Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

A SIMULATION MODEL FOR THE FLOW OF CIVIL LAWSUITS

Minghui Yang
Mingjian Yuan
Ali K. Gunal

Research and Information Services
Office of the Administrator for the Courts
1206 S. Quince Street
Olympia, WA 98504

ABSTRACT

A preliminary simulation model developed from a project for Washington State Superior Courts is presented in this paper. The project is one of a few attempts in applying simulation technique to judicial system. The model will be used to experiment with alternative management policies to improve court operations. The issues in modeling and implementation including the software development and its linkage with Windows graphics user interface are discussed.

1. INTRODUCTION

In response to public concern about delay and increasing costs of litigation, courts, like any other business, face the management of limited resources and various demands. Providing efficient operations and quality justice has been a primary challenge to court administration. To help achieve this, a versatile tool has been in imminent need. Because of the appeals outlined below, simulation is being introduced to court management (see Yang 1989 for an example): (1) Simulation is a flexible tool for answering "what if" questions. It provides a good experimentation environment. To test a potential policy, courts were often forced to time-consuming and costly processes of conducting pilot experiments on designated subjects. For instance, to compare alternative case-scheduling methods, a court may have to be split to conduct different practices simultaneously. Instead, by carefully modifying the model parameters and functions, simulation can get timely comparison results, without the risk of disturbing the real system. (2) Simulation can provide numerical support for decision making. For instance, some rules of thumb are expected to expedite caseflow; such as earlier court intervention with cases, adding more hearings, or granting less continuances. Those rules help shorten the pending time of cases, but at an unknown cost of more judicial resources. With careful simulations, estimates for the additional resources can be obtained. The estimates can be referenced to evaluate whether a proposition is cost-effective.

This paper presents a preliminary simulation system for the flow of civil lawsuits. It is part of a current project for Washington State trial courts. In the next section, we introduce the background on the processes of civil lawsuits. In Section 3, a simulation model is presented. In Section 4, we discuss the implementation issues. In Section 5, we introduce the current development and validation. The paper is concluded with discussion in Section 6.

2. BACKGROUND

A civil lawsuit represents a complex judicial process. Significant differences can be observed in this process from one state to another, as well as from one courthouse to another within the same state. The complexity of the process is that at any stage there are a large variety of events that can happen. For example, Figure 1 displays some of the events that can take place at the beginning of a civil lawsuit. It can be observed in the figure that after the plaintiff files a complaint, the defendant may choose any of six different actions leading to different points. In the case of multiple plaintiffs or defendants, the process is much more involved.

Although complicated, the process can be described in simpler terms through some generalizations. First of all, it can be observed that any civil lawsuit will

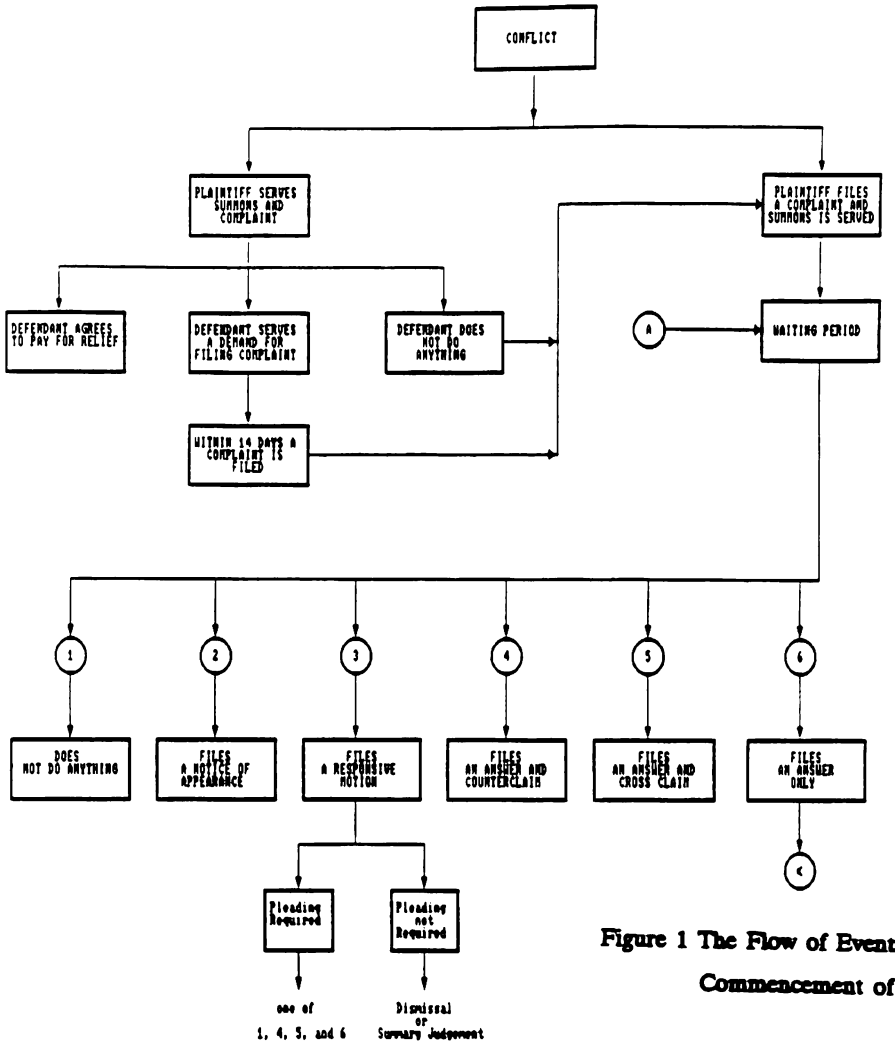


Figure 1 The Flow of Events at the Commencement of a civil Lawsuits

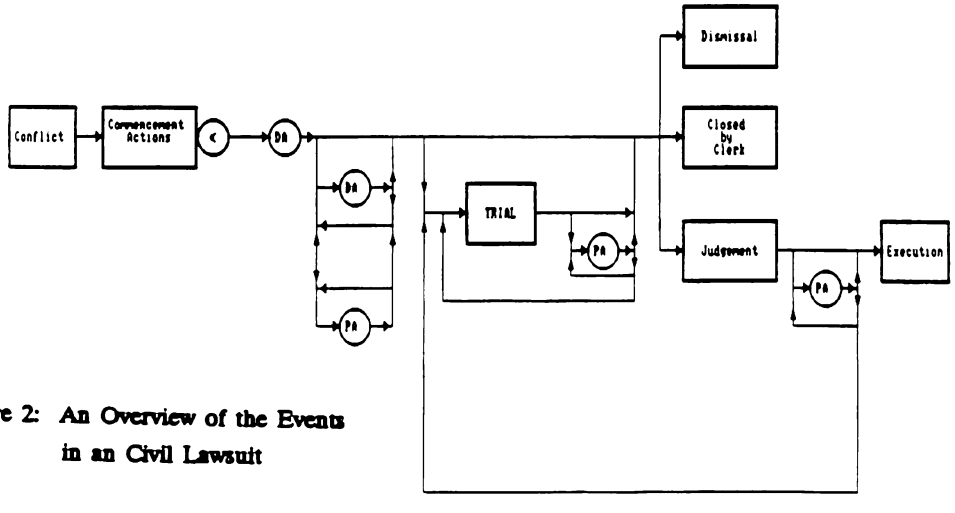


Figure 2: An Overview of the Events in an Civil Lawsuit

begin when one of the parties involved files a complaint with a court. Following the filing of the case, there may be a series of motions filed by one of the parties. These pretrial motions can be classified into two groups: procedural and dispositive motions. A dispositive motion can be defined as a motion which may cause the case to be resolved. For example, a motion for a summary judgment is a dispositive motion. A procedural motion can be defined as any non-dispositive motion. For example, a motion to request a medical examination or to request a trial is a procedural motion. There is not a predetermined sequence in which procedural and dispositive actions (PA and DA) can take place. However, it is clear that before a case is closed, there will be at least one dispositive action. Although there are various situations under which a case may be resolved, the resolution of a case can usually be classified as one of the following: dismissal, closure by court clerks, or by a judgment. After a judgment is filed, there may be a series of post-procedural actions which may result in the execution of judgment. It is also possible that one of the losing parties takes the case to an appeals court, and following the review of the appeals court the case may come back to court for further consideration. Figure 2 displays this general view of civil case process.

3. MODEL

As introduced in the previous section, civil lawsuits may consist of complex processes. In order to adequately model a process, three key components need to be considered: (1) a component describing the flow of cases, (2) a component describing the interactions of cases with a court, and (3) a component describing the statistical features of a court. By convention, caseflow is represented by an event graph. Figure 3 demonstrates the model.

To describe the flow of cases, a couple of views can be taken. One is to observe the flow from a case perspective. Figure 4 is a typical example: A case was filed at time t_1. It requested a sequence of hearings at time t_2, t_4, and t_8. The requested hearings were either heard or continued by the party. The case also requested a trial at time t_5. The scheduled trial at time t_{10} was continued (adjourned) at time t_7, when it requested a new trial date. The trial was assigned and held at time t_{11}. After that, the case requested a posttrial hearing and finally completed at time t_{14}. Mathematically, we may relate these events by a set of Markov chains. In particular, the transition probabilities are state dependent; i.e., the probabilities are conditional on case history. This implies that a number of conditional probabilities need to be estimated. In practice, the data collection for probabilities estimation is difficult and controversial. This prohibits us from approaching the model from this direction at the early stage. (However, a model adopting this view is currently under investigation.)

The other view observes the flow from a court perspective: Since the major actions that cases may request are either hearings or trials (which are also hearings, strictly speaking), we may divide the model into two subsystems: One describes the trial-related actions of a case, the other describes the hearing-related actions. Figure 5 demonstrates the two samples which are extracted from Figure 4.

In the trial-related subsystem, corresponding to each trial request, there is a trial scheduled. The scheduled trial can be continued if there is any other trial request, or it can be stricken if the case is resolved before the trial. There is also a possibility that a case is ready at the trial day but the court does not have resources available. The holdover cases will be rescheduled. The rescheduling rules vary from court to court. This subsystem is represented in the upper part of the event graph in Figure 3.

In the hearing-related subsystem, we use pairs of hearing requests and hearings. Similar to a trial, a requested hearing can be heard, continued, or stricken. Holdover hearings are not considered in this model because they are rare. This subsystem is represented in the lower part of the event graph in Figure 3.

The dual subsystems are used to model case images on trials and hearings. This separation allows watching case progress from two different angles. In particular, the dual angles allow us not to consider the interactions between hearings and trials, not because they are not important, but they incur huge complexity. The statistics estimated from these two images are combined to describe the whole system. Being based on *marginal* estimation, the estimates will be close to actual measurements.

Finally, while we may simulate the two systems separately, they can be put in one run to preserve as much synchronization as possible. That is, after a case is filed, we simultaneously create for the case a trial image and a hearing image whenever necessary. From then on, the two images progress independently as if they are simulated separately. This technique results in the combined event graph in Figure 3. Please be aware that the two subsystems are actually separate in concept.

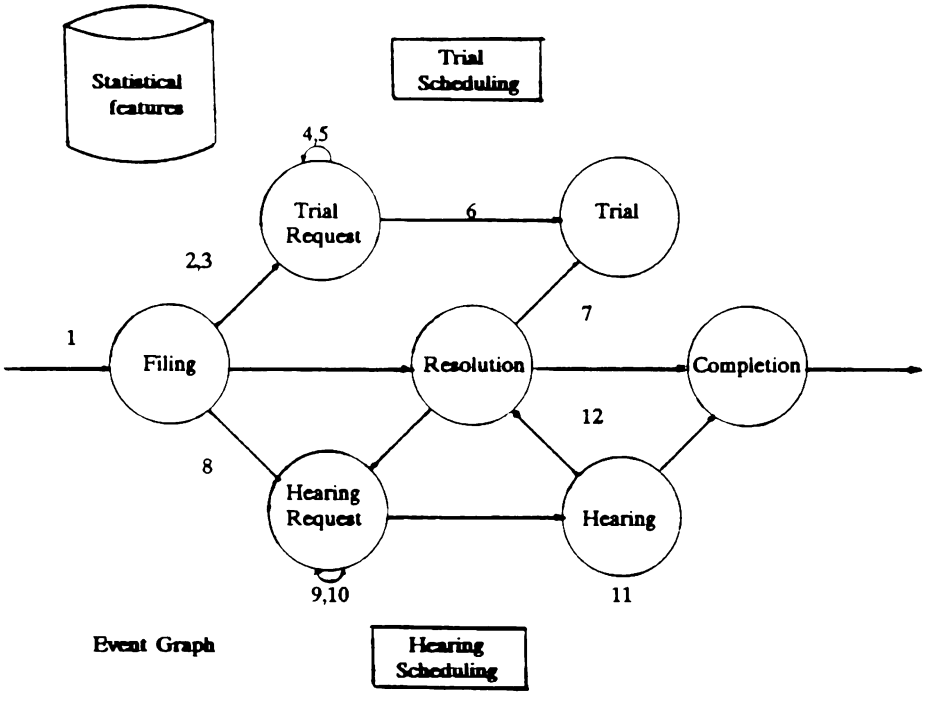


Figure 3: Civil Lawsuit Simulation Model

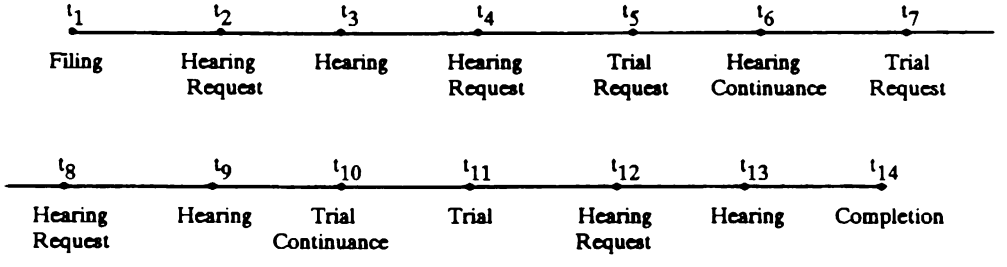


Figure 4: Sample Caseflow

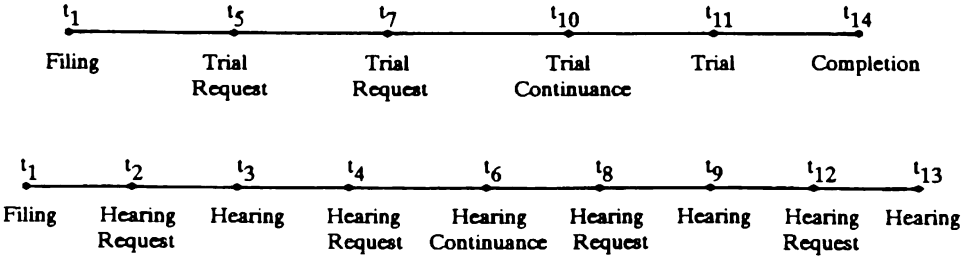


Figure 5: Sample Trial and Hearing Images

The second component of the model describes the interactions of cases with a court. The interactions mainly occur at the following occasions: (1) when a case requests or continues a hearing, (2) when a case requests or continues a trial, and (3) when a holdover case is rescheduled. The interactions are reflected in calendar scheduling, which is court dependent. The scheduling practices are translated into program logic and encapsulated in a couple of modules as indicated in Figure 3. By replacing these modules, different courts can be simulated, or alternative scheduling propositions can be experimented for the same court.

The third component of the model characterizes court statistical features. A typical set of statistics include the following: (1) case filing rate and related information; e.g., cause of action, (2) the probability that a case will request a trial (this determines whether a case will have a trial image), (3) the time distribution from filing to the first trial request, (4) the frequency distribution of trial requests if there is at least one, (5) the time distributions between neighboring trial requests, (6) the time distribution from trial request to the assigned trial date (this statistic will be provided by a scheduling module), (7) the time distribution from a trial to case completion, (8) the time distribution from filing to the first hearing request if there is one, (9) the frequency distribution of hearing requests (this determines whether a case will have a hearing image), (10) the time distribution between neighboring hearing requests, (11) the probability that a scheduled hearing is not held (because of continuance, unconfirmed, or stricken), (12) the time distribution from the last hearing to completion if a case is not resolved by trial. These statistics are the input that users need to enter. The numbers appear in Figure 3 indicate where these information is referenced in the model.

4. IMPLEMENTATION

For the selection of software to code the simulation model, three most important criteria were: (1) the dynamic data storage capacity, (2) the friendliness of the resulting simulation program, and (3) cost of each copy of the resulting simulation program. For a typical courthouse in a metropolitan area, the total number of lawsuits filed is usually described in thousands (e.g., 59,975 cases were filed in 1990 at King County Superior Court in Seattle area). Furthermore, most of these cases stay in the system for a long period of time (sometimes years). Considering that each case is represented by an entity in the model and that each entity has several attributes (such as number of hearings, trial date, etc.), the data storage requirement of a realistic simulation model can easily go up to megabytes. Therefore, it is imperative the simulation environment should provide access to such large amounts of computer memory.

The second criterion concerns the friendliness of the system since it could be accessed by less sophisticated computer users for analysis and decision making purposes. This requirement led the simulation team to consider a graphical user interface for the final model because of the clear superiority of such an interface in providing friendly and easy-to-use programs.

The last criterion in choosing the software was the cost of each copy of the resulting simulation program since installations at different courthouses were required before and after the model was operational.

After evaluating various alternative simulation software, a decision was made to develop the model in C language using the Windows environment for the user interface. The reasons for this choice are as follows: First, Windows allows an application program to have a large data storage space allocated dynamically at run time. With a personal computer that has four megabytes or more RAM memory, most of the courts can be simulated without running into memory problems. Secondly, the Windows environment provides a sophisticated graphical user interface that enables an application program to have a high level of visual interaction with the user. Consequently, even naive users of personal computers can, after brief training, effectively use the application program. Finally, in comparison with commercial simulation software with equivalent capabilities, the cost of this approach is negligible although the development cost may be slightly higher due to the more intensive programming requirements (see Gunal, Yang and Yuan 1992). Besides these advantages, the Windows is a multitasking environment; multiple copies of the same program can be run simultaneously with different inputs to make comparisons on computer screen easier. Also, during long simulation runs, a user has the possibility of performing other tasks inside the Windows environment (e.g., word processing, spreadsheet application, etc.) although doing so may slow down the execution of the simulation program. Figure 6 demonstrates the software structure.

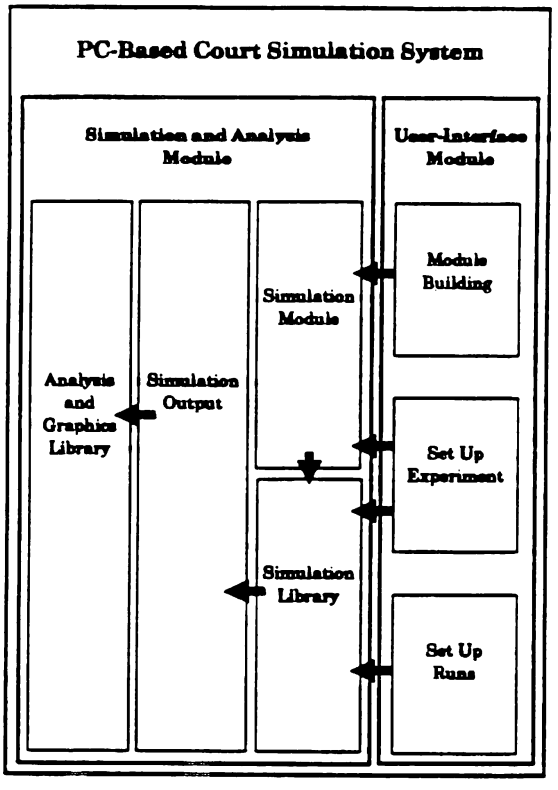


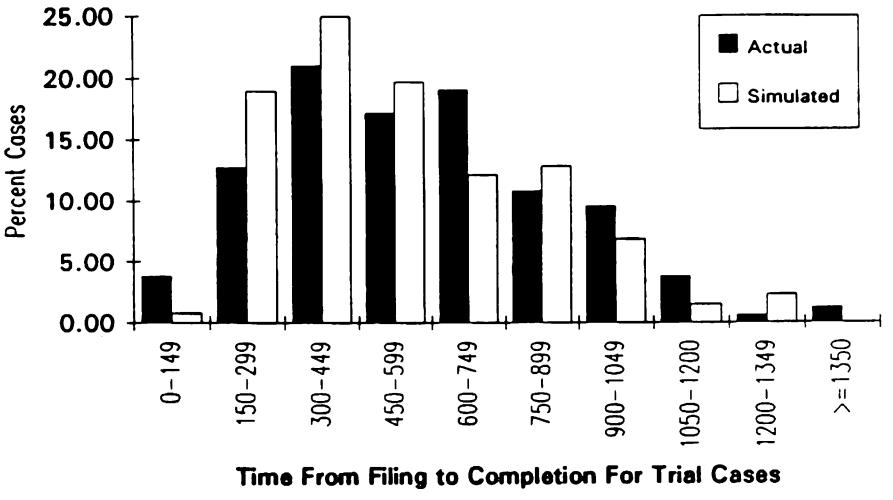
Figure 6 Court Simulation Software Structure

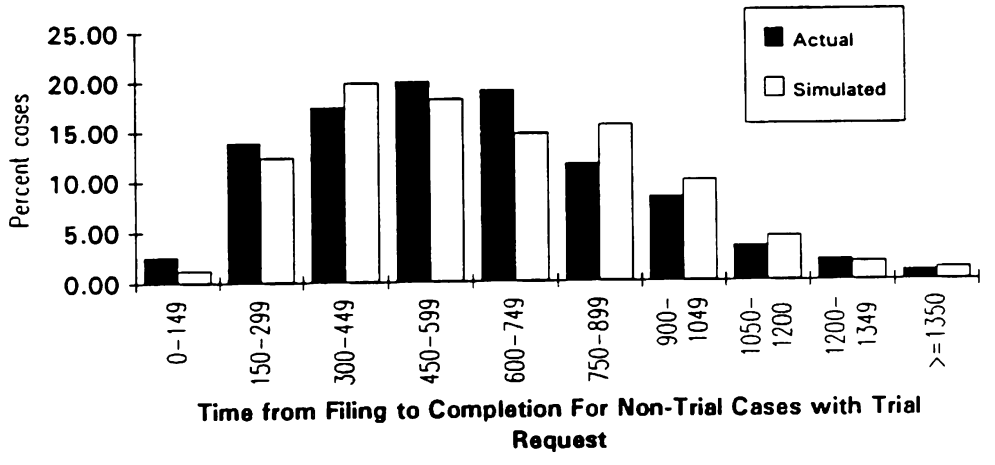
5. CURRENT DEVELOPMENT AND PRELIMINARY VALIDATION

The development of the court simulation software is delicate and time-consuming. To help verification and validation, the software development is separated into six modules. In each module, a set of special functions will be designed, tested, and linked with other developed modules. A module is to be fully tested and validated before it can be combined with any others. The following are brief descriptions for the modules: (1) Simulation library: This module contains the simulation clock mechanism, event list, random number generation, dynamic memory management, and other support functions. We adopt part of the library from DISK++ (Blair and Selvaraj 1989). (2) Basic windows interface: This module provides the functions to start a simulation, monitor the simulation progress, and report final statistics inside the Windows environment. (3) Basic model: This model consists primarily of caseflow. We replace the calendar scheduling functions by the statistics collected from actual data. The judicial resources are not included in the model at this stage. The skeleton model is used to validate the dual-system view. We run the model and compare the simulated results with the actual statistics. If there is any discrepancy, it results from the modelling. We have constructed such a model for one of our pilot courts and have done extensive input analysis. We select as the validation comparison criterion the time from case filing to disposition. In addition, the comparisons are made with respect three types of cases. Figure 9 demonstrates the comparisons. The comparison result indicates that our model seems to get good approximations. More tests will be designed to further validate the model. (4) Enhanced model: At this stage, more details will be added to the basic model; e.g., the calendar scheduling logic, judicial resources requirements. One function will be incorporated into the basic model at a time until it can reasonably reflect the reality. (5) Complete interface functions: Only after we have certain confidence on the model, a set of interface functions will be developed. This module will include model building, parameters input, and experiment capability. (6) Statistical analysis module: It is known that simulation is essentially a statistical experiment. To get meaningful interpretations, appropriate statistical procedures need to be employed. This module will provide the analysis capabilities. To the time being, we have finished the first 3 modules and is now trying to enhance the model.

6. DISCUSSIONS

The simulation system presented in this paper is preliminary. The model described here provides a basis for constructing more sophisticated models. Some additional issues discovered so far are as follows: (1) Most of the smaller courts have to allocate their resources to the processing of all types of lawsuits unlike larger departmentalized courts where each department is dedicated to the processing of a subset of the case types. Consequently, in such smaller courts there is a significant amount of interaction between civil and other types of lawsuits. The interactions occur when different types of cases compete for court resources. Because of the time limitations placed on the processing of criminal cases, they are always assigned a higher priority. As a result, it is difficult to isolate the processing of civil cases from other cases. We have thought of including criminal cases in the model, but only





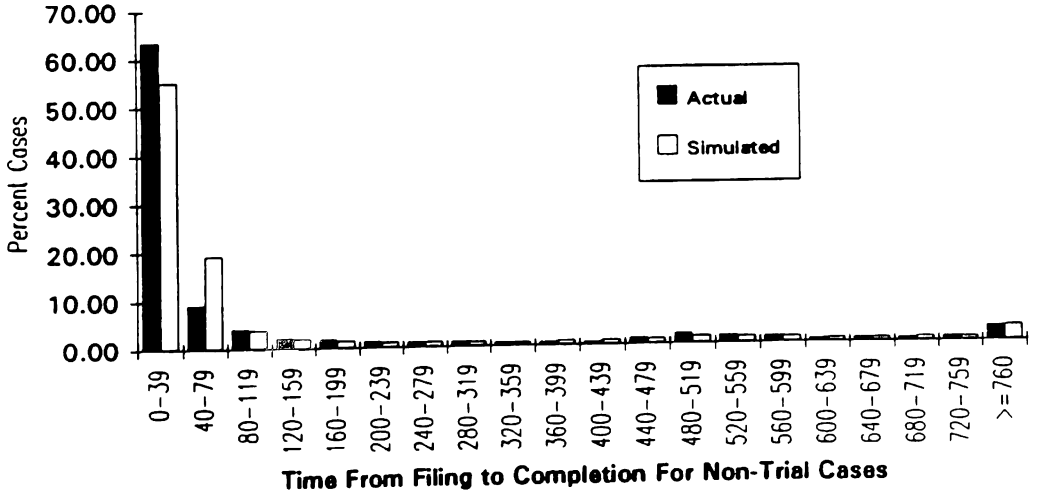


Figure 7 Comparisons for the Real and Simulated Values of the Time from Filing to Completion

limited to the places where interactions arise.
(2) As mentioned earlier, different courts have different practices or legal culture. However, the design of simulation needs to be general. We plan to use some concepts from object-oriented programming to achieve generality in our model. We are attempting to encapsulate the complicated flow and scheduling logic into a set of parameters so that by varying the parameters different courts can be defined. This attempt is still under investigation.

ACKNOWLEDGEMENTS

This material is based upon work sponsored by State Justice Institute Grant No. SJI-91-06F-A-034 and Washington State Office of the Administrator for the Courts.

The authors would like to acknowledge the helpful comments of Robert P. Barnoski and the anonymous referee.

REFERENCES

Office of the Administrator for the Courts (1991), *A Citizen's Guide to Washington Courts*, 1991.

Blair, E.L. and Selvaraj S. (1989), " DISK++: A C++ BASE LIBRARY FOR OBJECT ORIENTED SIMULATION", *Proceedings of the 1989 Winter Simulation Conference*, 301-305.

Gunal, Ali K., Minghui Yang, and Mingjian Yuan (1992), "Using Windows for Discrete-Event Simulation", paper presented at TIMS/ORSA Joint Meeting at Orlando, Florida, April 1992.

Office of the Administrator for the Courts (1990), *The 1990 Report of the Courts of Washington.*

Solomon, Maureen and Douglas K. Somerlot (1987), *CASEFLOW MANAGEMENT IN THE TRIAL COURT- Now and For the Future*, American Bar Association, Chicago, Illinois.

Yang, Minghui (1989), "A Simulation Model for Washington State Juvenile Detention Facilities", *Proceedings of the 1989 Winter Simulation Conference*, 957-962.

AUTHOR BIOGRAPHIES

MINGHUI YANG received his M.S. in operations research and Ph.D. in statistics from Oregon State University in 1986 and 1988. Since April 1988, he has been a research specialist in Research Division, the Administrator Office for the Courts, Washington State. With a background in systems engineering and electrical engineering, he was a visiting scholar at Department of Operations Research, Stanford University in 1982. He taught at Northern Jiaotong University in Beijing, China before he came to the United States. His main research interests are in stochastic modeling and its applications.

MINGJIAN YUAN is a research specialist in the Administrator Office for the Courts of Washington State. He received his M.S. and Ph.D. in Industrial and Systems Engineering from The Ohio State University in 1987 and 1991. Starting from this Fall, he will join the faculty of Industrial Management Department at National Yunlin Institute of Technology and Management (Taiwan, R.O.C.). His research interests include decision support simulation analysis and variance reduction techniques.

ALI K. GUNAL received his Ph.D. degree from Texas Tech University in December 1991 in Industrial Engineering. He worked for the Office of Administrator for the Courts in Washington State as a research specialist. Currently, he is working in a manufacturing simulation center of University of Michigan- Dearborn. He is a member of ORSA/ TIMS, SME, and ACM. His interests include object-oriented simulation and simulation software development.