CHAPTER 5

EXPERIENCE WITH THE MASH SYSTEM

Conceptual design of the MASH system began in the summer of 1970. Implementation of initial modules of the system began in the spring of 1971, and a basic framework for microsimulation and associated activities was completed by the end of 1971. During the next six months, the initial demographic source model was completed, translated into object model form, and was implemented within MASH. Initial and exploratory simulation results were available in May 1972.

Active development work continued on the system through the summer of 1974, but emphasis shifted gradually from refinements in the system to the development of additional source and object submodels to form the complete microanalytic model of the household sector. Policy experiments were formulated resulting in the introduction of alternatives into the source model, and were executed using MASH. Recent effort has been directed toward substantial source model extensions and revisions and the accompanying object model modifications with only moderate changes to the system structure and content of MASH.

Since 1971, there have occurred approximately 5,000 separate computer executions of the MASH system for a variety of purposes--system checkout, model development, model testing, simulation activity, and production of results. Based upon this experience, it is appropriate to ask a number of questions concerning the development and use of the system. Were the hypotheses described in Chapter 1 concerning the utility of a time sharing environment justified? How well did MASH meet its own system design goals and how well did it support the goals of the overall project? What lessons were learned in the development and in the use of MASH that might be helpful in further work in this field? What directions seem promising for future evolution of MASH and other such systems? The following sections address these and similar questions and provide some tentative evidence concerning the underlying issues.

Development Experience

The MASH system was essentially developed in a three and one half year period from March 1971 to September 1974. Approximately three and one half man years of effort were used in the development of the system, and resulted in approximately 40,000 lines of operational source code. Since September 1974, system modifications and extensions have continued but at a much slower rate.

The interactive time sharing system used for MASH development work met our initial expectations in most ways. The ability to enter, translate, test and modify programs rapidly and interactively contributed substantially to efficient system development. On-line program and text entry contributed substantially to the generation of adequate documentation of source program modules. On-line access to project files allowed multiple users to access and use the project's modules and outputs effectively. Development time was substantially decreased as a result of working in an interactive computing environment. Shortcomings in the environment were related the specific environment itself.

One unexpected and annoying problem arose as a result of problems encountered with PDP-10 hardware and several earlier levels of monitor software. During periods when the system was failing several times a day, it became extremely frustrating to perform any but simple, straightforward tasks. During those periods when mean time to failure was of the order of several hours, each failure could mean the loss of the past 15-30 minutes of work, regardless of how quickly the failure was remedied. Under such circumstances the strategies and actions of all interactive users on the system became quite defensive and efficiency suffered.[1] While such failures might occur in a batch environment, they would have manifested themselves to the user generally only in terms of increased turnaround time.

The use of Fortran as the primary implementation language, augmented by assembly language where required for efficiency, was a pragmatic decision. It allowed a group of programmers of different skill levels working on object model modules to interface easily with the MASH system. The standard recognized benefits of use of a higher level language -- relatively rapid code generation, reasonably efficient generated object code, some self-documentation, and others -- were realized. Since the system itself was written largely in Fortran, its source programs could be distributed to programmers using the system and could be understood by them for purposes of understanding system functions exactly and interfacing with them.

As anticipated, the choice of Fortran also had drawbacks. The lack of more general data structures introduced cumbersome constructions into the code. The lack of any meaningful character string capability was quite annoying. The absence of more powerful syntax such as implicit array calculations and nested if-then-else control structures caused additional code to be written that would not have been required in some other higher level languages.

---

[1] Such periods of frequent failure occurred only very occasionally during system development and did not affect it adversely in any substantial manner. As hardware and software for interactive computing becomes reliable, there is every reason to believe that such episodes will become increasingly rare. Nevertheless, it was interesting to observe how the impact of such failures was amplified through the interactive network.

MASH is a system that supports socioeconomic research and policy evaluation, and as such -- like much similar computer based activity in the social sciences -- it reflects the manner in which socioeconomic research and policy evaluation processes occur. The model for such activity is not sequential; rather it reflects an environment in which progress occurs as the result of a complex search process. Programming activity that supports such research rarely begins with a coherent and well-defined set of specifications, but rather a general set of research objectives which are translated into computer based requirements as research proceeds. Since research contains a component of trial and error, to the extent that computer based activity "tracks" it there will be inefficiencies in the implementation process and vestigial modules of code generated in intermediate versions of the system.

Within such an environment, the conventional wisdom dictates that generality should be emphasized as much as possible in the system design stage so that implementation activity will be allowed substantial dimensions of freedom in which to follow and meet perceived requirements. Such a strategy was adopted with respect to MASH within the restrictions imposed by the choice of computer and language and by the resources available for the task. While such a strategy is generally effective, it often implies a trade of efficiency for flexibility.

The trade of efficiency for flexibility is one that is often made in social science computing because the end product is generally not used for production purposes. In the case of MASH, however, the system had both to continue a process of gradual evolution and to serve as a reasonably efficient production program for simulation exercises. These dual and conflicting goals were approached by generalizing system design to the maximum extent possible and then optimizing parts of the system implementation on the basis of operating experience. This approach is nicely described by Knuth [K4] as follows:

> "There is no doubt that the grail of efficiency leads to abuse. Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil.
>
> "Yet we should not pass up our opportunities in that critical 3%. A good programmer will not be lulled into complacency by such reasoning, he will be wise to look carefully at the critical code; but only *after* that code has been identified. It is often a mistake to make *a priori* judgments about what parts of a program are really critical, since the universal experience of programmers who have been using measurement tools has been that their intuitive guesses fail."[2]

After MASH had been running in production status for a period of time, it was possible to determine areas of substantial inefficiency and make investments in these areas to increase their efficiency. For

---

[2] Donald E. Knuth, "Structured Programming With Go To Statements," *Computing Reviews.* Vol. 6, December 1974, p. 268.

example, the substitution of a pseudo-cache buffer for time series values in place of a strategy of searching the data bank in the aggregate model solution program reduced the costs of that module by approximately 90%. Analysis of specific operating characteristic modules led to ways of reducing the cost of their execution without altering them functionally. Analysis currently being performed on the virtual memory simulator offers the promise of increasing the efficiency of that major module. It is expected that such performance measurement activity will continue to take place throughout the life of the system, and a variety of measurement devices have been embedded in it to assist the identification of appropriate targets for such investment.

Insufficient evidence was generated on the question of the extent to which an interactive environment helped to link a community of professionals through ability to share results easily. Significant interplay between members of the model development team occurred, but such a result would have been expected under any circumstances. The user feedback mechanism through the protocol file was more useful for documentation than for feedback, since there have not yet been any "unknown" users of the system.

Operational Experience

MASH has now been used for approximately 3 years to produce results of a variety of simulation experiments. During this period, the source models have undergone substantial revision, and the nature of policy questions to which modelling activity has been oriented has slowly broadened.

In general, it appears that most of the functions required to support research and policy exploration have been included within MASH. The open ended command language, coupled with the free form of the various commands, has allowed the addition of functions as the need for them was realized. Further, interpretation of such commands is rapid relative to the response time requirements of a system user. MASH commands are generally powerful commands in the sense that one or a few commands can initiate a complex resource consuming and result producing procedure. Thus, the overhead required by the interpretation of commands is more than compensated for by the ability to optimize the process invoked by the command, independently of the form of specification. Such flexibility is available because of the interpretive nature of the system.

One expectation that has not been realized to date is the use of MASH by persons who are not computer oriented. It was hypothesized that the user orientation of the MASH command language would encourage social scientists to use the system directly. Such direct use would increase the self-sufficiency of researchers in the use of such systems by reducing substantially their dependence upon technically trained intermediaries. To encourage such use, features such as data browsing were added to the language to

strengthen the user's interaction with data. The command language was designed to be pseudo-English in style on the assumption that such a format would be more appealing to a non-programmer user of the system. However, direct use of the system by non-programmers was observed only occasionally; rather, it was observed that social scientists preferred in general to use the system through trained technical intermediaries.

There are a number of plausible explanations for this observed behavior. First, the design of the system may have been unappealing for use. What may appear to the designer of a system to be an appropriate design and easy to use may not be comprehended by the intended users in the same manner or with the same degree of enthusiasm. Second, the pseudo-English nature of the command language can be deceptive and frustrating. While commands do appear to be sentences, there is nevertheless a reasonably inflexible set of syntactical rules to which they must adhere in order to be decipherable. To the user, small variations in form may appear insignificant, but to the system many such variations are illegal. While this problem can be alleviated to some extent by the use of synonyms, optional "noise" words and free ordering of phrases, the user senses the underlying rigidity and reacts cautiously to it. Further, some users have a personal reticence to become directly involved with machinery. For such users, direct contact with a computer terminal may provoke more anxiety than dealing with computing machinery at some distance, such as in a batch environment or through an assistant. Finally, the more traditional methods of division of labor allocated hierarchically among members of a research team may be perceived as yielding greater benefits than direct contact between senior research staff and computer systems.

To the extent that computer related work is performed by trained intermediaries, the strategy of creating command languages oriented to users without computer skills deserves to be questioned. Command languages oriented toward more technically trained users could provide more detailed control functions that could be combined more flexibly than higher level mechanisms, since it could be assumed that system users would obtain a good technical understanding of both internal and external system functions. On the basis of experience to date with MASH, it appears that more consideration should be given to recognizing explicitly the role of technical intermediaries in such group research efforts, training staff to fulfill such roles, and orienting computer based tools toward use primarily by them.

There has not yet been time to test thoroughly the hypothesis regarding the adequacy of access to a central system as opposed to system portability. The evidence that exists is essentially positive. Some use of the system has been made from remote locations using the switched public telephone network for communication. More recently, two other PDP-10 installations have been used for large scale simulation experiments, and little difficulty was encountered either in moving the system or in accessing it remotely.

The more formidable barriers to remote access that have existed in the past have been communications costs and administrative barriers to use of foreign facilities. Prior to 1975, almost the only alternative to the use of voice grade lines for low speed data communication was the use of privately multiplexed lines, which are only economical given geographically concentrated demand of sufficient volume to justify the investment and dedicated line costs. More recently, licensing of "value added" communications networks such as Telenet promises to reduce data communications costs drastically while at the same time minimizing the effect of distance. Such networks will reduce the cost of remote low speed data communication with systems like MASH to an almost insignificant level, probably within several years.

It was hypothesized that administrative barriers to use of remote computing facilities would be observed, but there was insufficient external use to be able to study the issue adequately. Such barriers have been observed in the past in other situations. They generally arise through the commitment of an institution to purchase or lease computer capacity which is not fully utilized. For the institution, the marginal cost of additional internal computer use is low, while external use is not only presumed to be at some higher average cost but also represents a transfer of real funds out of the institution. It is therefore not surprising that institutions often establish barriers of varying severity to use of external computing facilities. It remains to be seen whether the availability of inexpensive digital communication will induce increased "trade" between these local computing economies and whether some specialization of labor will result from the resultant increase in the size of the market. A good discussion of these issues is contained in Berg [B10].

The costs of performing simulation experiments using MASH have been moderate, but not as low as was initially hoped. At the present time, the price of executing one year of a simulation run consisting of three iterations through a micropopulation of 20,000 persons and solution of the associated macroeconomic model is approximately $100. This price is based upon average costs incurred by a computer center operated by a not-for-profit organization; the institution's charging algorithm sets prices of individual computer resources in approximate proportion to their cost. More recently, commercial service bureau costs have declined to approximately the same level, partially as a result of the introduction of more cost-effective compatible computer hardware.

Within a typical year of simulation, the time spent in the virtual memory software is approximately 40% of the total central processor time used.[3] This figure should be compared with a minimum of one-fifth of it which would be required if all data accesses were to real memory--assuming that a sufficient

[3] This figure excludes the time spent executing the paging functions which is relatively low given a reasonably local address trace and an adequate number of page frames in primary memory.

amount existed to satisfy all data storage requirements. Thus, the use of simulated virtual memory to provide substantial data address spaces buys this space at a cost of an approximate 47% increase in execution time during simulation activity.

In summary, while the operational experience acquired with MASH to date is not sufficient to provide definitive answers to the hypotheses conjectured in Chapter 1, some conclusions are possible. MASH has been heavily used, with some worthwhile results. The attempt to narrow the gap between social and economic research staff and computer systems has not been very successful, and it now appears that the alternative of explicitly training technical intermediaries and constructing systems oriented to them deserves stronger consideration than it received in the design of MASH. While MASH has been sufficiently flexible to accommodate all object models constructed to date, the operating costs have exceeded the expectation of the research team. A continuing program of identifying areas of poor performance and making appropriate modifications will ultimately make the existing implementation near optimal within the present design, and to the extent that cost expectations for the processes specified are realistic, costs will be close to an acceptable minimum for the tasks to be performed.

Directions for the Future

Programming systems, like proverbial old Generals, do not often die, but they do have a tendency to fade away. Unless they continue adapting to new environments and new demands, they are in danger of being replaced by other products based upon newer technology and responding to more recent requirements. A program or programming system is a piece of vintage capital resulting from an initial investment at one time and from possible subsequent investments at later times. As technology progresses more effective investments become possible, so that there is reason to expect that eventually investment in newer and more effective programs will occur.

If an obituary were to be written for MASH, however, it would probably look somewhat like this:

> "Washington, D.C. 1982. The MASH system for microanalytic simulation died this year of old age. It was 11 years old.
>
> "MASH was conceived in 1970 and was born in 1971. Its formative years were spent at The Urban Institute in Washington, D.C. During its most productive years (1973-1978) it was a prodigious producer of simulation results relating to social science research and policy exploration; at one time, it was one of the most powerful programs in its field.
>
> "As time passed, however, its comparative advantages declined, and the rigidity of its structure became more noticeable. The increasing complexity of the demands of the research and policy world in which it existed created a burden that was difficult to cope with. Several years ago it began to operate only in a limited area, performing only those tasks for which it was most capable.

> "It is survived by a number of more powerful and applicable systems for socioeconomic research and a respectable variety of useful research results and policy assessments."

While the ultimate demise of MASH is not inevitable, there are already apparent weaknesses which, if not remedied, will be sufficient to cause it to be replaced in the medium term future. In general, these weaknesses fall into three areas: (1) hardware base; (2) software implementation strategy; and (3) system structural design.

The implementation of MASH began in 1971 on a PDP-10 computer with a KA10 processing unit. Since that time, Digital Equipment Corporation has announced the KI10 processor, the KL10 processor, and most recently the DECsystem 20 computer. All of these components are upward compatible from the KA10 processor and offer improved price-performance ratios. The effect of these new product offerings is that in 1976 it is possible to obtain a DECsystem 20 which is about 50% faster than a KA10 processor based PDP-10 for approximately 50% of the price of the earlier machine. This is a gain of a factor of 4 in price-performance. It is highly likely that there will be additional products in the future that will be functionally equivalent to the PDP-10, since the product has been well received and there exists a robust collection of software available for it.

Nevertheless, there may come a time when no further improved models of the PDP-10 computer will be made. Although such an event is unlikely, its occurrence would limit MASH to existing hardware stock that would soon be rendered ineffective by technical progress. Transfer of the MASH system to different hardware might then be required for it to continue to be economically viable.

Microsimulation experiments depend substantially upon the processing resources of the computer environments in which they are executed. That is, much of the cost of simulation experiments derives from the substantial processor time required to apply a sizeable number of complex operating characteristics to a large population of micro entities. Even with the moderate costs achieved to date, the cost of large simulations is still quite high. Further, the number of operating characteristics would be expected to increase as model development continued, and the complexity of each characteristic would probably increase also. Larger populations will be required as distributional questions become important and as small cell conditions become of interest to users.

Current informed prognoses regarding the capacities and costs of the pure hardware component of future computer systems are in general agreement that the costs of such processor hardware for approximately equivalent capacity will continue to decrease steadily at a rate of at least 15-25% per year. For entire computing systems, such a consistent decline is increasingly offset by other costs such as input-output equipment, and software which do not rapidly decline and which are already beginning to dominate

computational costs. Thus, for many computing environments, the declining cost of computation may not have a significant effect since processor resources may not be a critical bottleneck.

Processor resources are critical, however, for some classes of computations. These are often referred to in the industry as "number crunching" tasks and include, *inter alia*, numerical weather prediction, trajectory and orbit calculations, nuclear reactor calculations, simulation of physical systems, and geological explorations. Such application areas have specifically encouraged the construction of "super machines" with processing capacities beyond the current state of the art, such as the LARC, the IBM 7030 (Stretch), and the CDC 6600, 7600, and Star machines. In the past, such machines have provided both a significant expansion in processing power, or bandwidth, (and often primary memory capacity also) and a lower cost per computation than had been associated with processor economies of scale existing prior to their operation.

More recently, it appears that significant advances in computational bandwidth can result from increasing parallelism in hardware system design. The Burroughs Illiac IV with its 64 parallel processors represents a recent large-scale effort to obtain such capacity. Another example of parallel processing is provided by the IMSAI hypercube design, in which Intel microprocessors are connected in a 4-dimensional hypercube and operate simultaneously, executing synchronized processes in parallel. IMSAI estimates that their largest hypercube containing 4x4x4x4=256 processors has a bandwidth of approximately 350 million instructions per second (MIPS). Since technical progress in large scale integration is now extremely rapid, such an approach could yield rapid decreases in the costs of high bandwidth processors as well as other benefits. One initial drawback is that programming techniques for such parallel architectures are not as well understood as those for uniprocessor systems.

Our experience indicates that large scale socioeconomic microsimulation is a processor intensive task. While other resources such as significant amounts of primary and secondary memory are also required, the complexity of such models, the size of the micropopulations desired for use, the length of simulations, and the number of simulation exercises required for thorough exploration of a set of policy questions all combine to place enormous demands upon the processing resources of a computer system. Within the next 5-10 years it is highly probable that new computer systems will be available that will provide significantly increased processing bandwidth. Provided that such systems are supplied with efficient user software that includes mechanisms for the effective use of parallel processors for user tasks, these systems will be extremely attractive to designers of application systems for implementing microanalytic simulation models. Unless there is a substantial increase in the processing power of future PDP-10 environments and in the cost of computation, it is quite possible that the focus of such activity could shift to these alternative computing environments.

Advances in software could also have the effect of reducing the effective lifetime of the MASH system. While the strategy of using Fortran IV as the primary implementation language may have been a good choice in 1970, it will become increasingly less so in future years. As the implementation of the microanalytic model described in [O6] has proceeded, it has become increasingly apparent that Fortran IV is an undesirably low level language for effective implementation of such models. The weaknesses of Fortran as an implementation language have affected both the development of the MASH system and the specification of object models that execute within it.

At the system level, it is increasingly apparent that a major function of MASH -- if not the dominant one -- is data management, though not in the sense that the term is commonly used. At present, MASH deals with many types of entities -- micropopulation units, structural and membership relationships, genealogical links, aggregate time series, lists, equations, classifications, name lists, commands and others. Many of these entities are used for reference occasionally, while others such as the micropopulation entities are subject to intensive processing during a simulation exercise.

The first group of entities is certain to expand if systems for microanalytic simulation become more complex, yet MASH does not now possess effective mechanisms for managing a complex or large "space" of such entity types. Further, implementation in Fortran allows only array structures to be represented efficiently, and although stacks, queues, hierarchies and lists are utilized within MASH, their management is cumbersome. Higher level languages and environments that support efficiently the maintenance of large collections of diverse entity types will have a strong absolute advantage as implementation vehicles for future systems for microanalytic simulation.

Management of micropopulation entities for simulation must be both flexible and efficient. MASH's management of these entities is only somewhat flexible and somewhat efficient. Greater flexibility will be required in order to allow object models to represent as accurately as possible the source specifications of the modeler. Greater efficiency is required so that the series of simulation requirements often required for meaningful results can be executed at a price that can be afforded by research teams and policy analysts. Certainly systems that structure, store, and process their microentity data in a more flexible and efficient way will have an advantage over MASH.

While the specification of object model modules can be expressed in Fortran IV, the resulting code often obscures the formulation of the module and is lengthy compared with that formulation. Of the higher level languages known to the author, only Simscript II comes reasonably close to being an effective language for model specification, and it has substantial shortcomings for implementing models such as that

described in [O6]. Given the experience obtained both with MASH and with other microanalytic simulation activities, it may now be possible to begin a first attempt to design a more powerful set of language constructs for specifying microanalytic models. Such a language would require many components for handling algebraic calculation, decision making, stochastic imputation, entity and attribute manipulation including creation and deletion of either or both, self-correcting feedback mechanisms or "tracking," summarization through event reporting and table generation, population housekeeping and maintenance, and other similar functions. If such a specification language existed, it would be of very substantial assistance in helping programmers to construct object model modules and in helping modelers to understand these modules.

Such a language would probably contain a large and diverse set of statement types and would ideally allow translations from source to object model modules with only moderate effort. However, the efficient implementation of such a language would also be important because of the compute intensive nature of microsimulation. The requirement for efficiency rules out interpretation, but a two-phase translation process utilizing a high level language supported by the host computer as an intermediate representation might be effective. Such a strategy would partition the process into two parts, the first consisting of a detailed analysis of the new language statements in the source program and the specification of a program in the high level language for their efficient execution and the second consisting of actual code generation for that module. Only the first part would require new investment in software.

Such a strategy has been used in a number of system implementations. For example, the STATPAK/TARELA table generation system, developed by Statistics Canada primarily for production of 1971 Decennial Census results, uses such a system. It shares with MASH the requirements of flexible specification but efficient execution. This system is described in [P3] and [P4].

Finally, the system design underlying MASH appears increasingly restrictive. While it is believed that it represents a substantial advance over earlier systems supporting work in this field, both the experience obtained in the past several years and the increased appetites of research workers for larger models combine to cause one to regard the current design as restrictive. While MASH supports a macroeconomic model with one sector greatly expanded into a micro population of 3-level hierarchical structures, more recently formulated models require either a more general data structure for the microentity in the microsector of the model or multiple heterogeneous microsectors or both. Such models can be supported by MASH either with great difficulty or not at all. Future systems that can support such a broader universe of models will be likely to be used more intensively than MASH.

Conclusion

It appears that in general many of the hypotheses concerning the use of MASH and the environment in which MASH was built and operated were supported by observations made to date. Certainly the use of an interactive environment was on balance a very effective mode of operation. On the other hand, the reluctance of social science researchers to use the system directly raises doubts both about the structure of the command language and the extent of the demand for direct control of a computer based interactive process by social scientists. But in general, the MASH system appears to have provided a reasonably productive environment for microanalytic simulation and will probably do so for a number of years in the future.

In the longer run, technical progress and demand for modelling environments and tools allowing more general models and more efficient execution of microsector submodels will cause MASH as it now exists to lose its advantages as a modelling environment. It is expected that these newer systems will be characterized by increased processor bandwidth possibly provided by new parallel architectures, higher level languages with a stronger correspondence with the components of microanalytic model specification and execution, and a structure that both allows a substantially larger universe of microanalytic models to be implemented and processes large microsector operations in a more efficient manner. However, before such a replacement occurs, MASH will have served a valuable purpose both in supporting the construction and use of a number of complex, second generation socioeconomic microanalytic models and as and a crude prototype of its more effective successors.