. Secondly, the user's equipment might not be suitable for heavy-duty computations or for storing and handling large state-spaces generated during solution.

The answer to these problems is to allow to use CPU and disk storage on machines other than the client's. However, security provisions in web browsers restrict the possibilities to address this issue. In particular, in Netscape's and Microsoft's browsers, while the applet is running, files cannot be saved, and programs cannot be executed, on any other machine than the server. Therefore, we take the following approach for saving, loading and solving models: a new URL is opened in the Java applet, and CGI is used to start an executable to solve, save or load the model. Effectively, we 'step outside' the applet, and load a new HTML page as specified by the CGI output.

3 Implementation

In this section we discuss the consequences of the described design choices for the implementation. We first discuss the Java applet implementation, and then the use of CGI.

3.1 GUI as Java Applet

Figure 2 shows the user interface implemented as a Java applet, in this case running in the Netscape Navigator browser. The modeling formalism shown is a simplified form of a stochastic Petri net or stochastic activity network, that represents discrete-time Markov chains. The circles represent 'states,' and the bars represent 'transitions.' Every small semi-circle attached to a transition corresponds to a 'case.' The cases are assigned probabilities, and every arc connects a case with the state reached if that particular case is chosen. For every transition, the associated case probabilities sum to 1.

To keep track of the model that is being constructed, the applet stores the coordinates and other variables of the model. So, in the applet, we keep track of the position of the attributes (circles, bars and arcs), as well as their labels and the probabilities associated with the choices at a transition. In other words, enough information is being kept to reconstruct the graphical display of the model.

Attributes can be added and displayed in the drawing panel by point-and-clicking. In addition to the basic model construction functionality, the GUI provides:

- Editing functions: *delete*, *move*, *select* any of the attributes on the panel; *refresh* the screen; *display* the model's parameter values in the text panel on the right hand side.

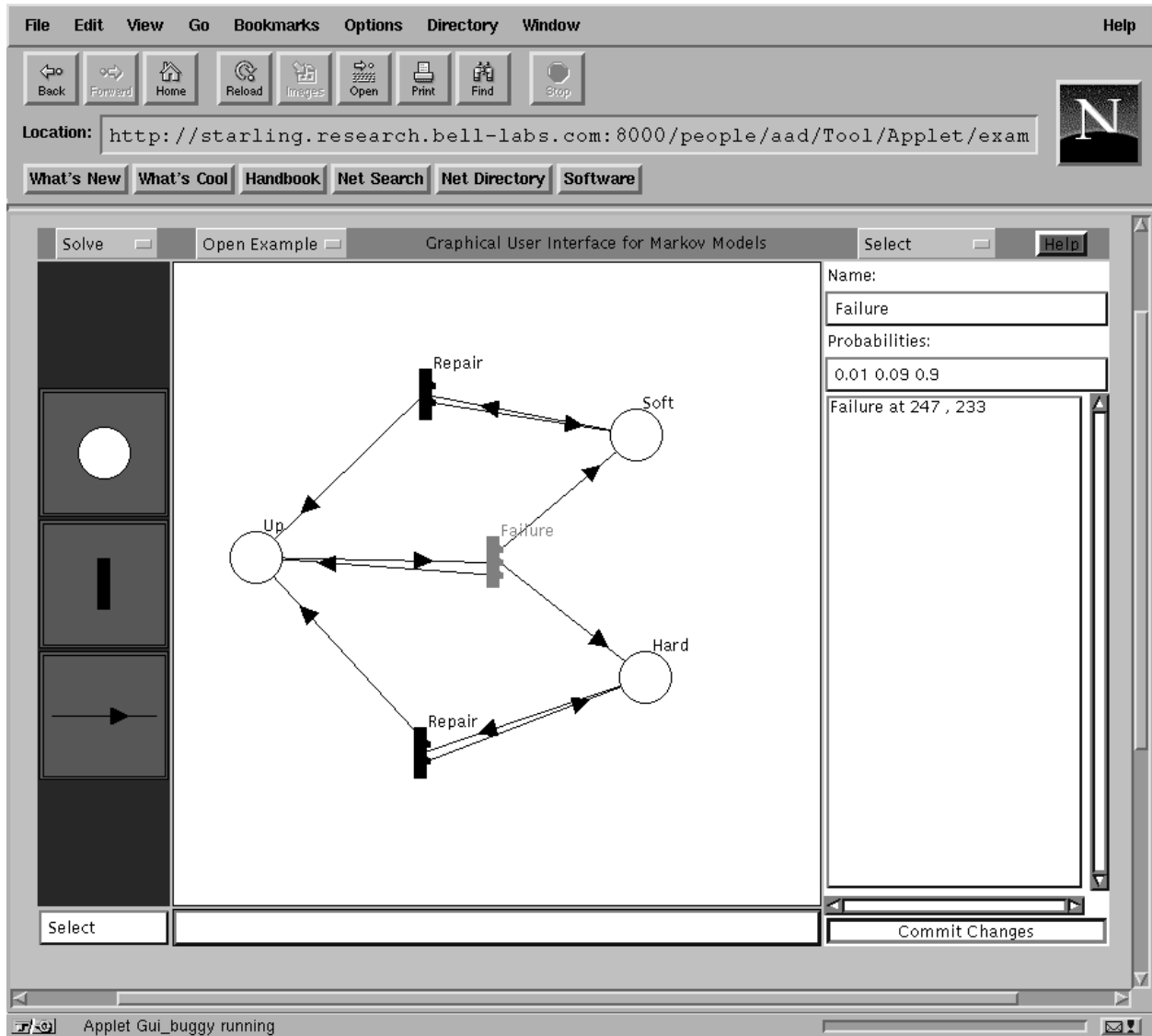


Figure 2: GUI.

- File operations: *save* and *open*.
- Computing operations: *solve*, *animate* and *sanity check*.
- Documentation: *help*.

All the above functions can be accessed using the pull-up menus in the menu bar at the top of the GUI. The *animate* option allows the user to follow possible evolutions of the model, which is important for model understanding and debugging. The *sanity check* does a structural test of the model to assure that it specifies a solvable Markov chain. *Animation* and *sanity check* are carried out within the applet, as are the editing commands, but *save*, *load* and *help* interact with the server. The *save* and *load* command start CGI programs, the *help* command loads HTML help pages.

Care has been taken in the design of this tool to allow for extensions or adaptations for other modeling formalisms. In particular, the individual attributes extend the abstract class ‘Attribute.’ This implies that states, transitions and arcs are instances of a sub-class of Attribute, all implementing at least the methods declared in the Attribute parent class. Other classes can access the instances as an Attribute, that is, without having to identify whether it concerns a place, transition or arc. The main portion of the program can therefore be reused with only minor adaptations if other attributes (like gates in stochastic activity networks) are added or if other formalisms (like queuing networks) are considered.

In addition to proper design, the extensibility and future improvement of the tool can also benefit greatly from the modularity of Java, in combination with the popularity of the language. Other programmer’s interfaces can be plugged in if they look nicer or provide better functionality, and, given the habit of releasing source code on the web, it can be expected that for GUIs plenty of publicly available Java classes will be developed (in fact, such classes have already been developed).

3.2 CGI for Saving, Loading and Solving

For saving and opening models, as well as for solving models, the design uses CGI. Using this interface, executables can be started on a web server site, resulting in a new HTML page in the user’s browser.

In the Java implementation, we use the URL class to open an HTTP connection to the desired CGI program. The CGI program is accessed using the “get” mechanism, that is, a string that represents the model is sent as part of the URL address.

So, when selecting *save*, the applet creates a string representing the model, opens the URL, and executes on the server side the C program that saves the string to a file. Note that the URL connection can be opened to any machine. After saving the model, CGI output provides a new page, reporting success of the save request. To continue using the applet, the browser has to return to the previous web page.

Using CGI implies that in case one selects the *load* option, the new model is not loaded ‘in’ the running applet. Instead, a new page is being opened as result of the CGI action, and the loaded model is opened in a ‘fresh’ (but further identical) applet. Typically, the

applet's class files are still available in the cache, so there is no need to re-load the applet from the server. The loaded model is specified as a ASCII string in a `<param>` field of the HTML page output by CGI, which is read by the applet. We note that the tool also provides the possibility to load an example, which is also specified in a `<param>` field of an HTML page.

4 Discussion and Conclusion

This report discussed the use of standard Internet technology to provide system analysis tools to system designers. The major goals are to simplify access to the tool for the user, to simplify releasing and maintaining tools for the developer, and to sustain or improve the performance and other quality characteristics of the tool.

The proposed solution completely fits in the world-wide-web paradigm. That is, there is a web browser on the user side, a web server on the server side, and all communication is carried out by means of HTTP. This approach should not be considered a panacea, but is a logical first step in applying Internet solutions.

Different user environments and different tools will put different demands on a possible Internet/Intranet solution. For instance, communication over the world-wide Internet cannot be expected to be as reliable as that over a company-wide Intranet, "heavy" users will have more strenuous demands than occasional users on the communication, CPU, I/O, etc.

To adapt to the different environments in which tools are being used, changes to the proposed solution will be in order. Adaptations can be made by relaxing the intimate ties our approach has with the world-wide-web schema. In particular, the Java-implemented GUI can also be used as a stand-alone interface, that is, outside the web browser. This would simplify saving and loading user files on the client's side. On the server side, one can add server software and communicate via sockets, which is more flexible than using a CGI-based approach since one can stay 'in' the running applet and does not open a new URL.

The Markov chain modeling tool described in this report has been shown to work very conveniently for Markov models with up to hundreds of states, using the tool in an Intranet environment. Further experiments must be carried out to fully assess the potential of the proposed Internet approach. The desire to use tools over the Internet/Intranet, however, seems to be continuously growing among potential users, and further exploration of the possibilities offered by current and future technology is therefore in order.

Acknowledgments

Thanks to Emerald Chung, Chandra Kintala, Reinhard Klemm, Navjot Singh and Tim Tsai for suggestions, discussions and coding expertise.

References

Plenty of reference material exists on the topics discussed in this paper. I do not include a detailed bibliography, but you can of course contact me for material I have used. The described Markov modeling tool can be accessed within Lucent Technologies. Mail me (aad@research.bell-labs.com) for the up-to-date URL, or access my web page at <http://starling.research.bell-labs.com:8000/people/aad>. For more information about performance and reliability evaluation tools, consult *W. H. Sanders, W. D. Obal II, M. A. Qureshi and F. K. Widjanarko, "The UltraSAN Modeling Environment," Performance Evaluation, pp. 89–115, Vol. 24, 1995*, and references therein.

Copy to
DH 1125
MTS 11256
W. H. Ninke
R. Sethi
S.C. Borst
G.R. Bruns
S.G. Eick
P. Godefroid
K. Kumaran
V.B. Mendiratta
D. Mitra
M.A. Qureshi
B. Sugla



Title: Providing System Analysis Tools over the Internet

Author	Electronic Address	Location	Phone	Company (if other than Lucent-BL)
Aad van Moorsel	aad@research.bell-labs.com	MH 2B-121	908-582-5183	

Document Nos.	Filing Case No.	Work Project Nos.
BL011256-970801-TM		311405-5607

Keywords:

Internet, Intranet, Java, CGI, HTML, performance and reliability evaluation, system analysis tools, Markov models

MERCURY Announcement Bulletin Sections

CMP- Computing

Abstract

With the advance of technology to offer services over the Internet/Intranet, new opportunities arise to provide system analysis tools over the Internet as well. In this paper we discuss an approach in which the developer makes a design tool available for public (or company-wide) access by installing it on a web server, and the user accesses and uses the software within the web browser with which he or she is most familiar, without having to actually download the tool. We will illustrate and discuss the design and implementation of such a tool for constructing and solving Markovian performance and reliability models. The tool's graphical user interface is implemented as a Java applet, accessible with a web browser, and the tool's solvers are supplied using CGI programs, executed by the web server. This approach is easily realizable in many environments and for many tools, since users are not required to have any equipment other than an Internet-connected 'web terminal,' while the server only needs standard web server software. Adaptations of this approach can be developed to fit specific tool or user requirements.

Pages of Text 7 Other Pages 2 Total 9
No. Figs. 2 No. Tables 0 No. Refs. 0

Mailing Label

Lucent Technologies – PROPRIETARY
Use pursuant to Company Instructions

Complete Copy

Cover Sheet Only

DH 1125
 MTS 11256
 W. H. Ninke
 R. Sethi
 S.C. Borst
 G.R. Bruns
 S.G. Eick
 P. Godefroid
 K. Kumaran
 V.B. Mendiratta
 D. Mitra
 M.A. Qureshi
 B. Sugla

MTS 11254
 MTS 11257
 MTS 11258
 MTS 11259
 A. Netravali
 E. J. Rosenthal

**Future Lucent Technologies
 Distribution by ITDS**

Release to any Lucent Technologies employee
 (excluding contract employees)

Author Signature

Aad van Moorsel

Organizational Approvals

Chandra M. Kintala

William H. Ninke

For Use by Recipient of Cover Sheet:

Computing network users may order copies via the *library -1* command;
 for information, type *man library* after the UNIX[®] system prompt.

Internal Technical Document Service

Otherwise:

Enter PAN if Lucent-BL (or SS# if non-Lucent-BL). _____
 Return this sheet to any ITDS location.

() AK 2H-28 () IH 7M-103 () DR 2F-19 () NW-ITDS
 () ALC 1B-102 () MV 3L-19 () INH 1C-114 () PR 5-2120
 () CB 30-2011 () WH 5E-233 () IW 2Z-156
 () HO 4F-112 () MT 3B-117