GENERAL ELECTRIC

General Electric Company
Command and Information Systems
450 Persian Drive
Sunnyvale, California

(408) 734-3571

A SIMULATION MODELING APPROACH TO UNDERSTANDING THE SOFTWARE DEVELOPMENT PROCESS

by

A. H. Stone
G. Y. Wong
J. A. McCall

Presented at the
Fourth Annual Software Engineering Workshop
November 19, 1979
Goddard Space Flight Center
Greenbelt, Maryland

A SIMULATION MODELING APPROACH TO UNDERSTANDING THE SOFTWARE DEVELOPMENT PROCESS

A. H. Stone
G. Y. Wong
J. A. McCall
Command and Information Systems
General Electric Company
Sunnyvale, California

ABSTRACT

This paper, resulting from research doen for the Air Force Office of Scientific Research (AFOSR), describes an assessment of the feasibility of utilizing simulation techniques to aid in the management of large-scale software developments. A model of the software development process was constructed, state-of-the-art prototype simulation tools used, and an experiment conducted to demonstrate the feasibility. A result of this effort is the concept of a Software Development Process Simulator which could be utilized to assist in project planning (cost estimation) and project control (progress status assessment).

INTRODUCTION

Significant progress has been made during the last few years in identifying the problems and complexities involved with the development of software systems and providing techniques to overcome these obstacles. What has evolved is a more disciplined environment for the production of software. Formal specification, design, and implementation methodologies are being developed. More milestones and visable software products during the development phases have been identified. Software support tools have become more sophisticated in providing assistance in the design and development of software. Considerable error and cost data have been collected and a better understanding of the software development environment is evolving. Cost, productivity, and reliability studies add to this understanding and provide data for prediction and estimation. The factors in software quality and associated metrics are being studied to obtain more quantitative measurements of the quality of a software product. Demonstration projects are being undertaken to prove the effectiveness of new techniques.

All of these R&D efforts contribute to a more disciplined and structured development process. This discipline and structure lends itself to more effecitve management. Most of the tools and techniques that have resulted from these R&D efforts support micro-level activities within the software development process. Few assist in the management of the entire process.

A potential management tool, made possible by the more disciplined approaches taken to software development, is a simulation model of the development process. Simulation models traditionally have been used by management for analyses such as system design studies, trade-off analyses, performance assessments, and impact analyses. A model of the software development process would facilitate these same types of analyses of the development effort itself. The analyses supported by such a tool would span both management planning (cost estimation) and control (progress and impact assessment).

The initial step toward developing a simulation tool to aid in the management of a software development involves developing the concept of such a tool and assessing the feasibility of using simulation techniques to construct a model of the software development process. This paper describes the results of this initial step. Specifically, under a contract sponsored by the Air Force Office of Scientific Research, the objectives were to:

- Determine the feasibility of applying simulation techniques to modeling the software development process.
- Describe the software development process in a manner conducive to developing a simulation model.
- Provide insights into modeling specific aspects of the software development process.
- Discuss the potential benefits and use of such a model.

MODELING APPROACH

A model is a representation of a system which gathers together in one place our understanding of the behavior of that system. The purpose of developing a model of a system is to have a vehicle for predicting the behavior of the system under various conditions. The adequacy of the model is normally determined by five criteria: (1) applicability – does the model answer the questions that we want to ask?; (2) confidence – is the model sufficiently accurate for our purposes?; (3) completeness – is the model broad enough to encompass all phenomena of interest?; (4) minimality – have system states that are unnecessarily discriminated been combined?; and, (5) independence – have system states that involve interacting factors been decomposed into multiple states?

The software development process has been modeled by researchers in software engineering primarily for the purpose of predicting the life cycle costs associated with developing computer software. The models that have been developed are macroscopic models which use analytic techniques to represent the behavior of a software development. However, where the analytic modeling approach treats the software development process as a "black box" process, the simulation modeling approach attempts to decompose the process and understand its internal behavior. With the simulation modeling approach, we view a software development organization as a collection of interdependent elements which act together in a collective effort to achieve the goal of implementing computer software. These elements are primarily personnel resources, such as programmers and analysts, and computer resources, such as terminals, computers and software tools.

Simulation modeling is the process of developing an internal representation and a set of transformation rules which can be used to predict the behavior of, and relationships between, the set of elements composing the system under study. The internal representation of a software development system is described by system state variables, such as software size and complexity, personnel productivity, and project status and progress. The transformation rules describe the interdependence between these system state variables. These transformation rules may be analytic – expressed in the form of functional relationships, or they may be representational – expressed in the form of an algorithm. Thus, the simulation approach provides for a microscopic view of the software development process.

The adequacy of the simulation approach of modeling the software development process is summarized in the following table.

Criteria	Evaluation
applicability	excellent
confidence	promising
completeness	excellent
minimality	excellent
independence	excellent

Applicability is excellent becuase a simulation model can be oriented toward studying any aspect of the software development process. Confidence is promising because if the accuracy of part of the model is not sufficient, then that part of the model can be expanded to a greater level of detail. Completeness is excellent because of the microscopic view that is taken with the simulation approach. Minimality and independence are both excellent because the feasibility inherent in the simulation approach allows system states to be combined or decomposed at the discretion of the modeler.

SOFTWARE DEVELOPMENT PROCESS MODEL

The challenges of managing a software development are immense because it is a multi-element process which is highly coupled and highly complex, and there are wide variations in controllable and uncontrollable variables between projects. In the past, the technique used by managers has been to decompose the software development process into "independent" subprocesses and manage those separately. This technique, generally following the Wolverton description [WOLV 74] , does not usually reflect a very accurate model of the way software is currently being developed or is not in enough detail to analyze the causes of poor performance [TURN 76]. There has been recognition in recent years that interaction occurs, and that a continuous configuration management effort is required to keep the product of each phase up to date and consistent. Thus the traditional widely used "model" of the software development process is outmoded, no longer representing a true picture of how software is developed.

An accurate model of the software development process must account for several things. First, the idea that redesign, revisions to requirements, and changes to the source code take place constantly during the process, as the knowledge of the system evolves, must be acknowledged. Secondly, these revisions and corrections take place as a function of the activities the development personnel perform, not as a complete recycle of a phase; as evolution not revolution. Figure 1 is a representation of this evolution of knowledge on a timeline.

DECOMPOSITION OF THE SOFTWARE DEVELOPMENT PROCESS

Conceptually, the software development process is a process which is driven by a concept of, or requirement for, a target system and utilizes the resources of a software production factory to produce an operational system. The target system is a software system which has certain desired

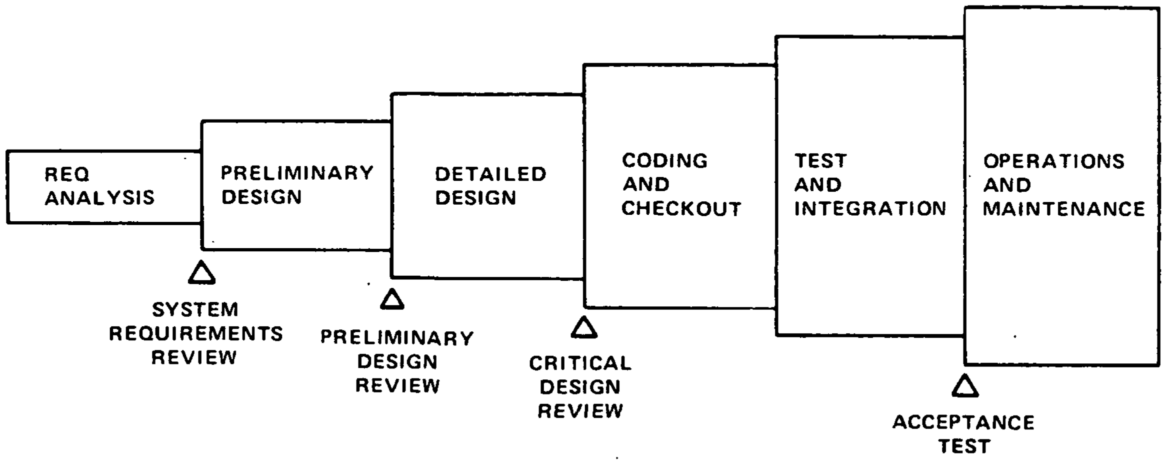


Figure 1. Timeline Representation of the Software Development Process

characteristics. These characteristics have an impact on the amount of resources which are consumed or utilized in the process of producing the operational system. The software production factory is the organizational, staffing, and development strategies superimposed on the resources of a project group (consisting of personnel and development tools), which provide the production capability and environment for accomplishing the system development. The operational system, the output of the process, is represented by the documents, data, and code produced as a result of the software development.

Imposed on this development process are a series of milestones which represent intermediate formal reviews of the progress towards the operational system. Further, there are documentation requirements which define what products are to be delivered. Almost all software developments have these milestone and documentation requirements. Perhaps the most rigorous set of requirements are those imposed by military standards.

Our approach to modelling the software development process was to decompose each of the phases in the high-level model described in the previous section into greater detail. The methodology used to accomplish this used military standards as a perspective and involved a three-dimensional view:

(1) Identification of the products of the software development process;

(2) Identification of the activities that comprise the process;

(3) Identification of the factors and resources that represent the target system and the software production factory.

Thus, we arrived at the model shown in Figure 2.

MODEL UTILITY

The conceptualization and decomposition of the software development process as a sequence of activities, as shown in Figure 2, provides a model which can be used at several levels. At one

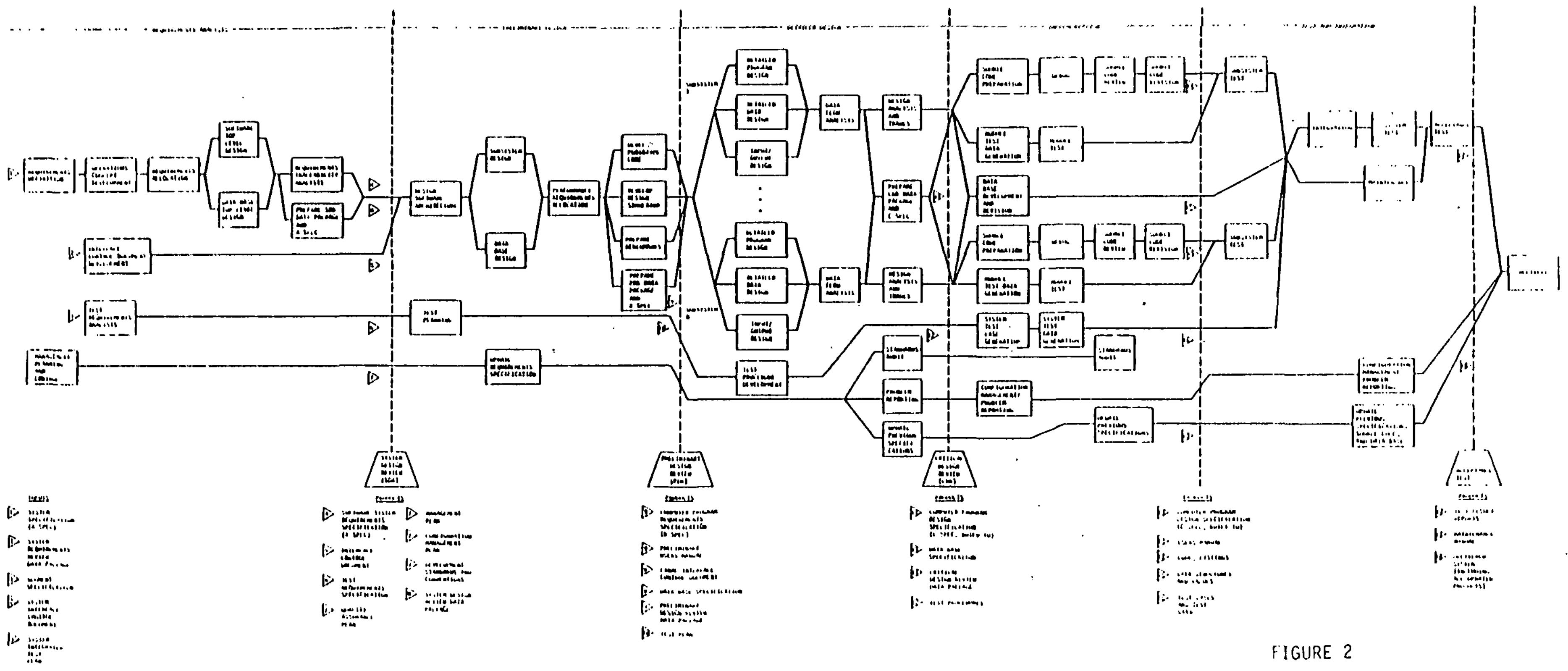


FIGURE 2

SOFTWARE DEVELOPMENT PROCESS MODEL

level, the model can be used as a checklist for planning and progress status. The list of activities typically performed during a software development, and the interaction can be used to plan the activities to be performed in a future development. Once this plan is established, completion of these activities can be used as a status measurement more accurate than the normally imposed milestones.

At the next level, the model can be used as a PERT-COST tool. The current prototype tool that has been developed has the capability with which activity delay times could be modelled as distributions representing worst case, most likely, and best case estimates of the schedules for those activites. The simulation would then result in the calculation of the expected time in which the network of activities would be completed. An enhancement to the PERT-COST approach available with our simulation approach is modelling resource usage as a function of time also.

A third level, that at which the prototype simulator was developed, is a high level process model. At this level, the activities are modelled at a relatively high level. Sensitivities in the development plan and in the assumptions made in the model development could be analyzed. At a high level, impacts of using different techniques and tools could also be analyzed.

The last and most detailed level, is a detailed process model. At this level all of the concepts introduced in prior sections would be utilized to model the activities. The analysis capabilities possible in the process model mentioned above would be of greater fidelity due to the finer detail of the activity models. At this level of capability, the full complement of support to the management planning and control of a software development project would be provided.

EXPERIMENTING WITH THE MODEL

To further evaluate the feasibility of simulating the software development process, a prototype simulator was constructed and demonstrated by modeling a past software development. This prototype was developed with the idea in mind that it could eventually be extended to provide a full software development process simulation capability. In essence, this prototype enabled us to experiment with the basic concepts of the software development process model and provided some experience during which lessons could be learned and refinements in our approaches could be accomplished.

DESCRIPTION OF THE EXPERIMENT

The experiment was oriented toward modeling a past large-scale software system development. The simulated results were then compared with the historical data that was maintained about the development effort. This approach to an experiment is more modest than a full validation of the simulation model in which the simulated results would be used to predict the actual results, and a comparison of predicted versus actual would provide a validation criterion. Our experiment was more a calibration of the model to assess if, in fact, a development effort could be modelled to some degree of accuracy. Calibration is utilized by analytic techniques also (RCA PRICE-S and Putnam's SLIM) to tune the analytic model to the development organization. We envision this practice also pertaining to the Software Development Process Simulator, where various parameters or internal tables within the simulator could be tuned to a particular development organization by modeling past developments.

The software system development that was modelled consisted of three major subsystems (or CPCIs). The system was a command and control ground system developed under contract for the Air Force. The three subsystems ranged from 75,000 to 150,000 lines of JOVIAL source code each (including comments). Complete statistics on the development activity were maintained, including the number of design problem reports, software problem reports, and source code statistical profiles, as well as manpower expenditures.

RESULTS OF THE EXPERIMENT

Again, we were trying to calibrate our model and therefore were interested in achieving the highest possible agreement with the recorded development data. For this experiment, we examined actual manpower data from the simulation. The line labelled "ACTUAL" in Figure 3 is the graph of the data from the Air Force project superimposed on the graphs of the simulation results. Table 1 is a legend to be used with Figure 3. Cursory examination of the data in Figure 3 shows a clear correspondence between observed and experimental values.

Comparison of the graphs in Figure 3 shows that there is a 4.98% error between the areas, underneath the observed and experimental data curves. These results are considered quite acceptable. Some of the peaks of spikes seen in the actual data can be attributed to five-week fiscal months, which plotted at a granularity of one month causes higher manpower expenditures to be illustrated.

Resource	Title
SYSE	System Engineer
ANLT	Analyst
PROG	Programmer
QA	Quality Assurance Personnel

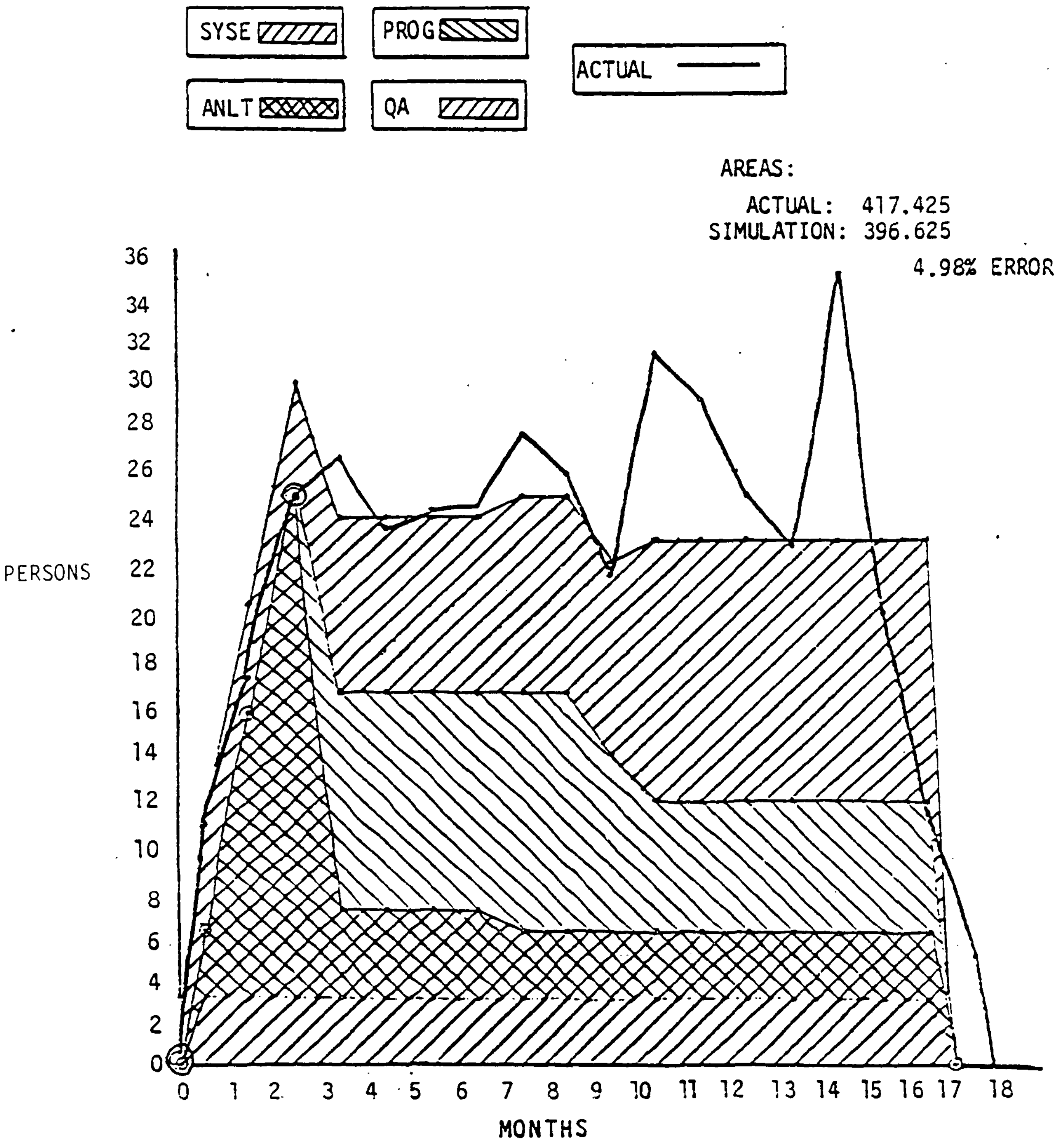


Figure 3 Experiment Results

CONCLUSIONS

Our research has covered the spectrum from concept formation to analysis to experimentation. First, we addressed the problem of how to apply simulation techniques to study the software development process. Then, we investigated what the characteristics of software developments are, based on the "world-view" established by our simulation approach. Finally, we studied when our modelling methodology is valid by learning how to design simulation experiments based on our modelling approach. Table 2 summarizes our major accomplishments, and indicates where we go from here.

Table 2

Conclusions Matrix

Accomplished	Future
(1) Simulation Approach	
Combined activity-product model forms conceptual basis.	Identification of simulation variables, model rules, model inputs and outputs.
(2) Process Decomposition	
Activity-product network demonstrates practical application	Detailed specification of activities, products, factors, and resources.
(3) Simulator Development	
Simulator prototype demonstrates experimental feasibility.	Data collection to support full experiment.

REFERENCES

MCCA 79 McCall, J. A., et al., "A Simulation Modeling Approach to Understanding the Software Development Process," GE TIS 79CIS009, June 1979.

TURN 76 Turn, R., M. Davis, and R. Reinstedt, "A Management Approach to the Development of Computer-Based Systems," International Conference on Software Engineering, October 1976.

WOLV 74 Wolverton, Ray W., "The Cost of Developing Large-Scale Software," IEEE Transaction on Computers, Vol. 23, No. 6, 1974.

MAILING ADDRESS

Albert H. Stone
General Electric Company
Command and Information Systems
450 Persian Drive
Sunnyvale, California 94086
(408) 734-3571, x44

ISP

A SIMULATION MODELING APPROACH TO UNDERSTANDING THE SOFTWARE DEVELOPMENT PROCESS

A.H. STONE

GENERAL ELECTRIC COMPANY

COMMAND AND INFORMATION SYSTEMS

SOFTWARE TECHNOLOGIES GROUP

SUNNYVALE, CALIFORNIA

ISP

STUDY OBJECTIVES

AFOSR CONTRACT NO. F49620-78-C-0054

CONTRACT MONITOR: LT. COL. GEORGE McKEMIE

<u>OBJECTIVES</u>

- DETERMINE THE FEASIBILITY OF APPLYING SIMULATION TECHNIQUES TO MODELING THE SOFTWARE DEVELOPMENT PROCESS
- DESCRIBE THE SOFTWARE DEVELOPMENT PROCESS IN A MANNER CONDUCIVE TO DEVELOPING A SIMULATION MODEL
- PROVIDE INSIGHTS INTO MODELING SPECIFIC ASPECTS OF THE SOFTWARE DEVELOPMENT PROCESS
- DISCUSS THE POTENTIAL BENEFITS AND USE OF SUCH A MODEL

ISP

WHY SIMULATION?

COMPARING TECHNIQUES:

CRITERIA	TECHNIQUE EVALUATION	
	ANALYTIC	SIMULATION
APPLICABILITY	LIMITED	EXCELLENT
CONFIDENCE	MARGINAL	PROMISING
COMPLETENESS	LIMITED	EXCELLENT
MINIMALITY	EXELLENT	EXCELLENT
INDEPENDENCE	LIMITED	EXCELLENT

IN ADDITION, SIMULATION:

- HAS DEMONSTRATED SUCCESS AT MODELING PEOPLE PROCESSES AND MAN-MACHINE INTERACTIONS
- ALLOWS FLEXIBLE YET DETAILED MODELING
- PROVIDES A USABLE TESTBED FOR EVALUATING THE MODEL
- PROVIDES A VEHICLE FOR MEANINGFUL "WHAT IF" ANALYSES

ISP

MOTIVATION FOR UNDERSTANDING

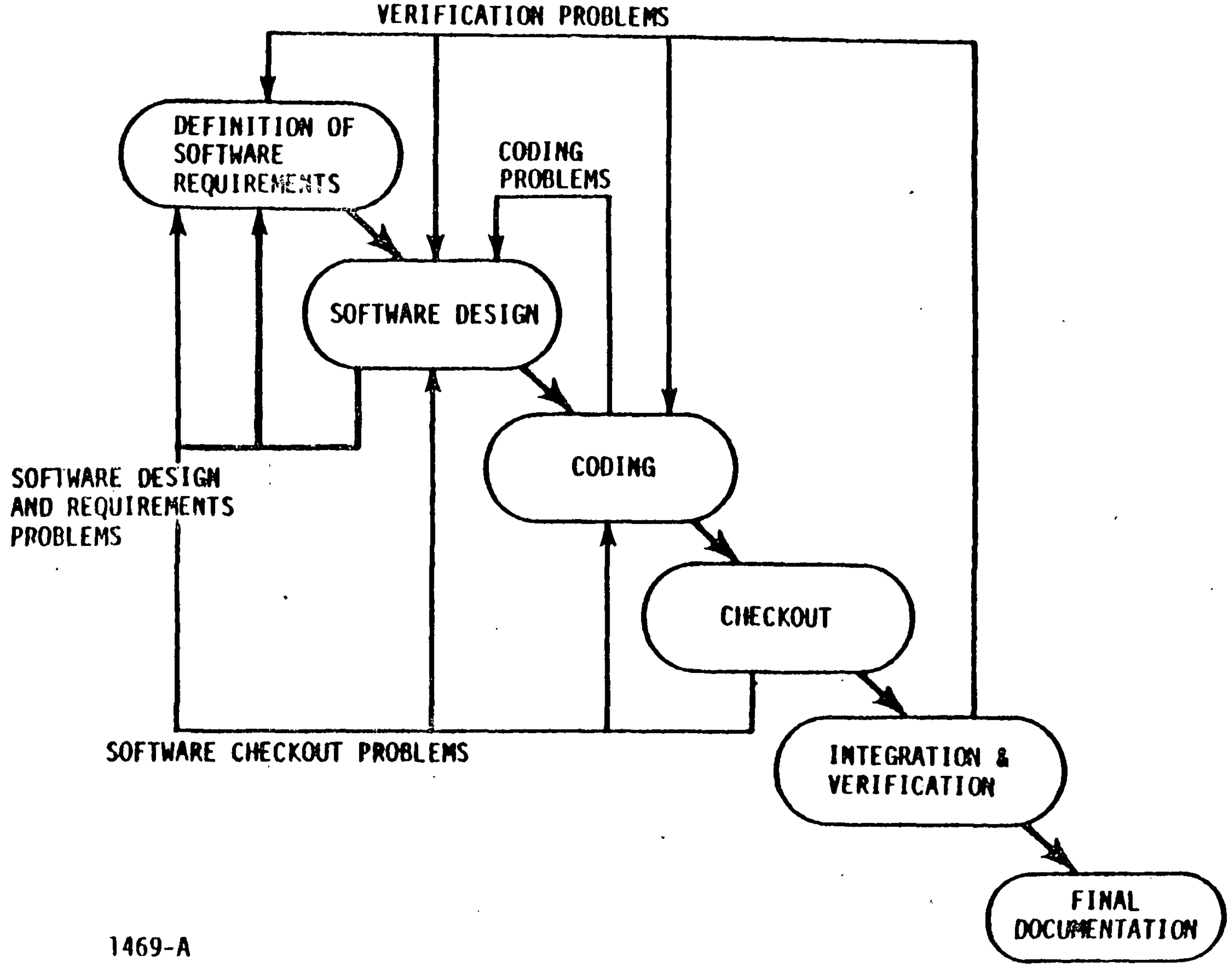
TOOLS ARE NEEDED FOR

- PROJECT PLANNING
  - COST ESTIMATION
  - TIME REQUIREMENTS
  - RESOURCE REQUIREMENTS
- PROJECT CONTROL
  - PERFORMANCE ASSESSMENT
  - BOTTLENECK ANALYSIS
  - RESOURCE TRADEOFF ANALYSIS
- TECHNOLOGY ASSESSMENT
  - ASSESSMENT OF IMPACT OF NEW TOOLS, TECHNOLOGIES, AND METHODOLOGIES
- CONTINGENCY PLANNING
  - IMPACT ASSESSMENT

ISP

6151

SOFTWARE DEVELOPMENT PROCESS

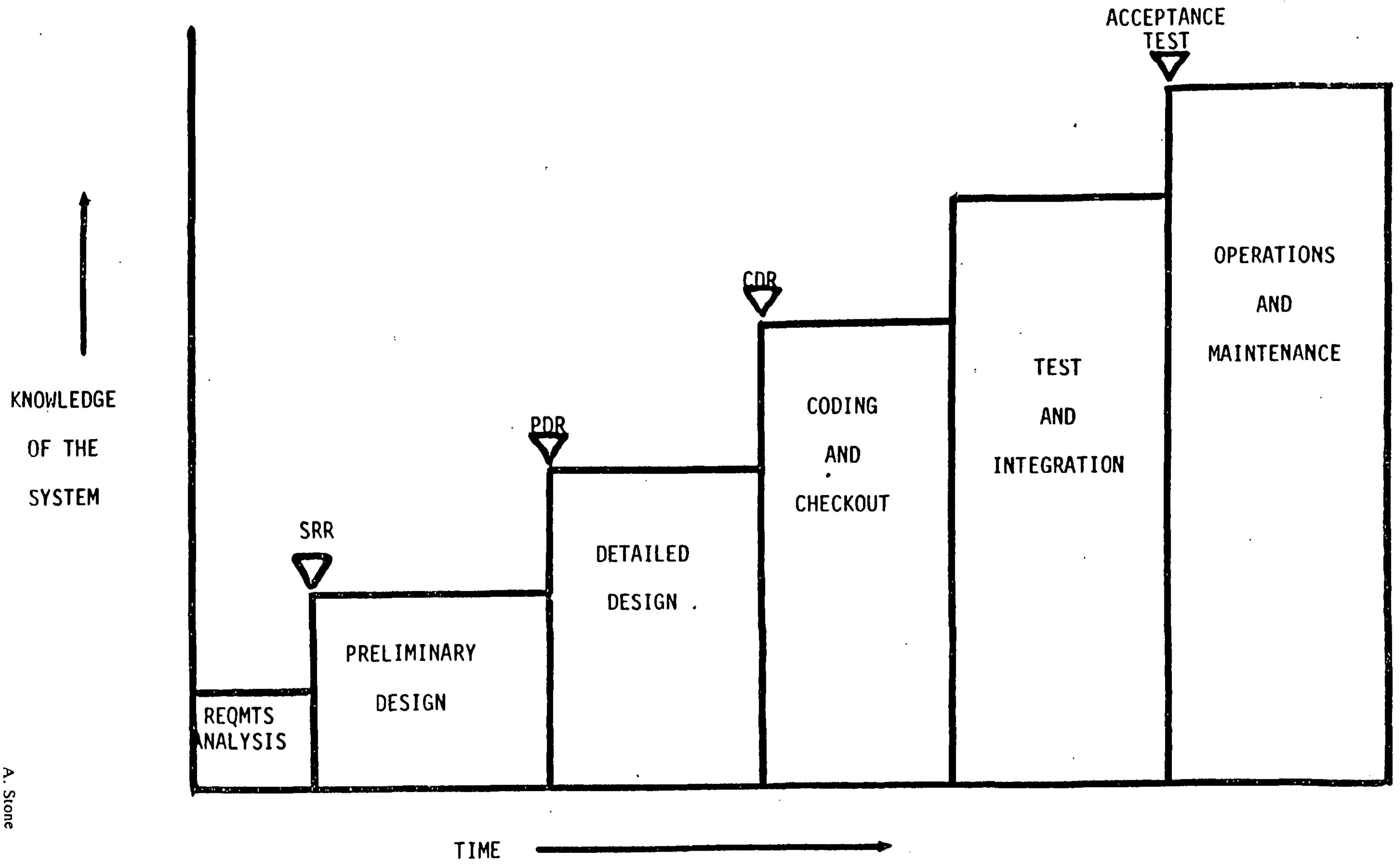


VERIFICATION PROBLEMS
DEFINITION OF SOFTWARE REQUIREMENTS
CODING PROBLEMS
SOFTWARE DESIGN
CODING
SOFTWARE DESIGN AND REQUIREMENTS PROBLEMS
CHECKOUT
SOFTWARE CHECKOUT PROBLEMS
INTEGRATION & VERIFICATION
FINAL DOCUMENTATION

1469-A

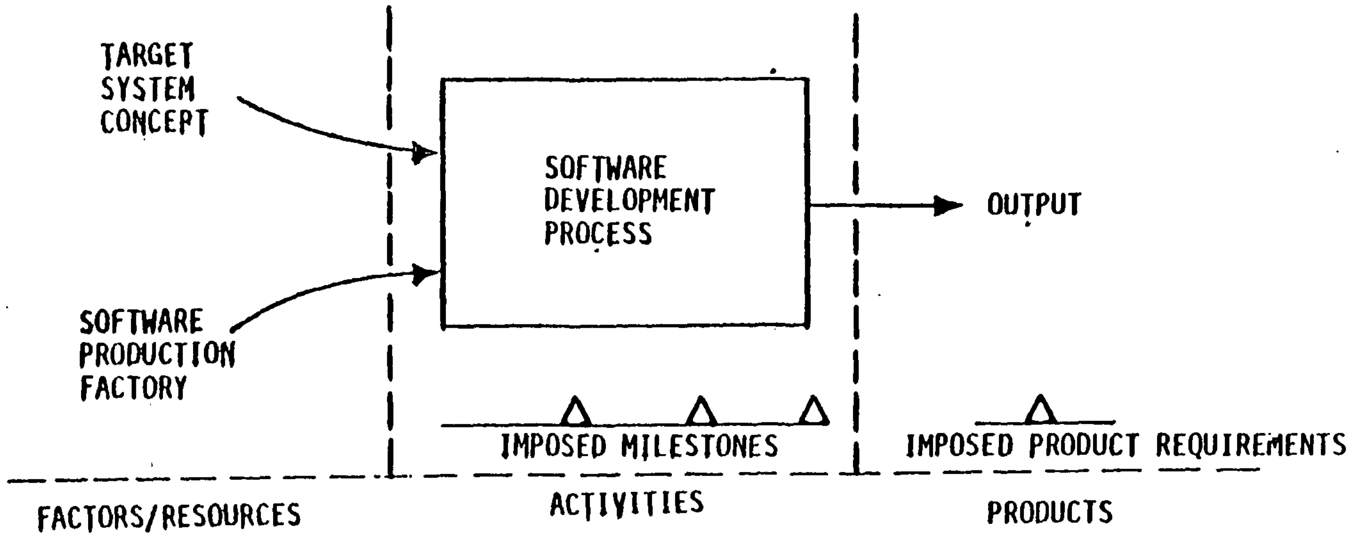
ISP

SOFTWARE DEVELOPMENT PROCESS MODEL



ISP

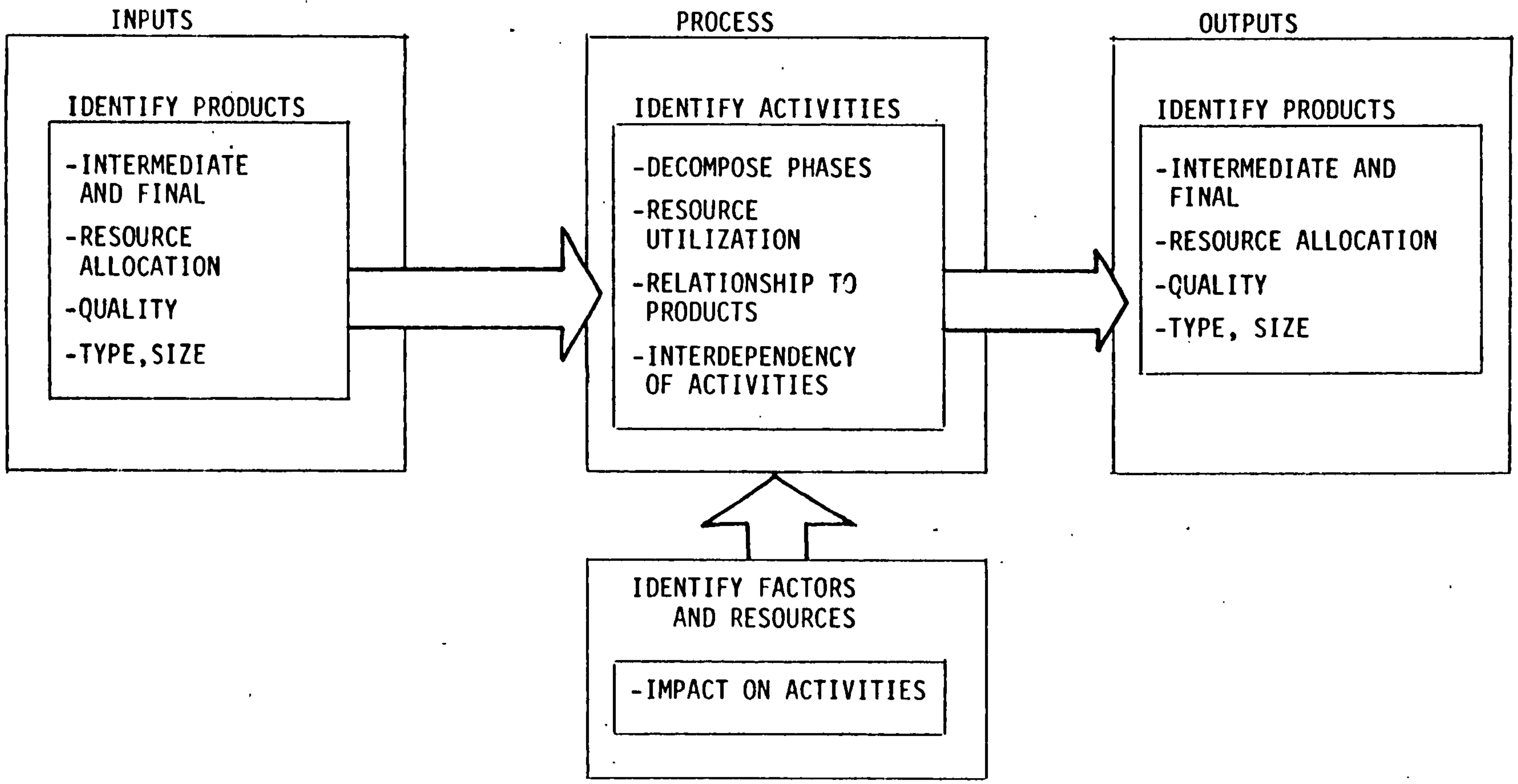
SOFTWARE DEVELOPMENT PROCESS CONCEPT


TARGET SYSTEM CONCEPT
SOFTWARE PRODUCTION FACTORY
SOFTWARE DEVELOPMENT PROCESS
OUTPUT
IMPOSED MILESTONES
IMPOSED PRODUCT REQUIREMENTS
FACTORS/RESOURCES
ACTIVITIES
PRODUCTS

1781A

DECOMPOSITION OF THE SOFTWARE DEVELOPMENT PROCESS

DECOMPOSITION METHODOLOGY



ISP

THE BASIC CONCEPTS OF SDPS

SYSTEM DEVELOPMENT PROGRESS IS REPRESENTED BY:

- THE MIX OF ONGOING DEVELOPMENT ACTIVITIES (ACTIVITY OR WORK BREAKDOWN STRUCTURE MODEL)
- THE EVOLUTION OF SYSTEM KNOWLEDGE IN THE FORM OF PRODUCTS (PRODUCT PROGRESSION MODEL)

HOW MUCH PROGRESS IS SHOWN BY:

- PERSON HOURS BY TYPE OF PERSON FOR EACH ACTIVITY
- LINES OF DOCUMENTATION OR CODE FOR PRODUCTS

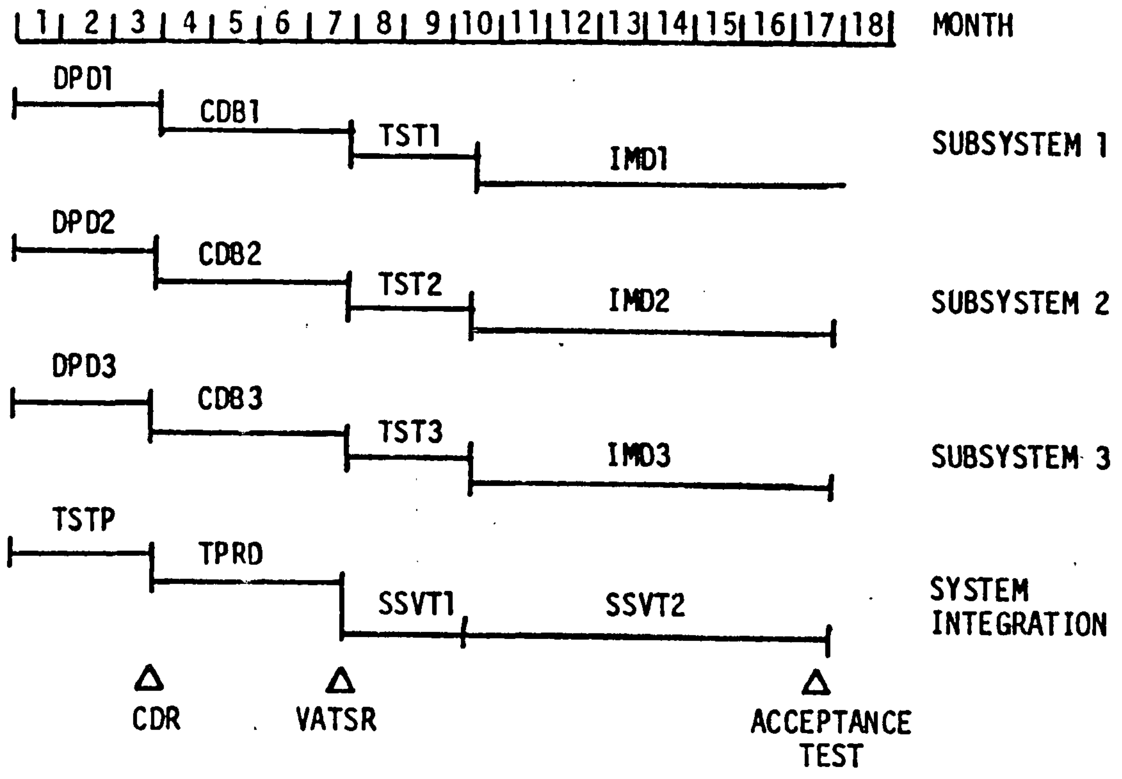
QUALITY OF THE EFFORT IS REFLECTED BY:

- PERSONNEL EXPERIENCE IN AN ACTIVITY
- QUALITY METRICS VALUES FOR EACH PRODUCT

ISP

MODEL UTILITY

- CHECKLIST
- PERT-COST
- PROCESS MODEL
- DETAILED PROCESS MODEL



	LEGEND
DPDi	DETAILED PROGRAM DESIGN
CDBi	CODING AND DEBUG
TSTi	SUBSYSTEM TEST
IMDi	INTEGRATION & MAINTENANCE SUPPORT
TSTP	TEST PLAN PREPARATION
TPRD	TEST PRUCEDURE DEVELOPMENT
SSVT1	SYSTEM TEST & INTEGRATION
SSVT1	
CDR	CRITICAL DESIGN REVIEW
VATSR	VALIDATION & ACCEPTANCE TEST SPECIFICATION REVIEW

ISP

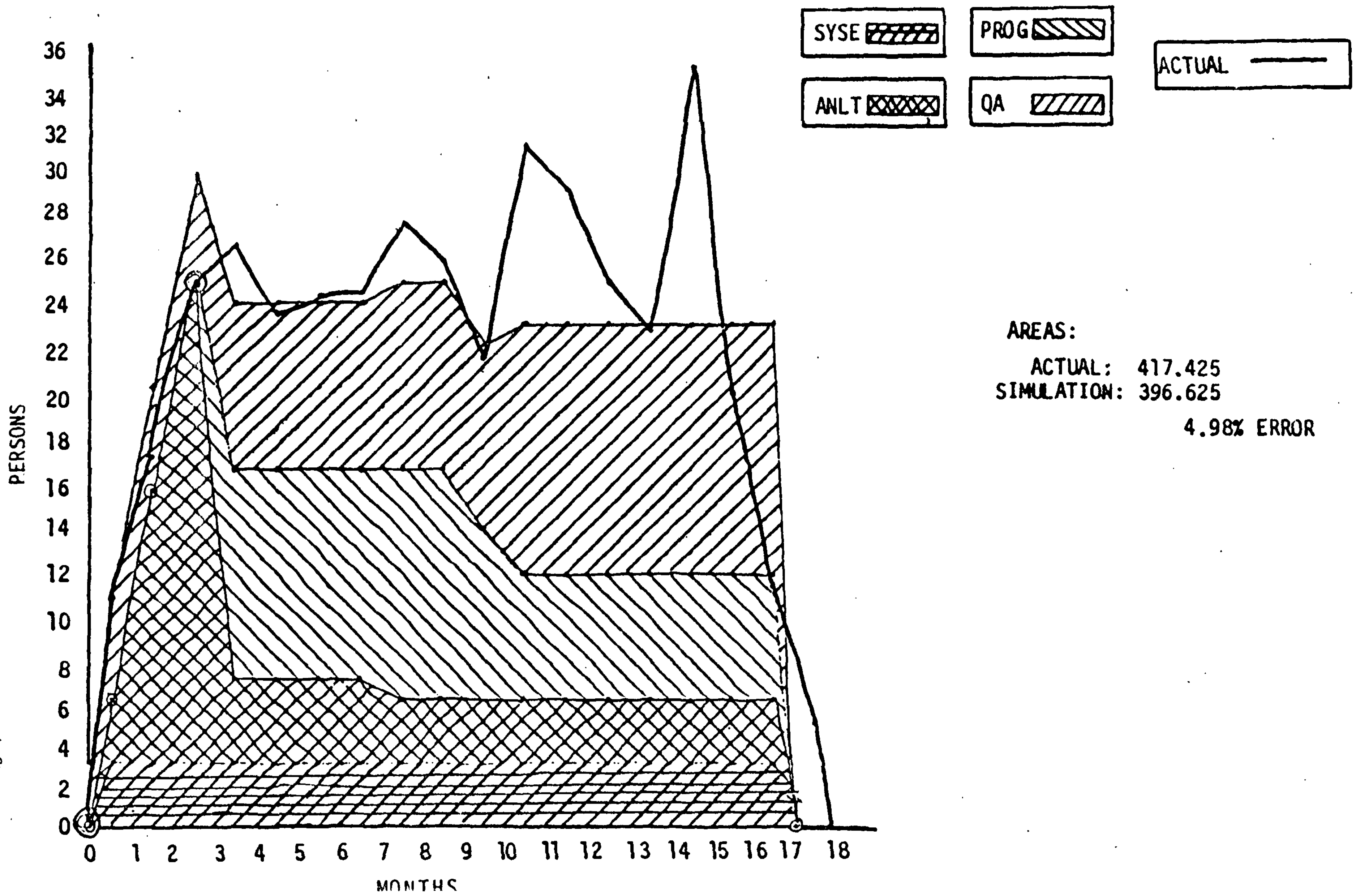
EXPERIMENT DESCRIPTION

PROCESS FLOW:

```
        05/29/79                   19.6930                                PAGE      1

                  PATH EXPRESSION PARSER          VERSION 1.5

           1     BEGIN SAMPLE
           2     ;
           3     ;    SDPS SAMPLE MODEL
           4     ;
           5     ;         BY    J.A. MCCALL
           6     ;               A.H. STONE
           7     ;
           8     ;         APRIL 1979
           9     ;
          10          MACRO-TABLE
          11             DESIGN = (DPD1:DPD2:DPD3)
          12             CODING = (CDB1:CDB2:CDB3)
          13             TEST   = (TST1:TST2:TST3)
          14             INTEG  = (IMD1:IMD2:IMD3)
          15          END-MACRO
          16     ;
          17     ;    BEGIN THE SIMULATION
          18     ;
          19          SIMULATE = (DESIGN:TSTP,         ; DESIGN PHASE
          20                      CODING:TPRD,         ; CODING PHASE
          21                      TEST:SSVT1,          ; TESTING PHASE
          22                      INTEG:SSVT2)         ; INTEGRATION PHASE
          23     ;
          24     END  ; SAMPLE

                24 LINES PROCESSED
                 0 NON-FATAL ERRORS
                 0 FATAL ERRORS

   --- PROCESSING COMPLETED
```

ISP

EXPERIMENT RESULTS



AREAS:

ACTUAL: 417.425
SIMULATION: 396.625
4.98% ERROR

CONCLUSIONS/FURTHER INVESTIGATIONS

CONCLUSIONS MATRIX

	ACCOMPLISHED	FUTURE
(1)	SIMULATION APPROACH	
	COMBINED ACTIVITY-PRODUCT MODEL FORMS CONCEPTUAL BASIS.	IDENTIFICATION OF SIMULATION VARIABLES, MODEL RULES, MODEL INPUTS AND OUTPUTS.
(2)	PROCESS DECOMPOSITION	
	ACTIVITY-PRODUCT NETWORK DEMONSTRATES PRACTICAL APPLICAITON.	DETAILED SPECIFICATION OF ACTIVITIES, PRODUCTS, FACTORS, AND RESOURCES.
(3)	SIMULATOR DEVELOPMENT	
	SIMULATOR PROTOTYPE DEMONSTRATES EXPERIMENTAL FEASIBILITY.	DATA COLLECTION TO SUPPORT FULL EXPERIMENT