# A MODEL FOR PERFORMANCE EVALUATION OF INTERACTIVE SYSTEMS

Edward A. Stohr
Department of Information Systems
Leonard N. Stern School of Business
New York University

Yongbeom Kim
Department of Information Systems and Sciences
College of Business Administration
Farleigh Dickinson University

August 13, 1997

Working Paper Series
Stern #IS-97-16

# A Model For Performance Evaluation Of Interactive Systems


## Abstract

We describe a quantitative model for the performance evaluation of interactive computer systems. The approach involves the development of an "interaction graph" or state transition diagram to describe the user-machine interaction. Given numerical data on transition times and probabilities, the model can be used to perform sensitivity analyses of changes in system parameters and user behavior. To illustrate the model, we use empirical data from field and laboratory experiments designed to compare a prototype natural language query system with a formal (relational) query system. The general approach is applicable in a broad range of other contexts including bibliographic retrieval and the analysis of web-log data. It should be of interest to both system developers and potential users of these systems.


## 1. Introduction

This paper presents a methodology for evaluating interactive computing systems along two dimensions: what proportion of the users' tasks are successfully performed? and how efficiently can the users perform their tasks? The methodology uses test results that might be obtained by web masters from World Wide Web (WWW) activity log files, by system developers from Beta tests or by prospective users from their own evaluation tests. A graph-based model is then constructed to help evaluate the test results and to perform sensitivity analyses to determine the results of changes in either the system or user performance measures. The model can help system designers learn which improvements to the system will most increase its utility to users. This will help them allocate development efforts to the most significant areas for improvement. On the other side of the fence, prospective users can obtain objective measures for comparing alternative systems that they might purchase.

The methodology can be used to evaluate any interactive system which involves repetitive operations and where the results of the user interaction can either end in success or failure. Many systems fit this description. Examples include WWW search engines, bibliographic retrieval systems and workflow systems involving rework. The criteria generally used to evaluate such systems include bibliographic measures such as recall and precision ratios [Chu and Rosenthal 1996], response time, percentage of queries successfully answered, and so on. Here we use a mathematical model to develop an additional criterion that involves both user time and accuracy: namely, the average number of correct queries per unit time.

Section 2 of the paper provides two examples of interactive systems that can be analyzed using our methodology. Section 3 discusses criteria for evaluating the effectiveness of interactive computing systems in general. Section 4 develops a Markov model of the interaction process that simplifies the computation of the criteria and can be applied in many situations. Section 5 uses the model to carry-out sensitivity analyses on data from two database retrieval systems . Finally, Section 6 presents some conclusions and suggestions for future research.

## 2. Illustrative Systems

### 2.1 WWW Search Engines (Yahoo!)

With the tremendous growth of the WWW, the ability to search for information has become a major component of user value in educational applications and one of the cornerstones of electronic commerce [Kambil 97]. Search engines on the world wide web process millions of searches daily. Obviously, their effectiveness and efficiency is a matter of major concern and interest.

To attack this problem, we first construct a directed graph ("interaction diagram") in which each node represents a user or system state and the branches represent probabilistically determined transitions to other states. The states are system or user decision or "branch" points. For user branch points, the state will usually be associated with a particular screen display (or a series of similar screens with different information content.) The transitions represent the actions that are taken and, in general, consume

some random amount of time. The idea is best illustrated with a familiar example. Figure 1 shows an interaction diagram for the Yahoo search engine. The first node is associated with the Yahoo home page and the decision involves the type of search (action) to undertake - use a search string (keyword search in the figure), search within Yahoo's predefined hierarchy of subject categories, visit the "cool" sites or search by specific locations, etc. Assuming the user decides to use keyword search, the decision and transit time is $T_{12}$ and we move to the next transition point (Node 2) at which the user decides on the keywords to be used and enters them in Yahoo's dialogue box. This consumes time $T_{23}$ after which we move to a system transition point (Node 3) where the system starts to retrieve the URLs it deems relevant (good matches) to the query. The associated transactions are "Stop/Halt" if the system fails or the user aborts the process because it takes too long or he/she wishes to reenter the query, "No Matches" if the search reveals no matches, and "Match" if there is at least one matching URL. In the last case, we move to Node 4 where the user begins to analyze the retrieved list of URLs (by default, Yahoo presents these in groups of 20.) The transitions from Node 4 include "Click URL", "Retrieve Next 20", "New Query" and "Give-up." If the first transition is chosen, we move to Node 5 where the user starts to analyze the information content at the foreign (to Yahoo) URL. The transitions from Node 5 are "Select Back-arrow", "Success", and "Give-up." In the first case, we move back to Node 4 and re-examine the list of retrieved URLs. In the latter two cases, we transition to sink nodes.

There are a number of modeling issues involved in constructing an interaction diagram. The first concerns the granularity of the diagram. For example, we chose to use separate nodes for the cases where there were zero matched URLs (Node 3) and at least one matched URL (Node 4.). This is because the branches (actions) that are possible in these two cases are different. However, we (arbitrarily) chose not to distinguish between the cases when Yahoo finds fewer than 20 URLs as opposed to cases where more than 20 URLs were found. Also, because we are interested in modeling the interactions within the Yahoo search engine, we represented the (possibly very complex) set of user interactions at Node 5 by a single node. Obviously, the transition times to Nodes 4, 7 and 8 from this node will have a high variance. The granularity of the representation is a matter of

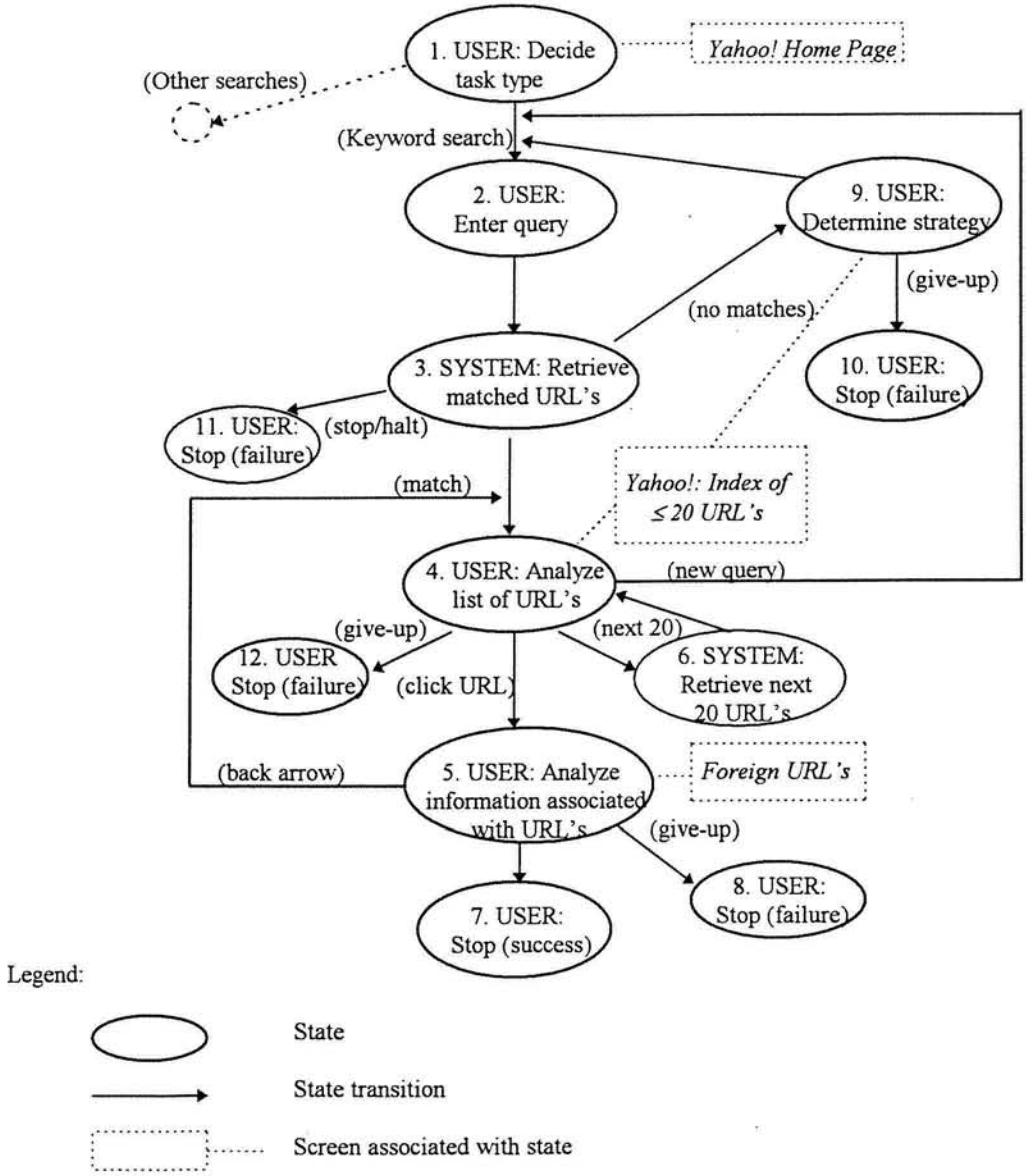


Figure 1. Interaction Model of Keyword Search in Yahoo!

judgment that depends in part on the objectives of the analysis but might also be dictated by the available data. In the Yahoo case, it will be possible (with one exception) to analyze the web log statistics over a period of time (say one week) to determine the frequencies and times of all transitions in the figure classified by user class and perhaps also by general

topic category. The one exception is that it would not be possible to distinguish between transitions to Nodes 7 and 8 without explicit feedback from the users. For the same reason, while we might like to do so, it will not be possible to gather data on what goes on at other URLs (Node 5) in the figure.

The user interaction with other search engines such as Excite, Lycos and Alta Vista can be modeled in a similar fashion. Note that these search engines employ very different strategies for searching, classifying and indexing the vast amount of information available on the web and perform differently with respect to different classes of query [Courtois et al. 1995, Leighton 1995, Gray 1996]. For example, [Chu and Rosenthal 1996] evaluates Excite, Lycos and Alta Vista in terms of two criteria, response time for the first 10 items retrieved and "Precision" defined as the percentage of the first 10 retrieved items that are deemed relevant to the query. (Note that precision is usually defined relative to the total number of items retrieved - see later.) While the authors found little difference in response times between the systems, the average precision ratios between the systems differed by as much as 50%. By diagramming the user interaction and collecting data on the frequencies and times of execution of the state transitions in graphs such as Figure 1, we aim to perform a comparative analysis of the various search engines. We will compare the search engines' performance on different tasks in terms of a number of criteria such as probability of a successful query and average time to success (or failure) as well as more traditional information retrieval criteria such as recall and precision. The graph approach allows a component by component analysis of performance to be carried-out. For example, what is the impact of a change in the time to perform a system or user step in the figure? We expect the user steps to be greatly influenced by the volume of data retrieved as well as by its presentation format. How do changes in the amount of information presented to users impact performance? Similarly, we can investigate the impact of changing the probabilities of the various state transitions.

We can also model user behavior. What choices do users make at each point in the interaction? How is this impacted by the type of task they are performing? How does user behavior vary over time? How often do they change search strategies? How quickly

do they give up if they are not successful? A better idea of actual user behavior will help inform and guide the system analysis just outlined.

### 2.2 Database Retrieval Systems

The laboratory and field studies [Jarke et al. 1985a] that form the background for our second illustration, were aimed at evaluating a natural language system (NLS) for querying databases. The NLS system contained a base grammar consisting of some 800 language rules and a lexicon of approximately 150 common words such as 'to be', and to 'to have'. An important feature of the system was that English natural language queries were translated into a formal data base query language, SQL, prior to execution. The coexistence of the NLS and the formal language system (FLS) led naturally to an experimental design in which the formal language was used as a standard of comparison for the NLS. Linked laboratory and field studies employing student subjects were used to perform the evaluations [Jarke et al. 1985b]. The field experiment provided the data for the example used in this paper. The experimental application involved a database containing demographic and donation information for about 25,000 donors to NYU's Graduate School of Business Administration. Information retrieval requests from administrative officers of the school were given to six student "advisors" hired for the experiment. A typical session lasted anywhere from 30 minutes to two hours. During this time, the advisors analyzed the overall request, broke the request down into a number of tasks and entered *queries* using one of the information retrieval languages in an attempt to solve each task.

Figure 2 shows the interaction diagram for the NLS and/or the FLS database retrieval systems. Note that the same diagram is used to portray both systems. The major loop in the graph represents multiple attempts by the user to satisfy a task by submitting different formulations that end in either success or failure. The path between states 2 and 8, for example, means that the query was aborted during typing (either because of system noise or because the user backed-out of the query after recognition of a false start.) Path 3-8 shows that the query submitted by the user did not pass the system check for the FLS or the syntactic and semantic analysis made by the NLS. Other measurement variables (not

described in this paper) gave a detailed break-down of the types of error found and the reasons for these errors. In particular, typing errors were more prevalent in the FLS as discussed below. After parsing, and prior to execution of a retrieval, the FLS/NLS displays an estimate of the time it will take to execute the retrieval and allows users to

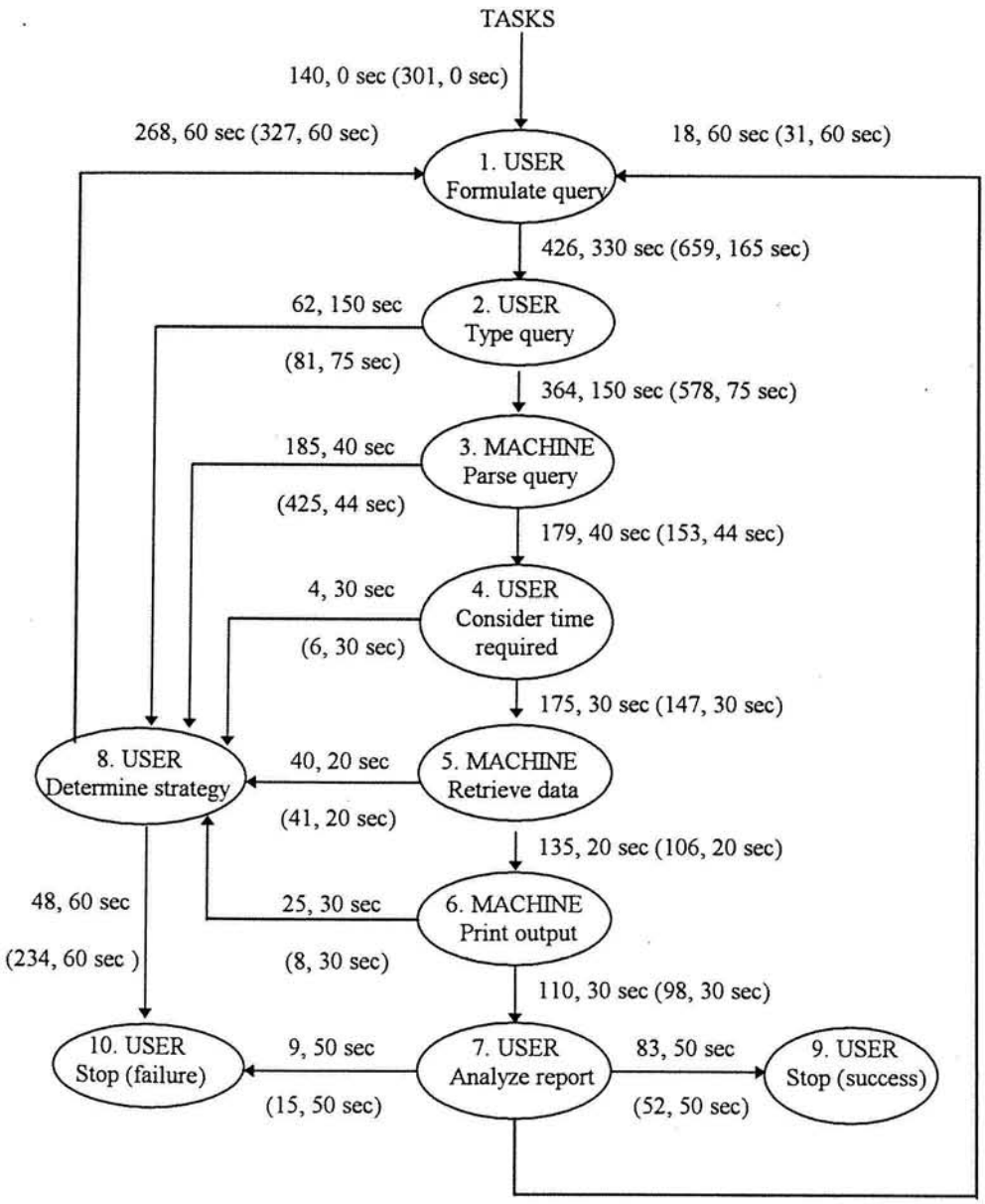


Figure 2. Query Processing - FLS and (NLS)

decide whether or not to continue. Path 4-8 represents a choice by the user to abandon the query. Paths 5-8 and 6-8 show queries that aborted due to either system failure or to user action during the processes of data retrieval, or printing the reports. The percentage of times these paths were taken is not high. However, due to poor system capability and slow equipment, the system was often resident in states 5 and 6 for considerable periods of time. In state 7, the user evaluates the results of the sub-query. If the query is considered successful, the process ends in state 9 (the task is completed.) Otherwise, the system moves to state 10 (the user gives up), or to state 1 to try another formulation of the query. In state 8, the user decides whether to give-up (path 8-10) or to try another query (path 8-1).

The estimated transit times and the frequency with which each path was used for the total FLS (NLS) experiment are shown in the figure. The success or failure of each interaction in the experiment was coded by a panel of judges. Table 1 summarizes the overall results (see [Jarke et al. 1985b]).

| | **FLS** | **NLS** |
|---|---|---|
| Number of Tasks | 140 | 301 |
| Number of Queries | 426 | 659 |
| Average Number of Queries Per Task | 3.0 | 2.2 |
| Average Number of Words Per Query | 34.19 | 10.64 |
| Average Time Per Task (minutes) | 31.1 | 13.9 |
| Average Time Per Query (minutes) | 10.8 | 6.9 |
| Average Success Rate of Tasks | 59.3% | 17.3% |
| Average Success Rate of Queries | 19.5% | 7.9% |

Table 1. Descriptive Statistics from Experiment

The most striking aspect of this data is the poor performance of both languages at the query level. Other results, and our own laboratory experiments [Jarke et al. 1985a], had predicted an approximately 60% rate of successful queries for the FLS. Several

physical limitations of the environment affected the design and execution of the field experiment and can explain at least a part of this difference in performance. In particular, the users interacted with the system on slow, hard-copy terminals using a communication system that was unusually 'noisy.' Also, the NLS was a prototype - several language features were missing or faulty. The coding scheme developed for the experiments was designed to compensate for some of these deficiencies and to measure the effects of others. Nevertheless, the conclusions that can be objectively obtained from such an approach to comparative language evaluation are bound by the state of development of the languages, the quality of the application design and the skills of the subjects. To illustrate, for the FLS and NLS systems, paths 5-8 and 6-8, and the proportion of path 2-8 due to typing errors as opposed to syntactic and semantic errors, capture system "noise." This represents a definite area for potential improvement in both language systems. What happens to overall system performance if the system "noise" is reduced by half?

A second striking aspect of the data in Table 1 is that the number of queries per task was lower for the NLS indicating less confidence on the part of users that their approach would succeed. They tended to "give-up" more easily than their FLS counterparts. On the basis of this observation and the much lower probability of success per query, one could conclude that the FLS is superior. However, there is one aspect of performance which favors the NLS - NLS users spent less time formulating and entering queries. In part, this is because the average length of queries measured in words was more than twice as long in the FLS as compared to the NLS. As the users were not expert typists and the interface characteristics were poor, typing was a major consumer of the subjects' time. Since, the idea of the NLS was to assist executive users, the time spent entering queries is very significant. Thus the percentage of correct queries is not the only criterion for judging the relative merit of the two languages. Perhaps one can be more efficient overall using the NLS because of the shorter time to formulate and enter queries. It would therefore be interesting to measure the impact of improvements in the probability of success on the relative efficiency of the two systems in terms of the average number of successful queries achievable per unit time. In particular, the ability of the NLS to parse natural language queries was quite poor. It is therefore natural to ask how much the

parsing capability (captured by $P_{34}$) would need to be improved so that the NLS would perform at the level as the FLS.

The model developed in this paper uses experimental data for calibration but attempts to free us from the constraints imposed by the particular circumstances of the experiment. Specifically the model provides a means for sensitivity analysis concerning: the effects of system 'noise' and the potential benefits to be obtained from various language improvements. In essence, we are proposing the development of a mathematical model that uses experimental data to calibrate its parameters but then allows sensitivity analysis to examine a number of "what-if?" issues. We will return to an analysis of the FLS and NLS with regard to the questions raised in this section after the development of the analytic model in sections 3 and 4.

## 3. Criteria for Comparative Evaluation of Interactive Computer Systems

Bibliographic systems are often evaluated in terms of two metrics - the recall ratio and the precision ratio. The recall ratio, r, is the ratio of number of relevant information items retrieved divided by the total number of relevant items in the information base. Information is relevant if it is judged useful by the user who initiated the search [Blair and Maron 1985]. Perfect recall, r = 1, is an ideal that is hard to attain in large complex systems such as library systems and almost impossible on the WWW. Another major problem is that recall is difficult to measure. While it is often possible to experimentally determine the number of relevant items that have been retrieved, estimation of the number of relevant items in the information base that were missed by the query may require a combination of extensive search, prior knowledge of an expert and/or sophisticated sampling. On the WWW, where there are millions of information sites and search engines sometimes retrieve as many as 30,000 items for a single request, estimating the denominator and numerator of r can be virtually impossible. The precision ratio, p, is the ratio of the number of relevant information items retrieved divided by the total number of total number of items retrieved. The precision is also difficult to measure when a very large number of times is retrieved. If the search engine rank orders the results, the first n items retrieved can be used as the base for determining precision as in [Chu and Rosenthal 1996]. Recall and

precision tend to vary inversely in searching (high precision being associated with low recall and vice versa.) Different users will have different requirements regarding recall and precision. For example, a user who wants general information on a topic might not have a high recall requirement - some general information might suffice. However, a user who wants comprehensive information will need high recall. The precision ratio is closely related to the user effort expended in information analysis (step 5 of Figure 1). If there is low precision and many items are retrieved, users may give up before they find the relevant information.

In contrast to bibliographic and WWW applications where r, p and response time are the most extensively used system evaluation criteria, most comparative studies of languages (for example, [Jarke et al. 1985b, Welty and Stemple 1981]) have used the percentage of queries correctly answered as the principle measure of language utility.

In this paper, we advocate a more general criterion that can be applied to a broad range of interactive computing systems including those discussed above. When choosing an interactive computer system one must weigh the benefits from the better, more timely, decisions that might be made as a result of using the system, against the costs involved in acquiring the system, developing the application, training the users and finally, the value of the time spent by users in formulating and entering queries and waiting for and analyzing results. The query results translate into organization profitability through the value (in improved decision-making) of executing each successful query or the opportunity cost of failing to obtain needed information from the system. While we do not attempt to measure the value of information, we advocate an efficiency measure - average time taken to execute a successful query - which takes into account both the probability of success and, indirectly, the value of user time. As noted above, the probability of a successful query is the usual criterion for evaluating query languages. As shown by Table 1, there is a lower probability of success per query attempt in the NLS. However, empirical evidence (see Table 1) shows that the queries are shorter and can be submitted in less time using the NLS as opposed to the FLS. Hence, the costs of user time per successful query may be lower with an NLS than an FLS if users can get the NLS to understand what they want in few enough iterations. This is the motivation for considering costs of user time while executing queries.

Consider a process such as that shown in Figure 2 where users repeat different formulations of a query until they get it right. Let $P_n$ be the probability that a user will succeed to retrieve the target information for the first time at the nth trial, then assuming a stationary Markovian process, the limiting probability of success is given by:

$$P_* = \sum_{n=1}^{\infty} P_n \qquad (1)$$

Note that $P_*$ gives an upper bound on performance. In reality, users tend not to repeat queries endlessly, hence, the probability of success ($P_s$) lies somewhere between $P_1$ and $P_*$. Let S(t) be the number of successful queries in t units of time, and $\mu_*$ be the expected time between successful queries. Then, the long-run average number of successes per unit time is given by:

$$\lim_{t\to\infty} \frac{S(t)}{t} = \frac{1}{\mu_*} \qquad (2)$$

We believe that $\mu_*$ may be a more useful criterion for performance evaluation than $P_s$. However, a sufficiently high query success rate is required to obtain system acceptance in the first place.

## 4. A Model for Evaluating Interactive Computing Systems

### 4.1 Performance Evaluation Using Directed Graph Models

It is difficult to compute the probability of success ($P_s$) and the expected time to succeed ($\mu_s$), for an interaction such as that shown in Figure 1 or 2 because the directed graph may consist of many states and the relationships among states can be complicated. To simplify the problem we make two very strong assumptions that are only true in an average sense but can give the designer a good idea of the major relationships in the problem. First, we assume that the transitions between states take a fixed period of time to execute - or rather, that the transitions, which will in general be random variables, can be approximated by their average execution times. Second, we assume that the transitions are Markovian - i.e., that the probability of a transition from the current state to another neighboring state depends only on the current state and not on the history of prior states that the user has experienced. This is obviously an approximation as, in particular, the probability of the user taking the path 8-10 (giving up on the task) is likely to increase with the number of iterations performed to date.

The approach we take builds on prior research on signal flow graphs [Shinners 1972] and the estimation of program run time [Graham 1977]. The usual approach to analyze a graph model such as Figure 2 is to utilize transformation rules that reduce complicated directed graphs to successively simpler forms and eventually to a single or at least a very small number of arcs from which the performance criteria (Ps and $\mu_s$) can be obtained directly. However graph reduction techniques get very tedious and time consuming as the number of paths and feedback loops increase. The innovation in our approach [Kim 1997] is to develop generalized formulae that permit calculations of success probabilities and expected times almost by inspection.

We now briefly review the transformation rules for graph reduction. Following this, we state our generalized formulae and illustrate them using the data in Figure 2.

Complex state transition graphs can be reduced to simpler forms using three transformations: serial, parallel, and loop. The serial transformation eliminates the middle node in a series of three nodes connected by single arcs as shown in Figure 3. Let $P_{ij}$ represent the transition probability from node i to node j, and $T_{ij}$ represent the transition time from node i to node j. In the serial transformation, branches (i, j) and (j, k) and node j are replaced by a single equivalent branch (i, k). The transition probability from node i to node k becomes $P_{ij}P_{jk}$, and the transition time from node i to node k becomes $T_{ij} + T_{jk}$.

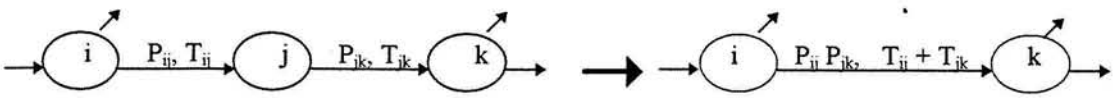


Figure 3. Serial Transformation

The parallel transformation is applicable to a directed graph when there are two branches both of which have the same origin node and the same terminal node as in Figure 4. The two branches between nodes i and j are reduced to a single equivalent branch in which the transition probability from node i to node j becomes $P_{ij} + P'_{ij}$, and the equivalent (fixed) transition time from node i to node j becomes $(P_{ij}T_{ij} + P'_{ij}T'_{ij})/(P_{ij} + P'_{ij})$. When there are more than two branches between two nodes, they can be reduced to a single equivalent branch by applying the parallel transformation repeatedly to two branches at a time.

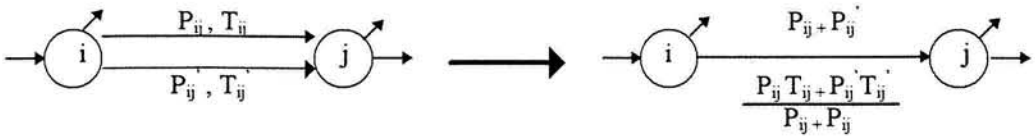


Figure 4. Parallel Transformation

The loop transformation is applicable to a directed graph when there is a loop at a single node that may be executed repeatedly according to some probability, $P_{ii}$, as in Figure 5. The transition probability from node i to node j becomes $P_{ij} / (1 - P_{ii})$, and the transition time from node i to node j becomes $T_{ij} + P_{ii} T_{ii} / (1 - P_{ii})$.

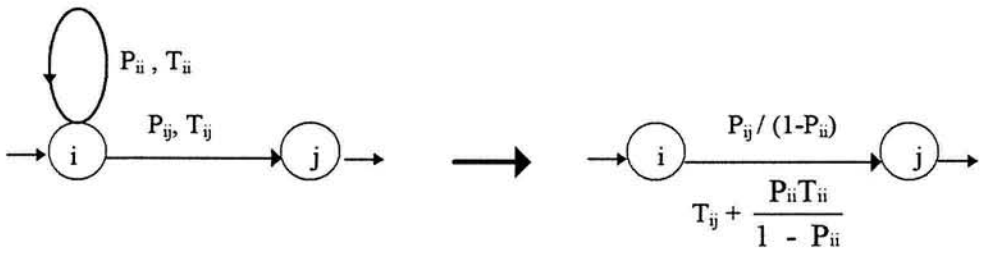


Figure 5. Loop Transformation

While it is possible to apply the above transformation rules to a complicated directed graph and to reduce it to one containing only a source node (which has only outgoing branches) and a sink node (which has only incoming branches), this process can be time-consuming as it requires repeated applications of the transformation rules until the final (simplest) graph is obtained.

To state generalized formulae that simplify the computation of the two criteria $P_s$ and $\mu_s$, we first define some terminology. Without loss of generality we can assume that the interaction graph has a single source and at least two sink nodes representing success and failure, respectively. A *forward path* is a path that originates from a source and terminates at a sink and along which no node is encountered more than once. *Touching loops* are loops that have at least one node in common. *Non-touching loops* are loops that have no node in common. The most general directed graph model has multiple forward paths containing multiple touching and non-touching loops. For example, in Figure 1, there are five forward paths (1-2-3-11, 1-2-3-9-10, 1-2-3-4-12, 1-2-3-4-5-7, and 1-2-3-4-5-8), three touching loops

(2-3-4-2, 4-6-4 and 4-5-4) and two pairs of non-touching loops (2-3-9-2 & 4-6-4 and 2-3-9-2 & 4-5-4.)

Figure 6 shows a somewhat simpler graph with both touching and non-touching loops that will be used to illustrate the definitions.

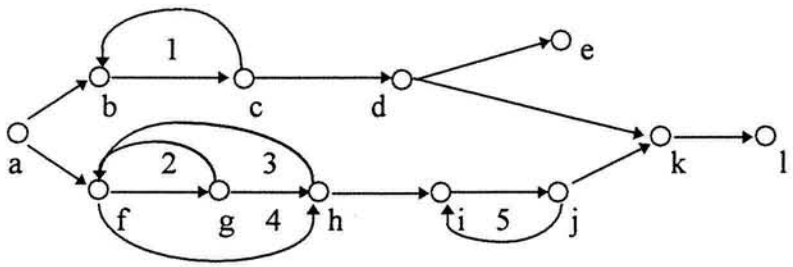


Figure 6 - Illustrative Interaction Diagram

In Figure 6, loops 2, 3 and 4 form a set of touching loops while loops 1 and 5 are non-touching loops with respect to each other and the set of touching loops 2, 3 and 4. As in control theory, the "gain", G, of a loop is the product of the probabilities of all the arcs on the loop. We define $L_1$ as the sum of the gains of all the loops in the graph, $L_2$ as the sum of the products of the gains of all pairs of non-touching loops, $L_3$ as the sum of the gains of all mutually non-touching triples of loops, and so on. (In Figure 6, for sink node l, $L_1 = G_1 + G_2 + G_3 + G_4 + G_5$ , $L_2 = G_1 \cdot G_2 + G_1 \cdot G_3 + G_1 \cdot G_4 + G_1 \cdot G_5 + G_2 \cdot G_5 + G_3 \cdot G_5 + G_4 \cdot G_5$, and $L3 = G_1 \cdot G_2 \cdot G_5 + G_1 \cdot G_3 \cdot G_5 + G_1 \cdot G_4 \cdot G_5$, where $G_i$ is the gain on the ith loop.) The determinant, D, of the graph is then given by:

$$D = 1 - L_1 + L_2 + \cdots + (-1)^p L_p \qquad (3)$$

where p is the size of the largest mutually non-touching set of loops.

Let $F_k$ be the product of the probabilities of all the arcs on the kth forward path to a given sink. (In Figure 6, there are two forward paths to the sink l.) Assuming that there are w forward paths to a sink, s, the probability of reaching the sink, $P_s$, can be computed as follows:

$$P_S = \frac{\sum_{k=1}^{w} F_k D_{\bar{k}}}{D} \qquad (4)$$

Where $D_{\bar{k}}$ is the value of D remaining when the nodes and arcs of the kth forward path are removed from the graph. (In Figure 6 for sink l, letting the first forward path consist of nodes a, b, c, d, k and l, the second forward path consist of nodes a, f, g, h, i, j, k and l, and the third forward path consist of nodes a, f, h, i, j, k and l, we have $D = 1 - (G_1 + G_2 + G_3 + G_4 + G_5) + (G_1 \cdot G_2 + G_1 \cdot G_3 + G_1 \cdot G_4 + G_1 \cdot G_5 + G_2 \cdot G_5 + G_3 \cdot G_5 + G_4 \cdot G_5) - (G_1 \cdot G_2 \cdot G_5 + G_1 \cdot G_3 \cdot G_5 + G_1 \cdot G_4 \cdot G_5)$, $D_{\bar{1}} = 1 - (G_2 + G_3 + G_4) + (G_2 \cdot G_5 + G_3 \cdot G_5 + G_4 \cdot G_5)$, and $D_{\bar{2}} = 1 - G_1$.)

To calculate the expected transaction time to a given sink, we need a number of definitions. First, let $T_{Fk}$ be the sum of the transition times for the kth forward path to the sink. Next, we form groups of forward paths to the sink as follows. A set of loops is a "touching collection" if each pair of loops in the collection has at least one node in common. (In Figure 6, loops 2, 3 and 4 form a collection.) All the forward paths to the sink that contain at least one arc from such a collection are said to form a group, q. Note that a forward path with no loops does not participate in a group, while a forward path with at least one loop participates in a group - either by itself or with other forward paths. (in Figure 6, with respect to sink l, paths a-f-g-h-i-j-k-l and a-f-h-i-j-k-l form one group, while a-b-c-d-k-l forms another.) Next, we partition the loops associated with a group of paths, q, into m(q) clusters of mutually non-touching sets of loops. Let the ith cluster consist of n(i) touching loops, j, j = 1, 2, ..., n(i). (In Figure 6, the group of paths a-f-g-h-i-j-k-l and a-f-h-i-j-k-l has two clusters {loops 2, 3 and 4} and {loop 5}.) Finally, let $G_{qij}$ be the gain from the jth loop from the ith cluster of loops in the qth group of forward paths and $T_{Lqij}$ be the corresponding sum of transition times. Then the expected transition time from the source to the given sink is given by:

$$\mu = \frac{\sum_k F_k D_{\bar{k}} T_{Fk}}{\sum_k F_k D_{\bar{k}}} + \frac{\sum_q F_q D_{\bar{q}} T_{Lq}}{\sum_q F_q D_{\bar{q}}} \qquad (5)$$

$$\text{where } T_{Lq} = \sum_{i=1}^{m(q)} \left( \frac{\sum_{j=1}^{n(i)} G_{qij} T_{Lqij}}{1 - \sum_{j=1}^{n(i)} G_{qij}} \right) \qquad (6)$$

Proofs of the above formulae are in Kim's unpublished note [Kim 1997]. Note that values from the above two generalized formulae give upper bounds (because an infinite number of trials is assumed) on both the probability and expected time to reach a sink node.

### 4.2 Illustration of the Computations for the FLS and NLS Retrieval Systems

For illustrative purposes a simplified interaction diagram for the FLS database retrieval system is shown in Figure 7. This is derived from Figure 2 by collapsing the sequential paths for states 2 through 6 and using the transformation formulae in figures 3 through 5 above. The transition probabilities are computed from the recorded frequencies on each branch. The transition times (which were not collected in detail in the experiment) are estimated but are consistent with the overall session times (see Table 1), other time data that was collected in the experiment and with our overall judgment as to relative times of the various transitions. (For example, we assume that it takes 50% longer to formulate a query in the FLS than in the NLS, that FLS queries take twice as long to type, that the NLS system takes 10% longer to parse a query and that the average data retrieval time is 20 seconds.)

The calculations for the FLS are as follows. For the success sink node in Figure 6, there is one forward path (k = 1 and q = 1) and two touching loops (i = 1 and j = 1, 2).

$$F_1 = 1.0 \times 0.258 \times 0.755 = 0.195$$

$$D = 1 - (1.0 \times 0.742 \times 0.848 + 1.0 \times 0.258 \times 0.163) = 0.329$$

$$D_{\bar{1}} = 1$$

The probability of successful completion of a task, $P_S$ is:

$$P_S = \frac{F_1 \cdot D_{\bar{1}}}{D} = \frac{0.195 \times 1}{0.329} = 0.593$$

The transition time for the forward path is:

$$T_{F1} = 330 + 270 + 50 = 650 \text{ seconds}$$

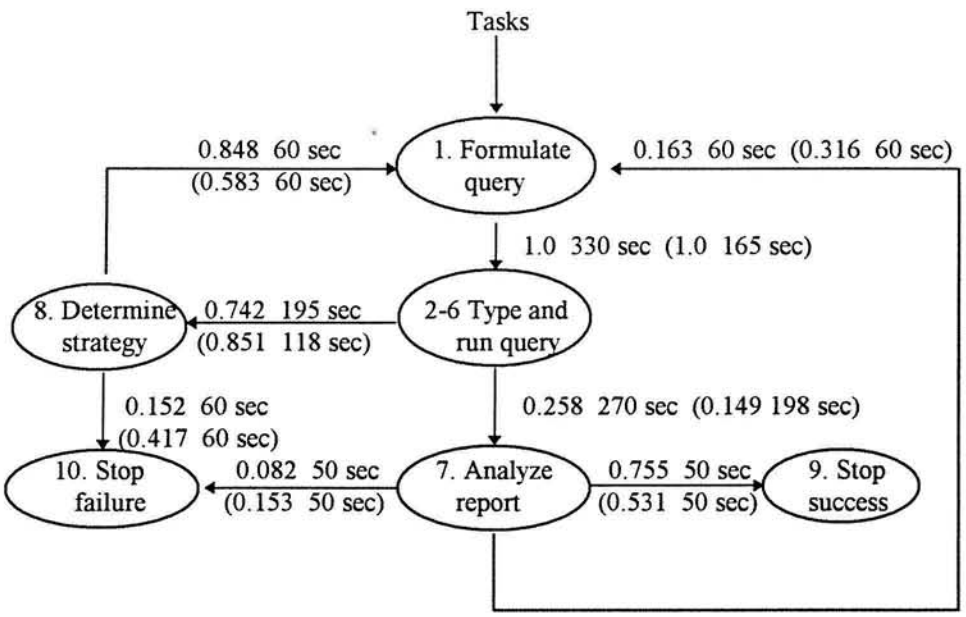


Figure 7. Query Level Interaction Diagram - FLS (NLS)

The transition time for two touching loops is:

$$T_{L1} = \frac{1.0 \times 0.742 \times 0.848 \times 585 + 1.0 \times 0.258 \times 0.163 \times 660}{1 - (1.0 \times 0.742 \times 0.848 + 1.0 \times 0.258 \times 0.163)} = 1203 \text{ seconds}$$

The expected time given that the task is successfully completed, $\mu_s$ is given by: $\mu_s = 650 + 1203 = 1853$ seconds or 30.88 minutes. Similarly, the probability of failure and the expected query completion time given that the interaction ends in a failure are given by: $P_f = 0.407$ and $\mu_f = 1798$ seconds. The expected total interaction time for a task, $\mu$, is the probability mixture of $\mu_s$ and $\mu_f$:

$$\mu = 0.593 * 1853 + 0.407 * 1798 = 1831 \text{ seconds or } 30.51 \text{ minutes.}$$

Finally, we can compute our second criterion, $\mu_*$, the expected time between successful task completions by $\mu_* = \mu_s/P_s = 52.07$ minutes.

Note from Table 1, that we have reproduced the actual probability values, $P_s$ and $P_f$ from the experiment. The computed task time, $\mu_*$ does not exactly match the empirical figure because we estimated some of the transition time data. However, it is satisfactorily close for our purposes. Table 2 summarizes the calculations for the FLS and NLS.

| | Base Case: Empirical data | | Case 2: Less system noise | | Case 3: Improved NLS $P_{34} = 0.40$ |
|---|---|---|---|---|---|
| | **FLS** | **NLS** | **FLS** | **NLS** | **NLS** |
| $P_s$ | 0.593 | 0.173 | 0.662 | 0.214 | 0.306 |
| $\mu_s$ | 30.88 min | 13.81 min | 26.46 min | 10.28 min | 10.83 min |
| $P_f$ | 0.407 | 0.827 | 0.338 | 0.786 | 0.694 |
| $\mu_f$ | 29.97 min | 12.75 min | 22.88 min | 12.19 min | 11.58 min |
| $\mu$ | 30.51 min | 12.93 min | 25.25 min | 11.78 min | 11.35 min |
| $\mu_*$ | 52.07 min | 79.83 min | 39.97 min | 48.04 min | 35.39 min |
| $n_t$ | 3.04 | 2.19 | 2.63 | 2.14 | 2.03 |

Table 2. Sensitivity Analysis: Effects of System Noise, and NLS Query Parsing

The "Base Case" corresponds to the data from the experiment as shown in Figure 2. It can be seen that the FLS is superior to the NLS on both criteria ($P_s$ and $\mu_*$). However, because it is easier to type queries in the NLS, $\mu_*$, the mean time between successes, is not all that much worse than that for the FLS (79 minutes versus 52 minutes.)

## 5. Sensitivity Analysis

We can now use our model to do some sensitivity analysis concerning the questions that were posed at the end of Section 2. First, we might wish to investigate the impact of a faster and more stable computer system on both the NLS and the FLS performance. (After all, the conditions under which the experiment was run were "unfair" to both systems!) Referring to Figure 2, suppose we reduce the system transition times $T_{34}$, $T_{56}$ and $T_{67}$ and the probabilities $P_{48}$, $P_{58}$ and $P_{68}$ to 50% of their former values. Note that we keep $P_{38}$, which measures the parsing capability (a software attribute), the same while decreasing $T_{34}$, the time to complete the parse. This scenario gives the results in columns 3 and 4 in Table 2. Both systems perform better in terms of our criteria, $P_s$ and $\mu_*$. The improvement in $\mu_*$ is more dramatic - of the order of 25% - because it reflects the improvement in both $P_s$

and $\mu_s$. One possibly unexpected insight from the model is that improving the hardware, ceteris paribus, improves the probability of success. This is because a reduction in system noise allows more queries to complete. Note that the FLS is still superior to the NLS on all counts.

To investigate how we can improve the NLS to match the FLS, note that $P_{34}$, the probability of being able to correctly parse a query, is much lower in the NLS than in the FLS (0.265 versus 0.492). Suppose the language designers believe they can improve the understanding capabilities of the NLS so that $P_{34} = 0.40$. What will be the impact on the relative performance of the FLS and NLS? To examine this question, we assume the improved system capabilities of case 2 but keep other system parameters (such as user typing and query composition capabilities) the same. As can be seen from the last column in the table, the NLS still performs relatively poorly with regard to the probability that a query attempt will be successful. However, it is now superior to the FLS in terms of the second criterion, $\mu_*$, the expected time between successful queries. Thus, if the users are willing to put up with a lot of trial and error, they should be able to satisfy their retrieval tasks faster with the NLS than with the FLS.

In the above illustration, we used empirical data from an experiment with a real system to calibrate the model prior to performing sensitivity analyses. If it were not possible to gather real data - perhaps because the system to be investigated is in the design stage - it might be possible to develop the interaction diagram with estimated probabilities and transition times.

## 6. Conclusion

As the whole world goes on-line so-to-speak, the relative efficiency of information retrieval becomes increasingly important. In this paper, we have outlined a general approach to the analysis of interactive computing systems that we hope will be helpful to system designers and others who are considering various design changes and the relative merits of different approaches and systems.

The paper makes two distinct contributions. The first is the streamlined method for computing the probabilities and transition times in directed graphs that was outlined in section 4. This method was then applied to the analysis of "interaction graph"

representations of human interaction with a computing system. In this context, the mathematical model is an approximation to the real nature of the interaction process. However, it is an approximation that can provide some significant insight. One generalization of the model that we are investigating involves making the probability of the user "giving-up" dependent on the number of previous unsuccessful trials (presumably, this probability would be a monotonically increasing function of the time already spent on the query.) Other generalizations, and other approaches such as visual simulation, are also being investigated. While simulation models are somewhat time-consuming to build, they have an advantage over closed-form mathematical models of the type described in this paper in that they allow analysis of the stochastic properties of a system as well as its average performance.

The second contribution of the paper is the overall approach to evaluation of interactive systems that we are proposing. Namely, the development of interaction graphs and the use of quantitatively-based sensitivity analysis to determine the impact of changes in system parameters and/or user behavior. We hope in the near future to adapt this approach to the analysis of web-log data at a major commercial Internet site.

## References


Blair, D. C., and Maron, M. E. An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System. *Communications of the ACM*, 28, 3 (March 1985), 289-299.

Chu, H., and Rosenthal, M. Search Engines for the World Wide Web: A Comparative Study and Evaluation Methodology. http://www.asis.org/annual-96/ElectronicProceedings/chu.html, 1996.

Courtois, M. P., Baer, W. M., and Stalk, M. Cool Tools for Searching the Web: A Performance Evaluation. *Online*, 19, 6 (November/December 1995), 14-32.

Graham, R. M., Performance Prediction. in *Software Engineering: An Advanced Course* Bauer, F. L (ed.) Springer-Verlag, 1977.

Gray, T. How to Search the Web: A Guide to Search Tools. http://www.palomar.edu/Library/TGSEARCH.HTM, 1996.

Jarke, M., Turner, J. A., Stohr, E. A., Vassiliou, Y., White, N. H., and Michaelson, K. A Field Evaluation of Natural Language for Data Retrieval. *IEEE Transactions on Software Engineering*, 11, 1 (January 1985a), 97-114.

Jarke, M., Krause, J., Vassiliou, Y., Stohr, E. A., Turner, J. A., and White, N. H. Evaluation and Assessment of a Domain-Independent Natural Language Query System. *IEEE Database Engineering*, (September 1985b).

Kambil, A., Doing Business in the Wired World. *IEEE Computer*. 30, 5 (May 1997), 56-61.

Kim, Y., Formulae for Performance Evaluation of Interactive Computer Systems. (unpublished private note), 1997.

Leighton, H. V. Performance of Four World Wide Web (WWW) Index Services: Infoseek, Lycos, Webcrawler and WWWWorm. http://www.winona.msus.edu/is-f/library-f/webind.htm, 1995.

Shinners, S. M. *Modern Control System Theory and Application.* Addison-Wesley, 1972.

Welty, C., and Stemple, D. W. Human Factors Comparison of a Procedural and a Non-Procedural Query Language. *ACM Transactions on Database Systems*, 6, 4 (December 1981), 626-649.