A RAND NOTE

Toward a Comprehensive Environment for Computer Modeling, Simulation, and Analysis

Robert H. Anderson, Steven C. Bankes,
Paul K. Davis, H. Edward Hall,
Norman Z. Shapiro

RAND

The research described in this report was supported by RAND using its own research funds.

Published 1993 by RAND
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138
To obtain information about RAND studies or to order documents,
call Customer Service, (310) 393-0411, extension 6686

A RAND NOTE

N-3554-RC

Toward a Comprehensive Environment for Computer Modeling, Simulation, and Analysis

Robert H. Anderson, Steven C. Bankes,
Paul K. Davis, H. Edward Hall,
Norman Z. Shapiro

RAND

PREFACE

This Note is a tentative "think piece" describing the authors' personal views about a new type of computer "environment" that is both needed and possible for modeling and analysis. It also describes a relatively concrete plan for developing and testing prototype versions of such an environment at RAND, with the intention of later exporting successful tools and methods to other analytic organizations inside and outside government. Work consistent with the plan is now well under way, but it will take years to realize the vision. Further, many elements of the approach reflect judgments and hypotheses that will need to be revisited and iterated as we gain experience with prototypes. Although we developed it originally for internal purposes, we are publishing the Note now because of interest in the effort expressed by many colleagues in the scientific and analytic community, many of whom are concerned with similar issues.

Preparation of the Note was sponsored by RAND's Defense Planning and Analysis department using RAND's own funds. Comments are welcome and can be addressed to the authors by electronic mail via Internet (e.g., at Paul_Davis@rand.org or Robert_Anderson@rand.org).

SUMMARY

We argue here for a vision of a new and powerful computer "environment" for modeling and analysis, with "environment" being construed broadly to mean a union of hardware, software, facilities, conventions, procedures, links to external networks and data bases, and organizational mechanisms for teaching, sharing, and communicating. Such an environment will make modeling and analysis better, more flexible, and faster. This Note describes objectives and a strategy for developing a prototype version called RANDSIM. If successful, it would be the basis for subsequent work, including exportable tools.

Our overarching concept for organizing ideas on an environment is that we should seek to maximize useful computer support for *all* the identifiable activities in studies—activities ranging from review of past studies to group brainstorming, development of models and data bases, "exploratory" analysis amidst massive uncertainty, careful analysis of alternatives, and multimedia presentation of results. Currently, only a few of these activities are adequately supported by computers and information science methods. For example, modeling with current tools often leads to programs that are difficult or expensive to comprehend and adapt, much less to pass on effectively to sponsors and other research organizations, or to use effectively in distributed simulation networks. Great strides are possible in both the scope and quality of computer assistance.

Our objectives consist, as mentioned above, of supporting *all* phases of the modeling and analysis process. As part of this, however, we have the specific objective of improving the state of the art in high-level languages. We seek language capabilities (including graphical tools) that will allow analysts, not just programmers, to design, understand, and modify major features of programs, and that will provide significant explanation capabilities. Further, we wish to emphasize a highly interactive style of modeling and analysis that encourages and facilitates the prototyping and exploratory analysis that we believe is important to innovation and top-quality modeling and analysis under uncertainty. And we seek hypermedia methods that will facilitate and improve documentation.

Our strategy for accomplishing the various goals is based on a number of principles that reflect subjective judgments in the context of the particular circumstances we perceive at RAND. Thus, while we hope the fruits of our development will be broadly applicable, our approach is organization specific. The objectives we developed in 1991 were and remain as follows:

- *Buy vs. Build.* Exploit vigorously the new and powerful products available commercially or without cost in the public domain. At the same time, retain control over the central tools and techniques most critical to achieving an *integrated* and adequately ambitious high-level environment. This means, in particular, retaining control over the central development language and related tools.
- *Software Reusability.* Encourage increased interoperability and modularity generally, which will require organizational changes with at least some centralized programming standards and registration of modules and, when appropriate, consistency with emerging community standards (e.g., those for the DoD's Distributed Interactive Simulation [DIS] efforts).
- *Diverse Users.* Design the environment for researchers working not only on military problems but also on social-policy and economic problems. Stimulate domestic-research "demand" by working closely with such researchers and encouraging experiments using the emerging tools.
- *Model Families as a Goal.* Seek scientific and practical advances in developing internally integrated hierarchical families of models, which make it possible for analysts to work effectively and consistently at different levels of detail (resolution).
- *Highly Interactive Modeling and Analysis.* By pursuing work on appropriate high-level languages and related graphical tools, seek major advances in highly interactive modeling and analysis.

Our proposed strategy has many elements designed with these principles in mind. In particular, the strategy includes the following aspects:

- *Commercial Tools for Pre- and Post-Processing (and Some Modeling).* Exploit personal computers (Macintoshes and IBM PCs) and related commercial tools such as Excel®, Word®, MacDraw®, PowerPoint®, HyperCard®, MacFlow®, iThink®, and 4th Dimension® rather than attempt to reproduce their functionality. Similarly, RAND should not develop its own data-base management system (DBMS), network facilities, or systems for communicating with the outside world.

- *A High-Level Central Language*. Develop a new high-level programming language (to be called Anabel) based on extending the RAND-ABEL®[1] language to include object orientation and many other features.[2] This will build on the ubiquitous C/UNIX® programming environment and will make *possible* the coherent integration of tools and other features of an environment while allowing RAND to pursue unique features related to model comprehensibility, hierarchical explanation capabilities, flexibility, interactivity, and ability to do exploratory analysis. It will also improve prospects for developing integrated families of models. Anabel will be an analyst-friendly language and a candidate for use in many of RAND's projects, which could increase commonality. We believe the language will prove popular and will be requested and used elsewhere as well.
- *Assuring Flexibility*. To reduce risks and maximize flexibility, Anabel development should neither start from a blank slate nor stand alone. In particular, (a) the Anabel language should translate automatically into C and C++, thereby assuring portability of this immediate code; (b) the entire development should build on the substantial and documented base of the RAND-ABEL language; (c) options should be maintained at all stages for falling back to the use of features of C and C++; (d) development should emphasize the requirement to link programs written in a variety of languages (including FORTRAN, Ada, and MODSIM®) since such diversity of languages is and will continue to be a fixture of research; and (e) the Anabel development should be phased with intermediate deliverables and milestones that can be assessed. The premises of Anabel development should be reviewed at yearly intervals.
- *Model Families*. Use the opportunity of current RAND work with such combat models and war games as RSAS, TLC/NLC, Janus, and Brawler to develop *internally integrated hierarchies* of combat models covering a wide range of resolutions. This will contribute to RAND's national security analysis and provide generic experience on cross-resolution work critical to interoperability on, for example, DIS networks.

[1]RAND-ABEL is a trademark of RAND.

[2]This decision was both difficult and controversial, but we concluded that none of the available languages was appropriate for our purposes. In some cases language source code was not available; in other cases the languages are not designed for high-level comprehensibility; in still other cases the language was highly specialized and not readily compatible with C/UNIX systems; and so on. Regardless of its potential value for large-scale DoD projects or embedded software, Ada was simply not competitive for this type of research.

- *Model and Data-Base Management.* Build on ongoing work in other projects and in RAND's Military Operations Simulation Facility (MOSF) to provide tools for configuration control, use of intelligent data-base techniques, model-specific data-base adaptations and manipulations, and greatly increased sharing of both data and models across projects.
- *Standards, Support, and Testing.* To assure continued maintenance and support, develop early documentation for Anabel and key environment tools. Plan on using RAND support operations, notably the MOSF and Information Sciences Laboratory (ISL), to assume some of the burdens associated with such matters with use of standards and conventions and with increased emphasis on verification and validation (V&V). Explore organizational mechanisms to provide education and training.
- *Long-Term Support and Exploitation.* Explore collaboration with commercial vendors when development is far enough along to make that useful. It may be desirable for a government sponsor to subsidize a support contractor when the time comes, assuming the environment tools come to be used by the government and its contractors.
- *Organizational Measures.* In the second year, after laying a technical foundation but while continuing Anabel development, begin aggressively to work with a broad range of potential military and nonmilitary users with applications involving a variety of model types such as discrete-event and time-stepped simulation, optimization, cost-effectiveness assessments, and statistical analysis. This will involve (a) creating appropriate discussion forums within and outside RAND; (b) seeking early testers with ongoing or prospective project work suitable to the new tools (including testers who need to use languages other than Anabel, at least in part); (c) experimenting with important nonlanguage aspects of the environment, such as distributed model-supported videoconferencing;
(d) experimenting with model and data-base management tools being developed in other projects; (e) developing organizational mechanisms for centralizing documentation, registering models and data bases, and providing effective training as it is needed; and (f) keeping abreast of and contributing to community developments through participation in professional organizations and other efforts.
- *Reviewing and Iterating Precepts.* Also in the second year, conduct tests to strengthen the basis for judgments about the relative merits of different language

features and tools for designing, understanding, and working with models. In connection with other ongoing projects, conduct experiments in "exploratory modeling and analysis," prototype efforts that emphasize dealing effectively with uncertainty. Reflect results in evolving plans for the modeling and analysis environment.

Where are we as of November 1992? Work on the full environment concept is still at a fairly early stage, but there have been important achievements in the last year since the original version of this Note was first written as an internal proposal. We have demonstrated effective and efficient coupling of Sun workstations and desktop Macintosh microcomputers. We have experimented successfully with hypertext and hypermedia for documenting models on-line and have developed operational software tools embodying these concepts. We are well advanced in design work and related documentation for the Anabel language and have completed versions with objects and lexical (static) inheritance. A first version was completed early in 1992 and is now in use on a major project. Work on class libraries is now under way. Cross-divisional plans call for submodels and tools developed in the RAND Strategy Assessment System (RSAS) and Theater-Level Campaign Model/ Nonlinear Combat Toolkit (TLC/NLC) theater-combat-modeling efforts to migrate into the emerging environment. This migration will include the MapView object-oriented map-graphics system, which is coupled to the RAND-CAGIS (RAND's Cartographic Analysis and Geographic Information System). Map-based presentations should be of substantial value to both regional specialists and domestic researchers as well as military analysts. Further, we believe the graphics system underlying MapView can be used for graphics-assisted model design.

We have yet to go beyond initial concepts on many other aspects of the envisioned environment. For example, RAND is only now beginning to use videoconferencing, and its use of external data bases available on networks is modest. The Anabel project is only beginning to pursue graphical design aids for Anabel, and we have not yet begun integrating work on the language and related tools with the tools for model and data-base management that will be very important in establishing a good analyst environment. Nonetheless, the first year has been very encouraging. If adequate funding can be obtained and maintained, which remains problematic, we are optimistic about prospects. In many instances, however, we are plunging into uncharted waters and there will need to be a great deal of exploration and iterative refinement.

ACKNOWLEDGMENTS

We appreciate informal reviews and other comments by many RAND colleagues, particularly Willis Ware, Iris Kameny, and Stephen M. Drezner, and a formal review by Richard J. Hillestad. All of them raised tough questions and made good suggestions. They may or may not agree with our conclusions, but the research effort described here will be better in any case.

CONTENTS

FIGURES

TABLE

1. INTRODUCTION

OBJECTIVES

This Note, which is somewhat of a think piece, has several objectives: (a) to convey an image of an idealized computer "environment" for modeling and analysis, (b) to sketch a development plan we first presented in 1991, and (c) to describe the first year's progress.

BACKGROUND

This work originated in 1990 as a planning effort. RAND is in the business of studies and analysis, which are changing rapidly with world politics and technological progress. Consistent with its institutional history, RAND is pursuing ways to exploit the emerging capabilities of information science to improve methods of analysis. This Note presents our motivations and describes one plan for doing so, a plan envisioning a new and highly ambitious overall computer "environment" consisting of hardware, software, facilities, conventions, procedures, links to external networks and data bases, and even organizational processes for teaching, sharing, and otherwise communicating about models. The ideas we present are speculative; they represent the authors' effort to provide a vision and plan to guide exploratory developments. Some of our ideas and judgments will prove wise; others will not. Even among our RAND colleagues there is a diversity of ideas about what is needed and what is most worth pursuing. Nonetheless, one must start somewhere and this Note describes our approach.

STRUCTURE OF NOTE

The Note is structured as follows: Section 2 describes shortfalls in current capabilities for modeling and analysis. This is the "demand-pull" part of the Note but goes well beyond what most practitioners of modeling and analysis would currently think of as requirements. Section 3 represents "technology push" by describing the opportunities we see as most relevant. Section 4 presents a strategy (first developed in 1991 but still accurate) for developing a prototype environment (which we will call RANDSIM). Section 5 describes where things stand and what is planned for the next year or so. It also describes the state of actual development as of November 1992. Appendix A summarizes our reasoning and judgments on key choices (e.g., the choice of our primary computer language). Appendix B gives more technical detail on the goals of the development work that is currently being supported. Appendix C describes some of our aspirations for providing management tools for models, data, and the run-time environment.

2. SHORTFALLS IN CURRENT MODELING AND ANALYSIS ENVIRONMENTS

A FRAMEWORK FOR IDENTIFYING SHORTFALLS

All modelers or analysts can quickly identify features they wish were available in their computer environment. It is useful, however, to have a conceptual framework for identifying such features systematically. Further, such a framework should be general enough to help us think about modelers and analysts working ideally in what is sometimes referred to as a world with "ultimate" computing power and pervasive communications.

One place to begin is with a depiction of the classic study process (for studies making significant use of models). There are many depictions, but Figure 1 is appropriate for our purposes. The process starts with tasking from policymakers or managers recognizing the needs of policymakers. The study begins with brainstorming to determine what the "real issues" are, which often are rather different from what was envisioned at the time of initial tasking. The brainstorming is also crucial in developing a general sense of how to proceed: what is important, what is doable, what depends on what, and so on. This phase may also produce a variety of hypotheses and options that should be investigated, tested, and compared. The next phase is developing a study plan, which is arguably the most important part of the entire study, because crucial decisions are made about scope, assumptions, uncertainty analyses, measures of effectiveness, and so on. The study plan may require building or adapting models and data bases. It specifies cases that will be examined by using the models. And so on. The overall process is highly iterative.

The individual processes in Figure 1 are shaded and annotated. The darker the shading, the more we already use and depend on computers for the function in question. As Figure 1 suggests, computers are still used primarily for running models. There is only modest support for building them and analyzing the results.

Our views on this have been heavily influenced by observing the influence and acceptance of the "Macintosh environment" and our experience developing and evolving a large analytic war gaming system (the RAND Strategy Assessment System) during a period in which it was impossible to completely avoid mixing languages and interface paradigms and in which competitive approaches were emerging rapidly. While "superstars" could handle the diversity, most users found it very burdensome, which has led us over time to move toward a more integrated framework.

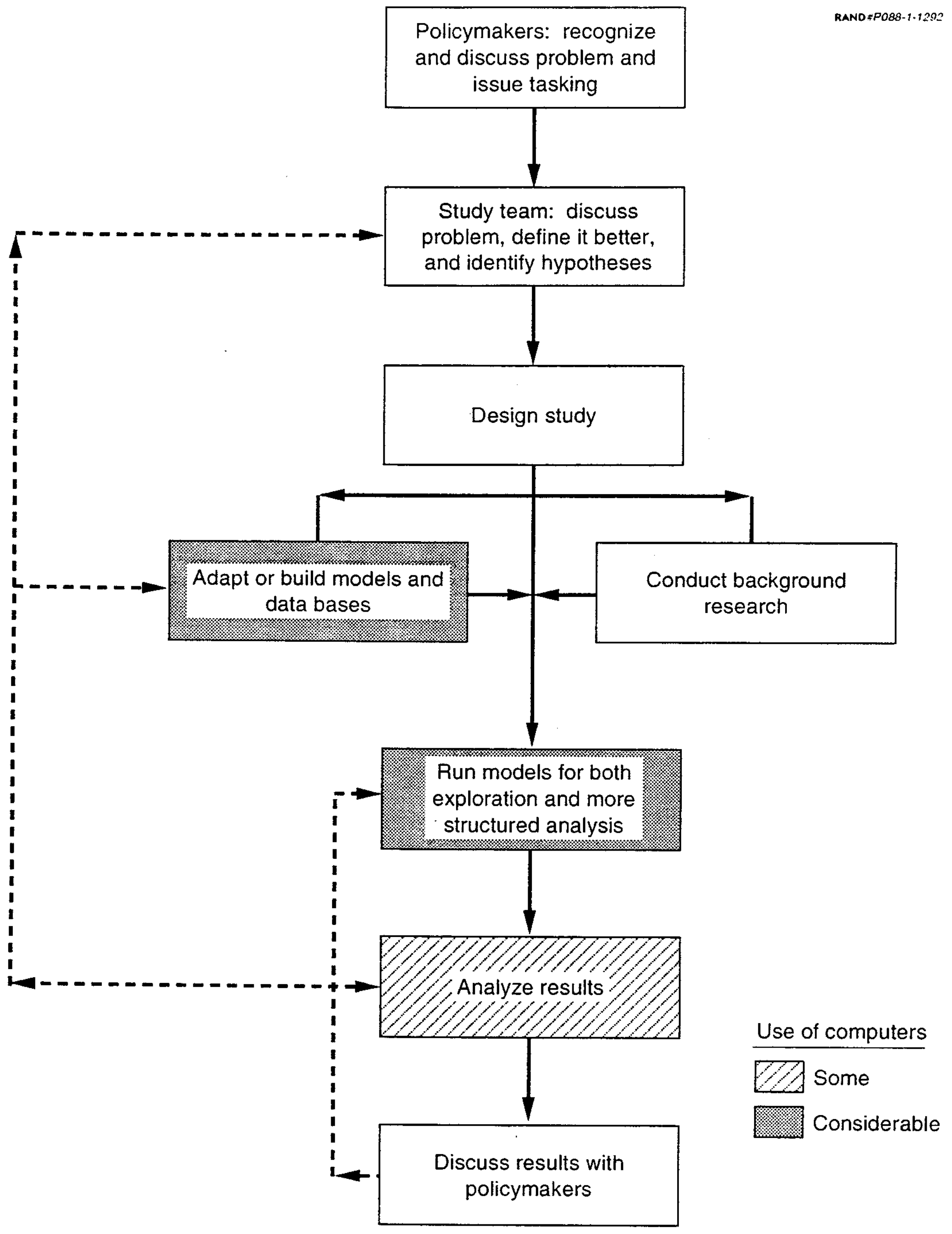
RAND#P088-1-1292
Policymakers: recognize and discuss problem and issue tasking
Study team: discuss problem, define it better, and identify hypotheses
Design study
Adapt or build models and data bases
Conduct background research
Run models for both exploration and more structured analysis
Analyze results
Discuss results with policymakers
Use of computers
Some
Considerable

Figure 1—A Schematic View of the Study Process

We now expand upon this image in Figure 2, using bubbles to show how some of the standard processes actually include quite a number of subordinate processes that have often been taken for granted but that are important and could benefit from extensive computer support (e.g., searching for and reviewing past studies, assembling a multidisciplinary team, reviewing and perhaps testing existing models, tailoring products to particular sponsors, and so on). The exclamation points !, !!, or !!! indicate subjectively the degree of pain and expense involved in using computers to the extent they are used. For example, computers are certainly used to build models (or at least the programs implementing them), but it is currently a tedious and expensive process because the tools are not yet well enough developed and available. It is striking how computer technology has been only minimally used, or even thought of, as a standard way of helping us do our business. To be sure, one could find counterexamples in each bubble, examples in which workers currently use computers well in enhancing their work. However, the potential is far greater than the reality. Further, even for activities in shaded bubbles, there is the potential for *much* more effective use of computers than in the past (e.g., reviewing and adapting existing models could be much easier merely if the models were programmed in more comprehensible languages).

We regard each of the activities represented in Figure 2 as something that should take place within a total environment for modeling, simulation, and analysis. In particular, Figure 2 includes many activities not often well supported, either by computer aids or otherwise:

- Searching data bases and the literature for prior relevant studies; reviewing such studies efficiently through a variety of media (e.g., written reports, videotapes, "canned simulations," and data bases of results).
- Assembling, in some sense, an interdisciplinary team of individuals with appropriate skills and diversity of both knowledge and viewpoint; brainstorming with such a team and perhaps conducting insight-generating games.
- Acquiring and reviewing existing models. Can they be adapted? Are portions reusable in this analysis?

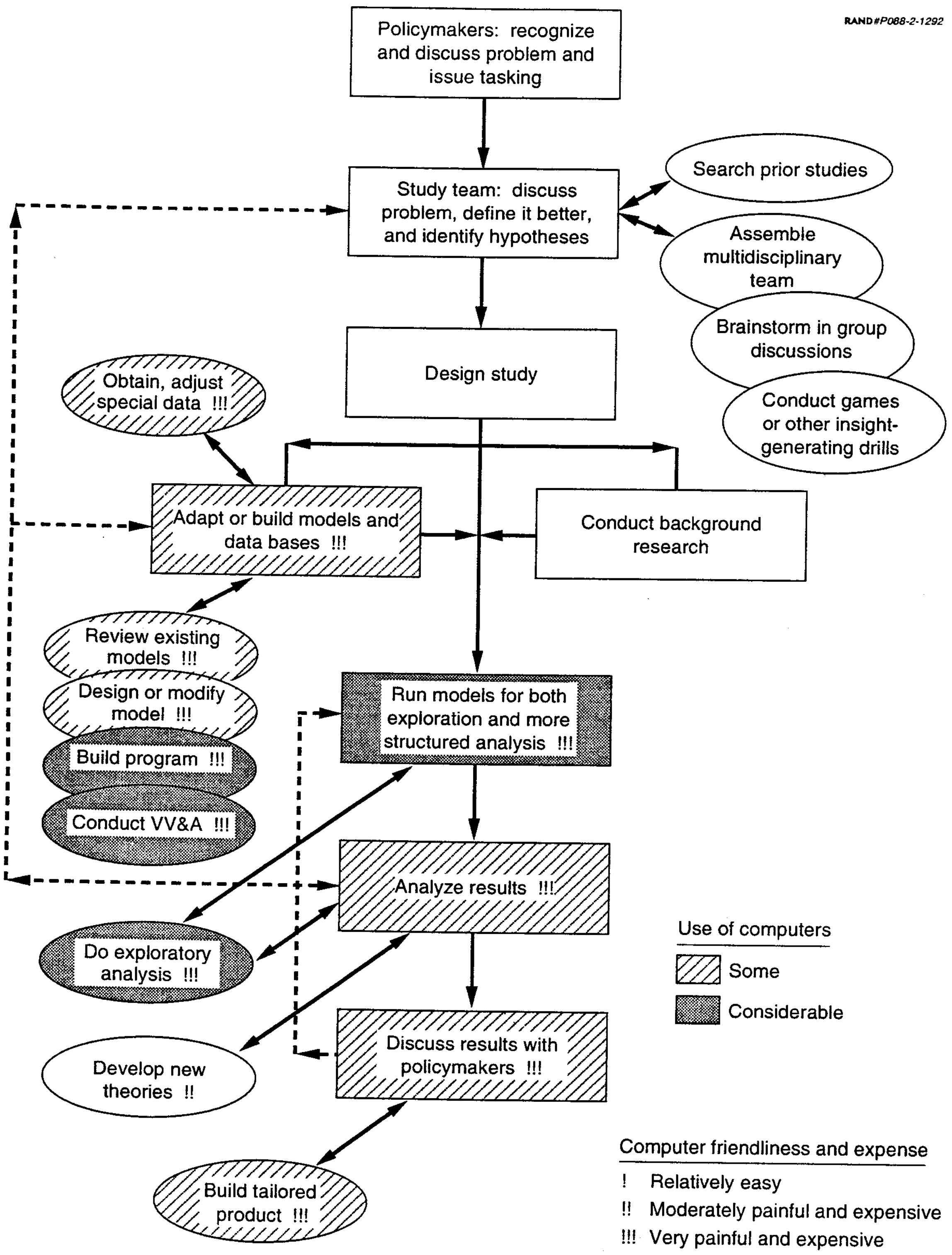
RAND#P088-2-1292
Policymakers: recognize and discuss problem and issue tasking
Study team: discuss problem, define it better, and identify hypotheses
Search prior studies
Assemble multidisciplinary team
Brainstorm in group discussions
Conduct games or other insight-generating drills
Design study
Obtain, adjust special data !!!
Adapt or build models and data bases !!!
Conduct background research
Review existing models !!!
Design or modify model !!!
Build program !!!
Conduct VV&A !!!
Run models for both exploration and more structured analysis !!!
Analyze results !!!
Do exploratory analysis !!!
Develop new theories !!
Discuss results with policymakers !!!
Build tailored product !!!
Use of computers
Some
Considerable
Computer friendliness and expense
! Relatively easy
!! Moderately painful and expensive
!!! Very painful and expensive

Figure 2—An Expanded Concept of the Study Process (shading indicates current degree of computer support)

- Building a tailored product from all of the existing resources that have been found plus any programming and developmental work that has been required. The "product" includes documents resulting from the analysis, briefings, demonstrations of the model, videotaped documentation of the model or a resulting briefing, and any other outputs deemed important to the client.
- Developing an analysis plan.
- Designing, building, and documenting any models developed (and some that were previously developed but not adequately tested and documented).
- Conducting verification, validation, and accreditation (VV&A) for the models.
- Obtaining special data needed, either from existing data bases worldwide or as derived from other data that are available.
- Developing new theories (e.g., new models for describing phenomena based in part on insights gained from exploratory modeling, exploratory analysis, and validation efforts).
- Comparing results across runs and models.
- Connecting to other models or man-in-the-loop simulations on a network such as SIMNET (Miller, 1992).

We believe that as part of the prototype environment (RANDSIM), all of the above activities, plus the ones indicated as already computer-supported in Figure 2, should be supported with excellent aids uniformly throughout an organization—in RAND's case, its division for social policy research as well as its divisions for defense studies.

Figure 3 represents our view of the extent to which computer technology could affect our work in the relatively near future (two to five years). The truth of this may even be self-evident since we are all aware of the revolution that has recently brought us powerful personal computers, networks, data retrieval programs, and so on.

What computer-based tools and techniques might provide relevant aids to the many activities shown in Figure 3? Table 1 is an initial attempt to relate aids to activities. We are not promising that all such aids will be developed or that we have listed all the most promising ones. Rather, our intent is to indicate the wealth of tools and techniques that can be brought to bear on many of these activities—but tools that must form a *coherent* and *mutually supportive* collection.

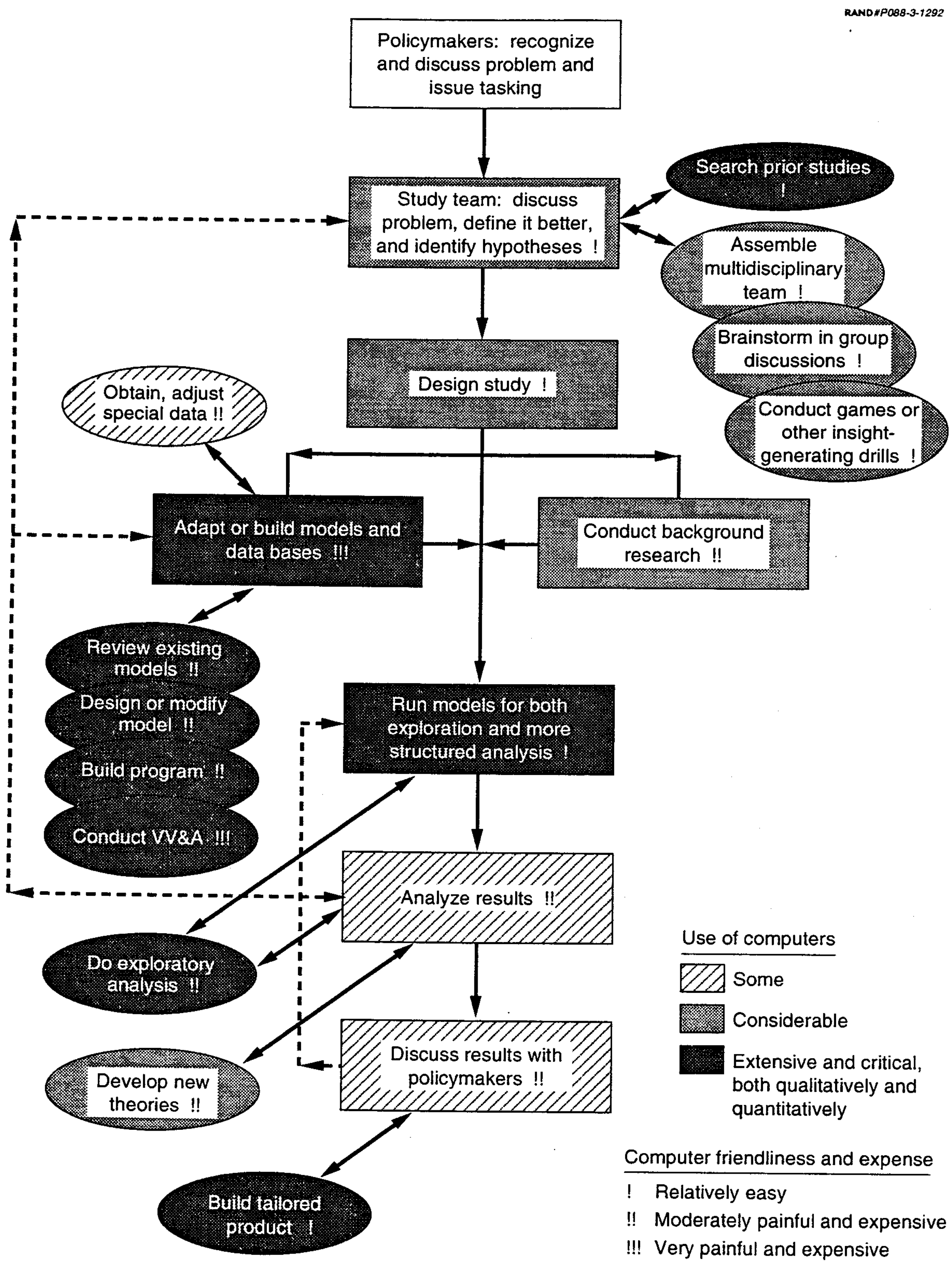
RAND#P088-3-1292
Policymakers: recognize and discuss problem and issue tasking
Study team: discuss problem, define it better, and identify hypotheses !
Search prior studies !
Assemble multidisciplinary team !
Brainstorm in group discussions !
Conduct games or other insight-generating drills !
Design study !
Obtain, adjust special data !!
Adapt or build models and data bases !!!
Conduct background research !!
Review existing models !!
Design or modify model !!
Build program !!
Conduct VV&A !!!
Run models for both exploration and more structured analysis !
Analyze results !!
Do exploratory analysis !!
Develop new theories !!
Discuss results with policymakers !!
Build tailored product !
Use of computers
Some
Considerable
Extensive and critical, both qualitatively and quantitatively
Computer friendliness and expense
! Relatively easy
!! Moderately painful and expensive
!!! Very painful and expensive

Figure 3—Potential Computer Support for the Study Process

Table 1

First-Cut Associations of Activities and Aids

Analysis-Related Activity	Potential Aids
Searching for prior studies and data bases	Electronic search via networks; browsing tools.
Assembling a multidisciplinary team	Videoconferencing and software for collaborative work (groupware) such as electronic mail, shared electronic notebooks, etc. (much available from homes).
Brainstorming to define the problem	Group-discussion aids; visualization methods; computer-assisted videoconferencing with distributed modeling (including simulation and gaming).
Reviewing existing models	Electronic search; browsing tools for new documentation methods; standardized data dictionaries; semiautomated model testing against standardized "scenarios"; high-level languages.
Building tailored products	Desktop publishing tied intimately to personal computers tied intimately to networked computers, printers, videotaping machinery, etc. There is a possibility of directly linking models with these "publishing" applications.
Developing an experimental plan	Case-development aids; retrieval of past study results; retrieval of data on policymaker attitudes, biases, and concerns.
Obtaining special data	Specialized retrieval, reformatting, and processing of data from networked data bases and other sources (e.g., speeches, proceedings, and journals).
Constructing models (design, build, test, and document, often building on existing models)	Graphical design aids; design conventions; design theories; high-level languages; modular design; standards promoting reusability; semiautomated verification testing; partially standardized validation; special methods to enhance ability for problem exploration.

Table 1—continued

Conducting verification, validation, and accreditation on models	Language features to assist verification and top-down visually oriented design and documentation with hypermedia features. Library functions to assist verification testing. Network links to other organizations to facilitate model comparison as part of validation and accreditation.
Connecting to other models and simulations	Networks and protocols (e.g., SIMNET and Distributed Interactive Simulation [DIS]), quickly comprehensible documentation of candidate models, VV&A tools, tools for adapting models for connection to dissimilar models.
Running models and doing so systematically as part of an analysis	Networking and parallel processing with user-friendly setup tools and semiautomated problem solving during long runs. Training methods.
Analyzing results	Visualization techniques of all kinds. Training methods.
Comparing results across runs and models	Semiautomated postprocessing coupled with visualization techniques and automated comparisons with standard cases; user-friendly case specification and version control; dictionaries and repositories related to previous results. Analyst tools to help design exploratory analysis and final, convergent analysis.
Creating and tailoring briefings, demonstrations, and other outputs	Portable equipment for multimedia presentations and simulation; distributed simulation; desktop multimedia publishing.

SHORTFALLS AND PROBLEMS BEING ADDRESSED

The problems being addressed in this Note can now be described in terms of the overall environment indicated in Figures 2–3. *The central problem we see is the lack of adequate and unified support for the activities that are unshaded or only lightly shaded in Figure 2*, with the result that these activities are not performed as thoroughly or well as they should be, are not performed with consistency across projects, and often leave little useful documentation in their wake. The unshaded activities of Figures 1–2 are as vital to the overall process as model building, programming, and analysis of run results but receive less attention because of the primitive tools and aids available for them.[3]

[3]There are many other problems associated with modeling and analysis, of course. These include the data-base problems we discuss only briefly here, the challenge of teaching the substance of

One reason for some of the shortfalls (unshaded bubbles) is a lack of standardization that has resulted from researchers, over time, optimizing their particular efforts along many different dimensions (e.g., speed, readability of code, ease of programming, graphics front-ends permitting interaction with a running simulation, convenience in using a particular data base). As a result, *a plethora of programming languages and computer aids is being used for modeling at RAND and elsewhere* with tools such as MODSIM, Extend®, Stella®, RAND-ABEL, C, FORTRAN, Lisp, SAS, Basic , Excel, LOTUS®, and iThink. There is a similar diversity of data bases and data-base formats. By no means is this all bad. To the contrary, most modelers and analysts recognize that different tools are useful for different problems and prefer the diversity of tools to having to use a single allegedly "universal tool." The different tools also encourage alternative approaches to the same problem, which can enhance innovation and broaden perspectives. Nonetheless, because of this diversity, it is difficult to build a pool of modelers, programmers, and analysts that are fungible among projects and to build a common culture of modeling and analysis with appropriate training, seminars, workshops, and documentation—or even possible sharing of subroutines or code modules. Further, *comprehensibility, user friendliness, and related issues of efficiency and interoperability suffer heavily when the diversity of methods is too great.*[4] Although it will always be necessary to work with a variety of languages and other tools, the current situation is extreme—within RAND and in the community at large.

We want to address a number of other problems as well, some of them subsidiary to the above issues but nonetheless important for a variety of reasons that include continuing scientific interests (e.g., in promoting high-level languages and exploratory analysis). They are as follows:[5]

- Current programming languages result in programs that are often not *transparent* (understandable at the line-by-line level) and very seldom *comprehensible* as a whole. It is often difficult if not impossible for analysts,

---

particular models to both senior and junior analysts on a continuing but unscheduled basis, and importing and using models developed elsewhere.

[4]Our views on this have been heavily influenced by observing the influence and acceptance of the "Macintosh environment" and our experience developing and evolving a large analytic war gaming system (the RAND Strategy Assessment System) during a period in which it was impossible to completely avoid mixing languages and interface paradigms and in which competitive approaches were emerging rapidly. While "superstars" could handle the diversity, most users found it very burdensome, which has led us over time to move toward a more integrated framework.

[5]See also DoD (1992), which includes appendices identifying shortfalls in both modeling methodology and technology. The document focuses primarily on issues relevant to distributed simulation.

much less clients and sponsors, to study and modify even moderately complex models and simulations.[6] To some extent these difficulties are inherent in the complexity of the problems themselves, but we believe great strides can be made to improve both transparency and comprehensibility. This will require better and higher-level languages, including, importantly, languages that exploit graphical tools.[7]

- Prototyping is typically difficult. It is therefore difficult to perform "exploratory modeling" (Bankes, 1992b) as a means of making detailed design decisions about the final model or to explore alternate structures or analytic approaches toward a model. There are great opportunities here, especially those exploiting graphical design methods.[8]
- Many models are locked into particular hardware configurations, especially if they use graphical input/output on a display terminal. Use of modeling tools should result in code and user interfaces that are as portable as possible to a variety of workstations and computers (e.g., Macintosh, Sun, IBM PC compatibles).
- Even models developed within a single organization seldom link well together or fragment into modules that can be developed in parallel. Even more serious, since most models used by analysis organizations are and will be developed elsewhere, there are few tools and no overall environment to assist importing, reviewing, adapting, and linking externally developed models. Some of this is changing in the DoD world with the emergence of Distributed Interactive Simulation (DIS) networks, but development is still at a fairly primitive stage overall, despite significant advances at MITRE (see Weatherly, Seidel, and Weissman, 1991) and Aerospace Corporation (Landauer and Bellman, 1992) and by RAND colleague Jed Marti with the Seamless Model Integration (SEMINT) system.

[6]This creates serious difficulties also for VV&A as well as for interoperability and reusability. See Davis (1992a), which reports work done for the Office of the Secretary of Defense (OSD) Defense Modeling and Simulation Office.

[7]More graphics tools are becoming available, including tools for constructing data-flow diagrams and other characterizations of models. Only in a few instances, however, are the diagrams hard-linked to the code itself. Exceptions include iThink and some of the tools used in expert systems.

[8]As discussed in the appendices of Davis (1992a), high-level languages (including graphical tools) should revolutionize the way modeling is accomplished and should eliminate some of the troublesome distinctions between "models" and "implementing programs" that have caused so much trouble over the years.

- In a related vein, connecting models that cross levels of resolution (or designing variable-resolution models) is neither well understood theoretically nor supported well by software tools (e.g., tools making it easier to see data flow and variable hierarchies, change variable names, or find substantive definitions of variables if such documentation is even available).[9]
- Documentation of models very often lags behind their development and appears mainly to be an afterthought. It is often not effective even when it exists or becomes ineffective as it rapidly becomes outdated. New methods are needed that can be applied as a model is being developed, without distracting from the development process.[10] Further, documentation needs to be understandable *quickly*.[11]
- It is difficult to design and implement comprehensible models dealing with "soft" issues, including behavioral factors and judgments, without using highly specialized and often inefficient artificial intelligence languages.[12]
- Collecting, manipulating, cleaning, and otherwise testing, using, and reusing complex data bases is very costly and often unpleasant, despite advances in database management systems (DBMS). Object-oriented DBMS are not highly developed as yet and more common DBMS (e.g., INGRES) often do not fit well with advanced models (e.g., object-oriented models). There are few tools available for automated or semiautomated verification of data bases, much less

[9]For recent work on variable-resolution modeling and connection of models with different resolutions, see Davis (1992b), Hillestad and Juncosa (forthcoming), and Hillestad, Owen, and Blumenthal (forthcoming).

[10]Some of the ideas we are pursuing on documentation stem from an exploratory project accomplished by colleague John Clark, now at the University of Colorado. Clark emphasized moving away from linear hard-copy documentation and toward hypertext concepts that recognize the need to shift routinely from one perspective to another and to delve efficiently into examples from time to time. Another reality is that most modelers and programmers work primarily at a workstation; having to move to hard-copy documentation is a distraction.

[11]This point was emphasized by one of us (Davis) in a DIS conference held by the Military Operations Research Society in September 1992. If DIS is to be successful, it will surely be necessary for users to be able to understand models developed by others and running at other physical locations. This, in turn, will require efficient documentation, preferably of a variety exploiting graphics wherever possible.

[12]RAND has had good experience building such models in RAND-ABEL (Shapiro et al., 1985; Shapiro et al., 1988; Davis 1990), ROSS (Klahr and Waterman, 1986), DMOD (Narain, 1989), ROSIE (Kipps et al., 1987; Sowizral and Kipps, 1985), and RLISP (Marti, forthcoming), but all of those languages have significant shortcomings. RAND-ABEL, for example, does not have object orientation, graphical design aids, graphical descriptions, or artificial-intelligence-style "inference" features of the sort that allow backward-chaining.

"intelligent" tools that would help users test for subtle and domain-specific problems.[13]

- Many high-level languages and other languages with important specialized capabilities result in slowly running simulations that do not permit appropriate exploratory analysis or sensitivity analysis. Some of the features that can improve understandability or provide specialized capabilities, such as goal-directed search, reduce run-time performance. Thus, there are tradeoffs to be made.
- Many current programming and support tools do not permit graceful extension into new technologies that are becoming available and important, such as object-oriented modeling and programming and use of hypertext and hypermedia and visualization systems.[14]

What follows, then, is a research effort to address these problems, an effort initiated with internal funding but that is now being supported in part by the government. This effort will draw upon computer projects already under way, plus new ones to be undertaken. Before describing current efforts and the scope of the proposed effort, we mention in the following section some recent technological developments that create opportunities that should be taken into account in any "next-generation" modeling environment development effort.

[13]Some data problems are, in our view, inherently substantive and will continue to be difficult even as technology advances. The basic problem here is that evaluating data bases requires, in many cases, intimate knowledge of the model in which the data are to be used and the problem to which the model is to be applied. We do not discuss these matters in the current Note, but we see significant challenges here for organizations with respect to standardization, centralization, and training—of both technicians and professionals.

[14]The issues here are many and varied. They include the character of the underlying languages, closed architectures, and conflicts of "style." So it is, for example, that DOS systems are gaining important functionality only by sophisticated intermediate mechanisms such as the Windows program developed at great expense by Microsoft.

3. TECHNOLOGICAL OPPORTUNITIES

The previous section took a top-down *enlightened demand-pull* approach to the discussion of a next-generation modeling, simulation, and analysis environment for RAND (i.e., an approach that looks not only at current demand but also demand that can reasonably be anticipated from objective needs, whether or not widely recognized). It is also important, however, to be cognizant of *technology-push* considerations. There are a number of recent developments that provide new opportunities for modeling and analysis but at the same time increase the danger of obsolescence in doing business as usual. We briefly introduce a number of these technologies here. These topics were chosen because of their relevance to the activities composing the total modeling "environment" shown in Figures 2–3 above.

OBJECT-ORIENTED PROGRAMMING LANGUAGES

Object-oriented techniques have long been seen as desirable in simulation. Indeed, one of the very first object-oriented languages (SIMULA) was developed 25 years ago (Dahl and Nygaard, 1966). However, it has only been in the past five years that highly organized concepts of object-oriented design[15] and taxonomies of object-oriented features[16] have been developed, resulting in mature programming languages such as C++ (Stroustrup, 1986), CLOS (Bobrow et al., 1988), and Eiffel (Meyer, 1988).

We believe there are many advantages in the use of a design and programming methodology that is *object-oriented.* This technique basically describes the world being modeled as a set of software objects, each of which is described by a set of attributes, and a set of operations that cause attributes of the objects to change. There is a clear distinction between processes that the object "owns" and those it does not, and considerable effort is made to limit the degree to which one object's processes depend on information owned by other objects. The result is a high degree of modularity in the resulting design and the programming code. Object-oriented programs also tend to be more concise, because part of

[15]See, for example, Rumbaugh, Blaha, Premeriani, Eddy, and Lorensen (1991); Coad and Yourdon (1992); and Zeigler (1990).

[16]Perhaps the sentinel event was the Association for Computer Machinery's (ACM's) 1987 Object-Orient Programming Systems, Languages, and Applications (OOPSLA) conference where proponents of various inheritance/delegation mechanisms generated the "Treaty of Orlando," outlining the benefits of static vs. dynamic inheritance, implicit vs. explicit delegation, and so on. Resolving that "different programming situations call for different combinations of features," the various factions were freed to develop systems supporting features suited for particular goals without the "religious" infighting that had characterized the language developer's world.

the definition of the concept involves the capability of an individual object to inherit characteristics from a class of objects to which it belongs. In this way, duplication of coding is minimized. Object-oriented techniques have been used in simulation for nearly a decade at RAND, with early seminal work on ROSS (see, for example, Klahr and Waterman, 1986, Chapters 3 and 7). More recently, object orientation has been used for combat modeling in the operational-level combat-modeling TLC/NLC project sponsored by the Air Force and Army (Hillestad, Moore, and Larson, forthcoming), and in an Army-sponsored project using the RISE system (Marti, 1988, 1990; Marti and Catsimpoolas, 1992). Object-oriented methods are also being used by the Army, Los Alamos National Laboratory and MITRE in development of the EAGLE combat model for corps-level battles. We believe the *option* to use object-oriented methodologies must be available (and their use encouraged) in any next-generation modeling environment.

At the same time, by no means do we believe object-oriented programming is a panacea. It is not even appropriate for real-world systems in which the organizing principle of "objects" is unnatural. This is somewhat analogous to physics in which one uses particle representations for some problems and wave representations for others. The former work best when the particles are readily identified and largely independent. The latter work best in problems with continuous phenomena. Object orientation is also inappropriate for certain systems that are being viewed from a process perspective.

Extreme versions of object-oriented programming are also troublesome. We observe, for example, that many real-world models necessarily involve a great deal of interaction among objects and that some of the interactions are not naturally represented by "messages." Further, many models require numerous global variables, because there are real-world features that affect many objects at many times (e.g., aspects of the terrain and weather).

It follows, then, that we believe an appropriate environment would make it easy to use either object-oriented or other paradigms as appropriate, preferably without having to change languages. We also believe, based on project experience, that special effort must be made to improve the comprehensibility of object-oriented models with respect to data flow among objects. For real-world systems with many object interactions, comprehending those interactions can be quite difficult in current object-oriented programming formulations.

PERSONAL COMPUTERS AND WORKSTATIONS

Personal computers and workstations now provide a great deal of modeling power, and the wealth of mass-produced inexpensive software available for them creates major opportunities for plotting, calculating, visualizing results, and interacting with the user through displayed "windows" of information. Any future environment should take maximum advantage of these off-the-shelf hardware and software systems and allow personal computers to be used as front-ends (clients) to other computers (servers) on the same network that are executing simulations or storing major data bases to be accessed. By using IBM-compatible PCs, Macintoshes, and workstations as the main user interface to models and simulations, the user is presented with a comfortable, familiar interface in which various interactions and actions have standard meanings and results. Within RAND, as the result of a strategic decision in 1990, almost every researcher has such a Macintosh, PC, or UNIX workstation on his or her desk, and that same equipment might be used to access and manipulate models, simulations, or analysis programs existing at various places on the network that connects the computers at RAND.[17] Some of this has been happening for the last year or so, but the potential has not yet been approached.

HIGHLY INTERACTIVE MODELING AND ANALYSIS ENVIRONMENTS

By contrast with many workers and most trends in nonspreadsheet programming languages, we also stress the need for a *highly interactive modeling environment*, in which a user can move routinely and efficiently among such functions as designing, programming, running, postprocessing, and viewing results of a model.[18] Anyone who has used a modern spreadsheet such as Excel should appreciate that high interactivity greatly enhances one's ability to build and iterate good models. Unfortunately, standard spreadsheet programs are inappropriate except when the level of complexity is modest (there are issues of both efficiency[19] and organization, limitations that can also encourage poor design practices). A next-generation workstation environment can support this highly interactive mode of operation with a general-purpose language (Anabel) by allowing portions of the model being run to be interpreted directly from the source code while other portions have been compiled into a more efficient (but less modifiable) code. Such flexibility has been exhibited for about seven years at RAND by the RAND-ABEL language used in the RAND Strategy Assessment

[17]Establishing the requirement to exploit PCs and Macs, and to move away from sole reliance on Sun workstations and the UNIX environment, was an early decision in our exploratory work on environments. The Mac and PC tools are becoming a standard. Further, many analysts clearly want to work at their desks with commercial tools such as those on the Macintosh and not in a workstation laboratory down the hall (although there are tradeoffs we will mention later).

[18]For the sake of run-time performance, interactivity should preferably be optional, as with languages that can be run in a compiled mode, an interpreted mode, or a hybrid mode.

[19]There are measures that can be taken to improve efficiency. These include compiling the code, adjusting memory parameters, and so on. In our experience, however, these measures are not yet practical for most modelers and analysts without help from technicians and programmers. Also, spreadsheet languages are still quite restrictive in many respects.

System (RSAS) and other RAND projects.[20] It is also common in systems written in the Lisp list-processing language, such as the RISE system (Marti, 1988; Marti, forthcoming). It is much less common in the general community, however, and we believe this is an important subject for emphasis, especially when combining highly interactive environment features with a *comprehensible* programming language.[21]

VISUALIZATION

Any next-generation modeling and analysis system should contain tools and capabilities for *visualization* of model structure and behavior of the data representing model results and of the analysis results themselves. Major new capabilities are now available for visually "flying over" battlefields to view them from various perspectives (notably the "Magic Carpet" of the SIMNET system developed by the Defense Advanced Research Projects Agency [DARPA]) or even to create a "virtual reality" in which one can be immersed in a three-dimensional artificial world created by the computer to represent a model or configuration of data. Any next-generation environment should be compatible with providing these capabilities to the user for visualizing model behavior or data configurations.[22]

There is also a growing body of literature and sets of commercially available software tools that support "*computer-assisted software engineering*." These tools, including graphical tools, assist in the design, programming, execution, and documentation of software systems, and provide a "data repository" in which definitions, software objects, versions of programs, and the other impedimenta and effluvia of model construction and operation can be stored in a consistent manner. Such tools and techniques are needed by the next-generation modeler

[20]Many hundreds of thousands of lines of code have been written in RAND-ABEL, which is used in a number of government agencies employing the RSAS. For discussion of RAND-ABEL, see Shapiro et al. (1988); Shapiro et al. (1985); and Davis (1990).

[21]Many skeptics exist on this matter because, in the past, promises of transparency and comprehensibility have been exaggerated. Ultimately, one cannot avoid the facts that many computer programs are complex and that understanding complexity is not easy. Further, full computer programs contain not only the segments that can be made relatively readable and friendly, but a great deal of material related to input and output, control flow, declarations, and so on, all of which tend to severely reduce overall comprehensibility to nonprogrammers. And, finally, programmers can persist in writing unintelligible programs even if they have a user-friendly language. Nonetheless, we are encouraged by the substantial progress made with RAND-ABEL in the 1980s and believe much more is now achievable.

[22]A dramatic example of visualization has been developed by DARPA in the *73 Easting* experiment, which consisted of using the SIMNET system to reconstruct and simulate an armored battle in the 1991 war with Iraq. A video tape providing some highlights is available from DARPA. For a cogent introduction to SIMNET, see Miller (1992).

(see, for example, Davis, 1992a, Appendix A), but they seem not yet to be available at reasonable costs.[23]

EXCHANGE AND REUSE OF MODELS AND TOOLS, INCLUDING DISTRIBUTED SIMULATION

It is now possible to imagine truly *distributed simulations*, in which portions of a simulation migrate to different computers, even to hundreds of computers linked by networks—not just locally but internationally. In this way, not only might simulations be run in hours that might take days or weeks on a single processor but analysts and other participants can learn and share insights and knowledge via interaction with the simulation. This is now technically possible, although there are difficulties to be overcome (Bankes, 1992a). The ability to create distributed simulations is also closely related to the need for object-oriented data-base management systems that are capable of storing "persistent objects" (i.e., whose existence and attributes persist in a data base beyond any particular simulation run). Distributed simulation will become commonplace in activities of the military services and commands (e.g., in distributed war games and studies using the revolutionary SIMNET system).[24]

A related topic involves developing *models that can be reused and even exchanged*, not only within an organization but with outside users, perhaps involving only an electronic transfer of the code representing the model. The DoD is attempting to move strongly in the direction of developing standards to permit model interoperability, reusability, etc. (DoD, 1992). There has been little done, however, to provide a base of textbooks and advanced documentation tools to make such transfers easier.[25]

There is now emerging a much more sophisticated understanding of data-base and data-manipulation problems. Some of these involve constructs such as *data dictionaries, data encyclopedias, and data-access facilities,* all of which are crucial to interoperability and

[23]One notable example of graphical assistance is that provided in the Stella and iThink programs, which implement System Dynamics. To a significant degree, at least, one builds models with the graphics rather than the graphics being an afterthought. Further, there are many controls built in to assure complete specification. Even these programs, however, go only part of the way toward what we have in mind here.

[24]In our judgment, however, current enthusiasms for distributed interactive simulation often obscure the substantive difficulties that arise when one connects dissimilar models built for different purposes by different organizations. We believe that careful *analysis* will continue to require much greater control and intimate knowledge of the relevant models than will soon (or perhaps ever) be possible in SIMNET and other DIS experiments. See Bankes (1992a).

[25]An important recent DoD program to enhance reusability is the Joint Modeling and Simulation System (J-MASS) effort sponsored by Wright Laboratory. The effort includes software development standards and tools (see, for example, SofTech, 1991). It is, however, strongly oriented toward Ada.

reusability.[26] Another class of activity involves "intelligent data-base systems," which include knowledge-based rules and algorithms to help review, format, adjust, or fill in data—often in an interactive setting. Such technologies will be very important in the future, because large and complex data bases will be network shared, but individual users such as RAND will need to do a great deal of review and adjustment. Today, that process is painful and unsatisfactory. As noted earlier, much of the pain appears to be unavoidable because reviewing and correcting data require intimate knowledge of the model and application.[27]

TECHNIQUES AND METHODS

Beyond "What If?"

Most complex military simulations are what RAND colleague Jeff Rothenberg has described as "toy duck" simulations: "you wind them up and see where they go." (Exceptions include games and other interactive simulations.) For most analytical simulations, given a set of initial conditions, one executes the programs and observes final results after n seconds or minutes or days of simulated time. They are answering the question: "What if I start with these conditions and the model adequately represents a pertinent slice of reality and the initial conditions are a sensible set in the context of the model?" A next generation of modeling technology should explicitly attempt to design models in which additional questions—often ones of more fundamental and direct interest to the end-user of a model—can be answered, questions such as: "Why did a particular event occur?" "Why did an event *not* occur?" "What was object x doing when object y did *some event*?" "What factors determine whether *event x* occurs?" These questions, and other similar ones, are referred to as "beyond what if" in a recent RAND research report[28] describing work toward the goal of addressing such questions. This work concentrates on building a notion of causality into the definition of a model, so that chains of causality can be traced. We believe next-generation modeling environments should broaden the scope of modeling to include facilities to allow development of models in which *"beyond what if"* questions regarding a model can be answered.[29] One difficult challenge here is thinking through how one can combine the beyond-what-if techniques, which currently rely on Prolog-style backward-chaining inference

[26]See, for example, Cammarata, Shane, and Ram (1991).

[27]See also discussion of verification, validation, and accreditation of data bases in Davis (1992a).

[28] See Rothenberg, Narain, Steeb, Hefley, and Shapiro (1989); and Rothenberg (1992a-b). See also Round (1989), which gives a good overview of knowledge-based simulation and applications.

[29]For a relevant paper on modeling and simulation written a decade ago by one of the authors (Shapiro) and colleagues, see Davis, Shapiro, and Rosenschein (1982).

and languages that are not analyst friendly, with techniques employing Anabel, MODSIM, or other relatively high-level procedural languages.

Explanation Capabilities

One aspect of the beyond-what-if issue is demanding of models a much higher degree of *explanation capability* than has been customary. Typical simulation models have very poor explanation capabilities, which greatly increases the time required for verification, validation, and analysis. We are fortunate that it is *relatively* easy to build substantial explanation capabilities into models developed in RAND-ABEL.[30] Further, extensions of the language should make it possible to include more self-testing features, which would greatly aid verification and some aspects of validation (e.g., testing for logical completeness).

Optimization and Other Adaptive Planning Methods

Optimization techniques remain important in simulation. RAND has made extensive use of one such method developed by Richard Hillestad, originally as part of the theater-level TACSAGE combat simulation. More recently, the SAGE algorithm has been made "generic" so that it can be employed in diverse types of simulation. We believe there should be a systematic effort to explore additional applications of SAGE and SAGE-like methods in policy analysis and that the proposed modeling environment should allow graceful application of this technology as appropriate. Another subset of planning methods involves knowledge-based decision models using highly structured (or simplistic) heuristic rules. These may be used in lieu of human participation in war games or as decision aids for humans. The strongly hierarchical *knowledge-based methods* used in the RSAS to describe political- and military-level decisionmaking should have broad applicability.[31]

Exploratory Modeling and Analysis

One of the key features of policy analysis is the existence of uncertainty in many dimensions. Analysts need to shift from using models as answer machines toward using them as devices for *exploratory modeling and analysis*—"exploring" the problem space and posing tradeoffs in a way that permits the application of judgment, intuition, and arbitrary

[30]"Explanation" is, of course, a matter of degree and we are not discussing "deep explanation," which is a frontier topic in artificial intelligence. Rather, we have in mind what amounts to hierarchically structured log statements providing reviewers of a model run the ability to step readily through model calculations and logic, much as one might do manually for simpler computer problems. We have found such explanation capabilities to be exceptionally useful in both combat models and political models developed in RAND-ABEL. See, for example, Davis (1988).

[31]For discussion of how decision models (or "planning models") might be used more effectively than in the past, see Davis and Hillestad (1992).

tie-breaking decisions amidst uncertainty.[32] Researchers developing tools and techniques to aid in this exploratory process (e.g., Bankes, 1992b) should work closely with persons developing the proposed modeling, simulation, and analysis environment, so that synergies are exploited in both directions as these efforts unfold.

Analytic Gaming

Another discovery of the 1980s is the feasibility and value of "analytic gaming," in which one combines the best aspects of human gaming with those of closed simulation. In much of the work conducted with the RSAS, for example, there is a crucial period of gaming and exploration, which is then followed by well-controlled simulations for the purposes of deductive analysis. The interplay between these phases of work is much stronger and more highly interactive than in the past. We believe that such an approach to analysis is general and should not be limited to defense analyses. Indeed, it is a generic problem of systems analysis that practitioners too often leave out crucial features of the real-world problem because they involve squishy topics such as perceptions, attitudes, organizational behavior, biases, and distinctly nonoptimal behavior—topics well addressed in human gaming. With highly interactive modeling and analysis systems, one can hope to do gaming, exploratory analysis, and deductive analysis in the same framework.

OTHER TECHNOLOGICAL DEVELOPMENTS RELEVANT TO A COMPREHENSIVE ENVIRONMENT

With the continuing commercialization of products providing *hypertext and hypermedia* capabilities,[33] fundamentally new opportunities are provided for documentation of models. They also permit model output and even code to be structured in flexible and

[32]There are many strands of this approach through RAND's history. The late 1970s Policy Analysis of Water Management (PAWN) study, for example, highlighted the use of subjective scorecards (Goeller et al., 1983). RSAS studies (e.g., Davis, 1988) emphasized multiscenario analysis and a shift from dubious cost-effectiveness tradeoff calculations to identification of Achilles' heels and high-leverage potentials. Recently one of us proposed a new set of tools for exploratory modeling and analysis (Bankes, 1992b), which he is pursuing in an ongoing project. Other relevant work includes a proposal by one of us (Shapiro) to build in the capability for routine sensitivity analysis (Rothenberg, Shapiro, and Hefley, 1990) and the extensive experience of colleague James Bigelow with Army-sponsored "repro models" that can be used for sensitivity testing and the development of approximate so-called response surfaces.

[33]Hypertext capabilities allow one to create networks of interconnected text items. One can move from one point to another by pushing a "button" on the screen, which brings up, for example, relevant documentation or examples. Macintosh computers come with a system called HyperCard® that is being used more and more extensively. Hypermedia refers to the ability to embed information represented in different media (e.g., video sequences, voice record and playback, graphic animated sequences, charts, and diagrams) within one document. See Nielsen (1990) for an introduction to these techniques and a brief overview of one of the systems, HyperCard, we will be using on this project.

nonlinear ways. For example, a user might insert voice annotations within his model code or data during an active session as reminders of the reasoning behind changes being made. Or a user might insert "movies" of sample model runs or even conditions for a "live" model run that the reader can execute as an example of a particular behavior simply by "clicking" a button.[34]

The environment for modeling, simulation, and analysis we propose involves much more than software. It also involves the ability of researchers to collaborate and coordinate their work, across the boundaries of time zones and distance. It is now becoming practical to link two or more remote sites with *video teleconferencing*, so that work groups—for example, in RAND's case, in Santa Monica and Washington, D.C.—can communicate much more effectively, and have tightly linked computer models and graphics in the process. Videoconferencing is proving to be a major success in portions of the U.S. government and in a number of large and small companies worldwide. Combined with the possibilities for distributed simulations (mentioned above), it becomes possible to conceive of truly distributed modeling, simulation, and analysis activities tying together all relevant persons and systems within an organization and to offices and individuals elsewhere in the world. In time, worldwide capabilities may become fairly commonplace. The proposed new modeling environment should be designed with such distributed activities in mind.

In a world in which developments in all of the above areas are creating new synergies and standards, it is time to set out clear goals and plans for modeling, simulation, and analysis activities that will focus scarce resources on those activities that will benefit the entire modeling community. The rest of this Note is essentially an elaboration of how we could achieve this potential and what our priorities should be. It reflects a combination of demand-pull (i.e., making Figure 3 real) and a significant component of technology-push thinking, to take advantage of the opportunities sketched above.

---

[34]We and other RAND colleagues have already conducted experiments with voice annotations and Quicktime® "movies," which are actually computer-stored and -generated displays of model graphics in a movie-like format.

4. PULLING THINGS TOGETHER INTO A STRATEGY

Given the background of demand-pull challenges and technological opportunities described in Sections 2 and 3, and given the particular organizational context that exists at RAND, what strategy for moving toward a more comprehensive and ambitious environment makes sense? This was a key issue for us in mid- to late 1991. The conclusions we reached and the strategy that resulted reflected many personal judgments and hypotheses.

OBJECTIVES

First, we reviewed objectives. Our overall objective was to improve computer support for all aspects of the study process, as suggested in Figures 1–3. In addition, we had the rather more specific objective of pursuing research on two topics: (a) friendly, comprehensible, and highly interactive programming languages and (b) variable-resolution models and hierarchically integrated families of models with varied resolution. RAND in general and we in particular have long-standing research interest in these topics. Further, they are important topics that appear to us understudied elsewhere. Both of these topics are highly relevant to establishing an overall modeling and analysis environment, but others interested in environments might be relatively less interested in the programming language or the challenge of linking models together across levels of resolution.

COMPONENTS OF STRATEGY

Given the overall objective and the more specific and consistent subobjective, we next sought to identify the major components of strategy. They appeared to us to be as follows:

1. *Central issues of software*: language(s), tools, architecture, and, in particular, use of commercial off-the-shelf (COTS) tools.
2. *Nonsoftware aspects of the environment* (e.g., network links and model-supported videoconferencing).
3. *Model and data management.*
4. *Organizational issues* (e.g., how to stimulate interest and "demand" for the emerging capabilities, how best to have current and prospective project needs influence development, and how best to bring into the overall environment-generating activity the fruits of many separate research efforts within RAND).
5. *Long-term support and maintenance.*
6. *Development schedule and associated funding requirements.*

SOFTWARE STRATEGY

We were determined from the outset to exploit COTS, but there were differences of opinion about what that meant. Several of us (Hall, Shapiro, and Bankes) had been working primarily with Sun workstations in a C/UNIX environment. The others (Davis and Anderson) had been working more extensively with microcomputers and associated tools such as desktop publishing and spreadsheets. Other RAND colleagues were working with various dialects of LISP, MODSIM, FORTRAN IV, and a number of other languages. Some colleagues were working with Hewlett Packard workstations. There was general concern about adopting an open architecture approach to the maximum extent feasible.

The Language Issue

On the one hand, we were interested in pursuing high-level languages as noted above. On the other hand, developing our own language was not a foregone conclusion because of the advances in commercial off-the-shelf systems, including object-oriented languages such as MODSIM, with which RAND has had good experience, and the spreadsheet language Excel, which is now far more powerful than spreadsheets of a few years ago. We also had to consider seriously the Ada option, since the DoD has been strongly encouraging use of Ada by its contractors. Nonetheless, after considerable discussion we concluded that developing our own central high-level language was desirable for a variety of reasons (see also Appendix A). We decided to do so following a concept for a language, Anabel, that had been conceptually defined by one of us (Hall) in an earlier effort. A very attractive feature of this approach was that it would allow us to start with a very strong base, the RAND-ABEL language mentioned earlier.[35] Anabel was to be an extension of RAND-ABEL with object-oriented features and a great many other features coupling the language into a mini-environment of graphical and other tools.

Significantly, Anabel development was to be only part of the overall effort and the "Anabel environment" only part of the overall environment (Figure 4). This was especially important because we were determined not to allow the development effort to turn into a

[35]By a strong language base we mean that RAND-ABEL has been well tested and used for a half-dozen years, both within RAND and, at least to some extent, in about a dozen DoD organizations. Further, RAND-ABEL incorporates many of the features most needed in a language that is to be both comprehensible and appropriate for complex programming. These include an efficient interpreter permitting highly interactive operations, an "active" data dictionary (i.e., one used by the compiler to detect a wide variety of errors such as mistyped names or equations relating data of different types), straightforward mechanisms for "dropping into C/UNIX" to exploit library functions and specialized capabilities, and explanation capabilities. Our experience is that a good programmer trained in, for example, Pascal, can learn RAND-ABEL in a matter of a week or less.

"hobby shop" effort to explore interesting language features. In particular, a firm element of strategy was the requirement that

- The environment should allow use of multiple languages (e.g., C, MODSIM, Ada, FORTRAN IV).
- It should be possible to have Anabel programs be called by programs written in other languages and vice versa.

The techniques for accomplishing this were to be based on "wrapper" methods, which are becoming widely used by those concerned with open architectures, distributed simulation, and interoperability (see, for example, Weatherly et al., 1991, and Landauer and Bellman, 1992).

To further hedge in this important realm of language, we agreed on a strategy that would preserve a great deal of flexibility. In particular: (a) Anabel will translate automatically into C and C++, thereby assuring portability of this immediate code to organizations that do not choose to have an Anabel compiler; (b) Anabel will maintain options for "falling into C/UNIX" as necessary to provide specialized capabilities; (c) Anabel development will be phased, with intermediate deliverables and milestones, at which points the premises of Anabel should be reviewed.

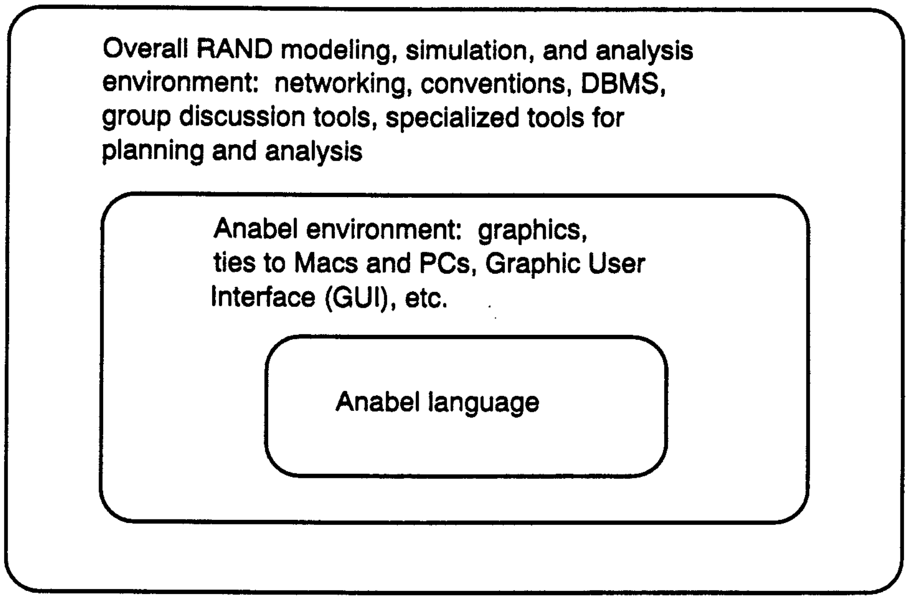
Overall RAND modeling, simulation, and analysis environment: networking, conventions, DBMS, group discussion tools, specialized tools for planning and analysis
Anabel environment: graphics, ties to Macs and PCs, Graphic User Interface (GUI), etc.
Anabel language

Figure 4—Relationship of Anabel to the Overall RANDSIM Environment

Software Associated with Microcomputers and Workstations

A second major decision was that we should tilt *early* toward microcomputers (particularly Macintoshes) to provide friendly and powerful human interfaces. There now exists a wealth of tools that it would be utter folly to reproduce from scratch on workstations.[36] These include Word, PowerPoint, MacDraw, Excel, MacFlow, HyperCard, iThink, and 4th Dimension. Although we were not yet satisfied with any of the commercially available data-base management systems, we decided that RAND should not develop such a system. Nor should it develop network facilities or systems for communicating with the outside world. Instead, we would await commercial developments.

This approach emphasizing microcomputers was not inevitable. In particular, we could have chosen to rely entirely on workstations such as the Sun Sparc stations. However, our judgment was that the tools on microcomputers were in many important cases better and that many of the analysts we intended to serve would want to work exclusively on their "personal" computers (Macs or PCs) and would not even want to wander down the hall to a laboratory with Sparc stations or to have two computers in their offices. Others, of course, would be quite willing to do so to gain full benefit of workstation power and the synergisms of rubbing elbows with other workers, but we decided, on balance, to tilt toward exploited microcomputer interfaces. This implied a major investment in learning the intricacies of Macintosh programming and the mechanisms by which such programs can be connected to C/UNIX and DOS systems.

One consequence of this approach is that we were unable to completely avoid machine-specific systems. For example, we decided to do prototyping work on hypermedia documentation by using the HyperCard system that comes with Macintosh systems. On the other hand, to the maximum extent possible, development of interfaces will follow open-architecture procedures and standards.

NONSOFTWARE ASPECTS OF THE ENVIRONMENT

Less detailed thinking went into this part of the strategy, primarily because we knew that we would be funding limited, with software development having an early priority. However, part of our strategy was to lobby, within RAND, for videoconferencing capability (which became operational in September 1992). For a wide variety of reasons RAND was pursuing other important capabilities as well (e.g., proliferation of microcomputers and

[36]This meant abandoning some excellent tools developed earlier in RAND for use on Sun workstations and other C/UNIX machines. However, we judged that those tools could not long remain competitive without continuing investments and support expenditures that we could not justify.

workstations with a comprehensive netting capability, both within our Santa Monica offices and to our Washington office). RAND is also reviewing its publication policies to recognize the growing significance of CD ROM technology. When all is said and done, then, we did not go much beyond "visions" in this aspect of the overall problem. Instead, we concluded that nonsoftware aspects of the environment would be a major theme of work in the second and third years if the project went ahead.

MODEL AND DATA MANAGEMENT

An extremely important part of providing a good modeling and analysis environment, especially for analysts using complex models and data bases, is providing the facilities and tools for model and data management. These affect the ease with which modelers and analysts can, for example, cope with models that are constantly changing in response to study-specific demands, review and enhance data bases, draw on repositories of standard "scenarios" (i.e., sets of initial conditions for model runs, each of which corresponds to an important "case"), and design and manage experiments that may involve dozens, hundreds, or even millions of model runs.

There has been considerable thought given to these problems in recent years (see Bennett, 1989; Nance, 1987; and Appendix C). Our strategy here is to build on other efforts that are ongoing in RAND and elsewhere and to try to integrate them into the overall environment after we have made substantial progress on the "software strategy" described earlier.

ORGANIZATIONAL ISSUES

Participation, Support, and Organizational Learning

From the outset we recognized that introducing a new modeling and analysis environment would require paying major attention to organizational issues. We needed to understand potential users and how their needs would be or should be changing in the years ahead. We needed to enlist their support and enthusiasm. We would need early experimenters. Further, gaining and maintaining organizational support would require providing at least some significant and useful services "early" (perhaps in the second year). There were issues of scope (which users did we want to support?), early priorities (which ongoing projects might be users?), early opportunities for integration of currently separate models, and similar considerations.

Although, once again, we recognized that a year or so of software development would largely have to precede progress on other matters, we agreed on a strategy that included the following elements:

- Designing to support research on social-policy, international political, and economic research, not merely military problems. That is, RANDSIM was to serve *all* of RAND.
- Extensive consultation with researchers using a wide variety of techniques and addressing a wide variety of problems, in part to establish requirements and challenges and in part to interest them in the endeavor. The range of model types should embrace, for example: discrete-event and time-stepped simulation, deterministic and stochastic modeling, cost-effectiveness modeling, optimization, decision support, and data analysis.
- Creating one or more RAND-wide organizations for interdisciplinary technical-level exchange of ideas, friendly peer review of alternative approaches to modeling, group discussion of state-of-the-art developments, and so on. This should be much more than a mere seminar series (of which we already had several).
- Creating, at the appropriate time, advisory committees with participants from both within RAND and outside universities and other research organizations.
- Further encouraging open publication of research findings related to principles, generic methods, generalized tools and methods, and other scientific results.
- At the appropriate time, seeking funds for experiments using emerging capabilities on actual projects that would, if successful, generate widespread interest and subsequent research support.
- Exploiting the opportunity of ongoing model-using or model-developing projects to identify and address specific problems. Within this activity, giving priority to efforts that could produce integrated hierarchical families of models with varied levels of detail, since having such families is extremely valuable to policy analysis and current families of models are neither well integrated nor easy to use. This, of course, relates back to the second of the subobjectives we mentioned above, advancing the state of knowledge in the area of variable- and cross-resolution modeling.

Standards

The subject of standards is often considered a foul and unpleasant topic in research organizations filled with independent, hard-charging, and busy individuals. However, we are now in an era in which software and modeling standards are increasingly desirable and inevitable. One sign of this is that the researchers involved are willing to identify many specific examples of where they themselves value standards. Another sign is that with the tightening of budgets as defense expenditures fall, it is no longer acceptable to have *inappropriate* redundancy of effort. A third sign is the increasing importance of distributed simulation and distributed computer operations of all kinds. This *requires* a great deal of standardization (e.g., the SIMNET protocols in military work or the protocols allowing researchers to exchange heavily formatted manuscripts with graphics over the telephone lines). Yet another sign is the increased emphasis being given to VV&A (verification, validation, and accreditation) by the DoD. The Army, for example, has recently issued firm regulations requiring VV&A on models used to support Army decisions. Standardization, VV&A, and efficient documentation are all going to be critical in distributed operations.

We concluded, therefore, that we would encourage a significant shift within RAND toward standards, VV&A, documentation, and centralized registering of models, data bases, VV&A records, and so on. This would not, of course, be an across-the-board affair. Nonetheless, we believed that a considerable shift in that direction was desirable. Our recommended strategy included the following:

- Encouraging all project leaders to emphasize modularity, open architecture, early documentation, and VV&A—even for spreadsheet modeling.
- Developing guidelines for VV&A, which are embedded in a larger study completed for OSD's Defense Modeling and Simulation Office (DMSO) (Davis, 1992a).
- Participating heavily in DoD activities to develop standards and guidelines.
- Recommending use of infrastructure funds to support researchers willing to put together internal standards (e.g., an internal standard for geographical information used in a number of RAND studies).
- Arranging serious debates and technically informed management decisions on issues involving alternative standards.[37]

[37]Examples here include decisions about (a) desktop publishing software (RAND has tilted toward use of the Microsoft family of products that can be used on Macintoshes and on IBM PCs and clones using Microsoft Windows); (b) the choice of graphics systems to use on workstations; and (c) a review of whether some emerging software would be readily portable to a different workstation.

Work began on all aspects of this strategy in late 1991 and 1992. The outcome will not be clear for years.

LONG-TERM SUPPORT AND MAINTENANCE

A major concern in thinking about development of a new and far-reaching environment had to be long-term support and maintenance. It is one thing to develop advanced languages and tools; it is quite another to make sure they can be readily used and that they will continue to work over time, even though computers and technical standards change, new features are introduced, the original developers move on to other projects or other organizations, and so on. Similar problems sometimes exist when one counts on a commercial developer, but if the product is widely used and profitable there is some reason to expect continuing support. An additional concern here is that if our environment tools were successful, and were exported to government agencies and even into the public domain, we would have an *obligation* to provide continuing support.

Our strategy for addressing these issues had two components:

- Enlisting the assistance of internal RAND support organizations when the time came (notably the Military Operations Simulation Facility, the Information Sciences Laboratory, and the more general computer-support division).
- Planning, at some stage of the development process, to investigate collaboration with a commercial vendor.

The second merits the most discussion. The question here was "Should RAND plan on full support and maintenance of the next-generation modeling and analysis environment or plan early collaboration with a commercial vendor who could assume professional documentation, packaging, maintenance, and support of the resulting product?" We concluded that RAND has neither the facilities, interest, nor charter to support indefinitely products of its software research activities that come to have extensive external use.[38] Further, except for some relatively small but important efforts (of which there are numerous examples over the years), it would not be appropriate for RAND to do so, because that is more in the province of commercial companies than of federally funded research and

[38]It is worth mentioning here that one of us (Shapiro) was a principal developer of the electronic-mail system mh, which has subsequently been further developed and implemented in a wide variety of organizations using UNIX systems. RAND, then, created the system, but others have maintained and extended it. This same pattern occurred in RAND's development of the Simscript simulation language, which became a CACI commercial product.

development centers. After some stage of the development process, then, collaboration with a commercial vendor should be initiated. However, we judged that the prospects for *early* success in seeking commercial collaboration are much fewer than some enthusiasts would suggest. We concluded that it would be a strategic blunder to delay pursuing our own initiatives until some such vendor relationship could be developed. It followed, we concluded, that we must plan explicitly to provide a high degree of in-house quality control and maintenance capability until some external vendor relationship is established.

DEVELOPMENT SCHEDULE AND FUNDING ISSUES

We describe here the original notional three-phase technical development for the prototype modeling and analysis environment. As of July 1992, we were almost one year into the effort, but the effort is underfunded, so progress has been slower than desirable despite major progress.

Appendix B contains a brief discussion and diagram of the system architecture proposed for the computer programs supporting the next-generation modeling and analysis environment. It focuses on the Anabel aspects of the overall environment.

Our plan shows results in deliverable systems, each with growing capability, at the end of each project year (the first year started in approximately June 1991). As mentioned above, these intermediate deliverables can be evaluated to assess the progress and utility of the modeling and analysis language and environment at key checkpoints in its progress.[39] Key features of the three "checkpoint" systems are expected to be as follows:

Year-One Environment Features:

- Initial RAND-ABEL language enhancements to include first-order object features
  —Extensible executable table types
  —Data structures and object-oriented "methods"
- Links to PCs and Macs, including tools such as Excel
- Table-oriented user interface providing table format access to all model variables
- Initial simulation modules for event- and time-management routines
- Initial data-base support, including initialization and storage of model state
- Hypertext documentation and browsing facility

[39]Note added, October 1992. The first year's features have been successfully developed. They are now being reviewed and evaluated.

Year-Two Environment Features:

- Full Anabel language
  —Highly flexible objects and tables
  —Ability to integrate nontextual elements
- Integrated tabular and graphical interface
  —Automatic display and manipulation of objects
  —Graphical display of object attributes under user control using Map View system
- Complete simulation library
- Extended data-base support
  —"Persistent objects" (i.e., objects transferrable from one model to another)
  —Tight coupling to external data bases
- Hypertext source code and configuration control that integrates model components, data, documentation, and cross-references
- "Wrapper" experiments involving combined use of models written in Anabel and other languages such as MODSIM or C
- Significant progress on nonlanguage aspects of environment, leveraging heavily on other RAND activities (e.g., installation of videoconferencing, extensive networking, and tools for data-base searches via telephone links)
- Significant progress in linking up to other projects developing methods and tools for model and data management

Year-Three Environment Features:

- Highly interactive interface
  —Graphical model-construction aids (probably not comprehensive)
  —Spreadsheet-style table interaction
- Tight coupling to external software
- Well-developed "wrapper" techniques allowing effective use of models written in diverse languages
- Integrated hypermedia (visual/auditory)
  —Documentation (e.g., audio or video annotation)
  —Presentation
  —Interface

- Extensive experiments with nonlanguage aspects of full RANDSIM environment, exploiting inexpensive opportunities likely to arise by the third year of the project

Funding requirements were uncertain, of course, but our initial estimate was that *at least* 10 man-years of effort would be required to develop a substantial prototype and that this figure depended on very effective leveraging of other projects and activities.[40] We should also emphasize that our estimate assumed top-quality talent.

[40]This figure is for development of the Anabel environment and low-cost integration of other ongoing activities into an overall environment. It does not include the manpower costs of the other activities. If all were included, the effort level would probably triple.

5. WHERE WE STAND NOW AND NEXT STEPS

We completed this Note in November 1992. Our recommended strategy remains largely unchanged. Further, we have completed the "first year" of development (in somewhat more than a calendar year) with the use of RAND's own funds and research support funds from RAND's federally funded research and development centers and a U.S. government sponsor.

Although we have been underfunded and future funding remains problematic, we are currently quite optimistic because of the substantial progress made in the last year, which will be reported separately in more detail. The remainder of this section summarizes the results of work to date and our planned next steps.

LANGUAGE DESIGN

We have completed a first pass through a detailed design of the extensions to the RAND-ABEL language, resulting in specifications for the language we call Anabel. That design has been described in an internal document that has been circulated for comments within RAND and to sponsors. Among the facilities and features included within this language design are (1) a generalization of the RAND-ABEL "Table" statement permitting a user to define a new table type and to provide its semantics in the form of Anabel code to be evaluated upon its execution; (2) the ability to define hierarchies of classes and create objects having both attributes and methods that are instances of these class structures; (3) lexical (static) and dynamic inheritance and both single and multiple inheritance among classes; (4) dynamic object creation and destruction; (5) overloading of operators and methods; and (6) a generalized iteration statement permitting, among other things, iteration over the objects within a class. In addition to these features, essentially all existing features of the RAND-ABEL language are retained.

PARSER AND COMPILER

We have completed both lexical and syntactic analyzers for the Anabel language and a generator of C language code resulting from this analysis. The semantic analysis is perhaps three-fourths complete. We are currently working on run-time facilities for model execution and completion of the syntactic analysis, including class libraries to support the run-time environment. Implementation of full dynamic inheritance within our class inheritance hierarchy is being deferred for now while these other facilities are being completed.

The lexical analysis program emits a stream of "tokenized" text in the form of SGML[41] annotations used by the hypertext authoring and browsing facilities discussed in the following subsection.

A colleague, Barry Wilson, has written a small sample Anabel program exercising many of the object-oriented features of the language for use as a test case in debugging and checking out the language-processing software being developed.

TOOLS FOR HYPERTEXT AUTHORING, BROWSING, AND DOCUMENTING

We have developed a specialized, yet general-purpose, editor for the Macintosh that works in cooperation with a special HyperCard stack to provide hypertext facilities for authoring and browsing Anabel code and related documentation. Among the facilities of this editor are the following.

Hypertext "Buttons"

The user may insert hypertext "buttons" that are associated with any other document or process on that Macintosh workstation or on other workstations accessible via a local area network. When double-clicked, a button brings the associated document or process window to the "front" of the desktop. If that document is itself an editor document, it may also contain hypertext buttons, including ones pointing back to the originating document. A special feature of these "buttons" is that they act as ordinary text and therefore may be part of text selections that are copied, cut, or pasted, or that partake in other normal text editing operations. The user may define the text string representing a button. Specialized textual icons representing standard processes (e.g., Microsoft Word, Excel, PowerPoint, HyperCard, MacFlow) are available as default characters that are used to represent buttons associated with documents created by these processes.

Voice Annotation

One specific type of hypertext annotation for which specialized features are provided is the creation of voice annotations that may be linked to a hypertext button. Voice playback occurs upon double-clicking of its associated button. The voice playback may be paused, then continued, or stopped, and the user retains full control over the editor (for example, to scroll or edit text within a document's window) while the voice playback is occurring.

[41]Standard Generalized Markup Language (see Goldfarb, 1990).

"Tokenizing" Anabel Source Code

Once Anabel source code has been created or revised, an editor menu command (or its associated shortcut key sequence) "tokenizes" the code by sending it over a network TCP link to a UNIX-based workstation containing the lexical analyzer. (The target UNIX workstation may be set at will as a preference by the user.) A text stream returned by the lexical analyzer, as mentioned above, is annotated in an SGML markup format to indicate—at present—text groups representing Anabel keywords or phrases, identifiers, constants, and comments. The user may select the mapping of these lexical categories into an arbitrary combination of text font, style, size, and color. The text groups, after tokenizing, are displayed using these text characteristics to distinguish them. For example, current default settings show keywords in boldface, identifiers in normal text, constants underlined, and comments in colored italics. Anabel code that has been tokenized may be retokenized. Hypertext "buttons" that have been inserted within the text file survive the round-trip through the lexical analyzer intact, retaining their meaning and operational behavior. Text containing these buttons may be copied, cut, and pasted within the same file or different editor files on a Macintosh while retaining their operational meaning.

When an Anabel keyword or phrase is double-clicked within tokenized text, a HyperCard card of reference information is displayed describing the syntax and semantics of Anabel statements in which that keyword may occur. This HyperCard stack also contains a scrollable index of Anabel terms from which any term or phrase may be selected to view its corresponding reference card. At present, this stack contains about 230 cards' worth of reference information.

File Suites

A set of files may be marked as a "suite" of files, and that suite given a special filename. When that suite filename is opened, all files within the suite are brought to the desktop together. They retain their ordering, front to back, that they had when last saved as a suite. (This ordering is obviously only of importance if the windows in which these documents are displayed are to some extent overlapping.)

In addition to these features, the editor has most of the normal text editor features, including the ability to open a file as "read-only." The editor is also Apple Event aware, permitting it to interact with other programs via the mandatory Apple Events that designation implies.

INTIMATE LINKAGE BETWEEN MACINTOSH AND SUN WORKSTATIONS

A principal goal of our development work is creation of a highly interactive modeling, simulation, and analysis environment. To this end, we have developed programs operational on both a Macintosh computer and a user-selectable Sun workstation that provide one or more high-speed ethernet TCP links between the two computers. We are now using these facilities for the editor-lexical analysis linkage. In the future, these same links will carry X-window information between the Sun workstation executing an Anabel model and an X-window on the Macintosh displaying dynamic model graphic output and additionally providing user interaction with that model.[42]

COORDINATION WITH OTHER PROJECTS AND ACTIVITIES

During the past year, RAND held a major conference on variable resolution modeling, documented in Davis and Hillestad, 1992. We believe full object orientation (e.g., including multiple and dynamic inheritance, and overloading of operators and class methods) in a model assists in creating models capable of operation at varying resolutions or capable of interacting with other models of differing resolutions. We therefore continue to explore avenues toward variable resolution while developing RANDSIM and the Anabel language in the hope that developments in variable resolution can be instantiated within the systems we are creating.

A companion project at RAND called "Exploratory Modeling," led by one of the authors (Steve Bankes), is developing innovative ideas for exploring the attributes and behaviors of families of models and creating a computer-based "notebook" recording experiments with models and data bases. See Bankes (1992b) for an overview of the aspirations of this project. We are coordinating with the Exploratory Modeling project so that at some point Anabel classes, objects, and inheritance relationships, as well as code authoring and browsing tools and Graphic User Interface capabilities, might be used to implement Exploratory Modeling ideas in a prototype demonstration system.

To further our goals of administrative and organizational activities that support a more unified modeling, simulation, and analysis environment at RAND, we are working closely with an "Advanced Modeling and Analysis Working Group" (AMAG) that has been recently formed to promote coordination, education, and shared experience among key

[42]As one example of what is possible here, colleague Manuel Carrillo has recently completed a development allowing users working on Macintosh or PC microcomputers to use, via a local network, a variety of UNIX applications running on Sun workstations (e.g., applications such as SAS data-processing and statistical programs, Ingres relational data bases, and the GAMS linear and nonlinear programming optimization algorithms). The work is documented but not yet published.

modelers and analysts at RAND. This group is conducting a series of seminars, demonstrations, and discussions that may lead to greater commonality of methods, tools, and terminology among RAND modelers. We expect Anabel tools and methodologies to be a major source of input to these deliberations and that the result might be broad concurrence among group members regarding the scope and characteristics of a unifying environment that we have called RANDSIM.

NEXT STEPS

Our activities during the coming year are focused on (1) completing the class libraries and other facets of a run-time environment for Anabel programs; (2) completing additional Anabel test programs that will exercise and demonstrate Anabel modeling capabilities; (3) development of cooperating software on both the UNIX and Macintosh platforms to provide extension of Macintosh-based browsing tools based on inheritance relationships among class hierarchies and on relationships among Anabel identifiers and their definition statements; and (4) creating graphic visualization facilities for monitoring the operation of a model, most likely by creating inheritable class libraries of graphic objects having a link with the facilities of the MapView program developed during the past several years at RAND.

The above activities are closely correlated with the second year of the three-year development schedule outlined in the previous section. In particular, they are oriented toward completion of a library of simulation objects, extending hypertext source code and configuration control, and the automatic display and visualization of objects.

Demonstration systems with increasing capabilities are owed to sponsoring clients as contract deliverables at the end of March and September 1993.

In addition to language and user-interface facilities being developed, we continue to participate in the larger discussions and organizational activities that are currently under way regarding the whole RANDSIM modeling and analysis environment.

Appendix A
SOME QUESTIONS AND ANSWERS

In conceiving of a comprehensive environment for modeling and analysis, a significant number of questions arise. We have given considerable thought to the following questions and have developed what we believe are the appropriate answers (although not all our colleagues agree). The tradeoffs they represent and our approach toward their resolution are discussed in what follows. Although many of the questions are very RAND-specific as expressed, we believe they are the same questions that other organizations will ask when contemplating advanced environments. Thus, we provide this material here as a case history of one group's thinking on the matter.[43]

HOW MUCH STANDARDIZATION SHOULD BE IMPOSED ON MODELERS AND ANALYSTS?

A degree of standardization in RAND's modeling environment and preferred modeling language is desirable to obtain the benefits of compatible models and interfaces, to develop a coherent set of training and documentation for model developers, and to develop a community of model developers and users that can communicate effectively with each other. Such standardization, however, restricts an individual researcher's or project's decisionmaking regarding environment or language, at least to the extent of requiring someone to "show cause" for variance. *We believe it is time for more standardization*—for much the same reasons that underlay (at RAND) the adoption of the Macintosh (or PC with Windows), Word, PowerPoint, and Excel as standards for desktop manuscript work.

Standardization would by no means be absolute, however. Researchers requiring specialized languages and tools would continue using them (e.g., the logic programming methods provided by the language PROLOG would not be easy to reproduce),[44] and personal preference is a legitimate consideration in language decisions. Nonetheless, there is little

[43]The views expressed here are those of the authors and not necessarily those of all of our RAND colleagues, with some of whom we in fact have technical disagreements on a few issues. However, the discussion here reflects numerous discussions and informal reviews with more than two dozen colleagues who have worked on a wide variety of modeling and analysis projects or on related computer-science developments.

[44]In this connection, we believe a small study should be conducted to explore the potential of using backward-chaining methods (e.g., those of PROLOG) in separable programs that could be called by larger RAND programs, written in Anabel or other standard languages, as subroutines or object-oriented "methods." This might capture a large fraction of the value of those techniques while encouraging standardization at a higher level. Work of this sort is currently being conducted under Defense Advanced Research Projects Agency (DARPA) sponsorship by colleagues Jeff Rothenberg and Michael Mattock but has not yet been adequately folded into the planning described here.

benefit in further *encouraging* the continuing cacophony of systems, languages, debugging aids, graphic interfaces, and data bases that currently characterize modeling in most other modeling and analysis organizations we are acquainted with. Further, substantial improvements are needed with respect to conducting routine and effective design reviews and producing timely and easily used documentation, challenges that can be addressed only by procedural and organizational standards. Progress on this front can be only partial, because RAND and other comparable organizations will continue to rely heavily on many models developed elsewhere in a variety of languages and styles, but at least in our own work we can move toward greater commonality—*if* the emerging language and tools meet the test of the marketplace!

It should be noted that care must be taken in the definition of standards and that technology must have achieved a level of relative maturity for standardization to succeed. Thus, standardization must be seen as an ongoing process, with standards being defined opportunistically when they become technically feasible and when workers believe they would be useful and largely agree on what they should be. What is important in the near term is to identify areas where standardization is possible and the creation of an organizational mind-set that will result in further standardization as it becomes possible.

CAN RAND ADOPT A COMMERCIALLY AVAILABLE MODELING LANGUAGE AND ENVIRONMENT?

There are many commercial modeling languages and systems available, some of them (e.g., CACI's Simscript and MODSIM) in active use at RAND. With a commercial product (usually) comes documentation; support in the form of a telephone number or "help line"; a user community with possible newsletters, conferences, etc.; new releases and enhancements; and bug fixes. With in-house development of a modeling and analysis environment come few of the above advantages, but instead one achieves control over one's destiny in that features and enhancements can be added and explored without convincing some third party (e.g., a commercial company with many competing priorities and interests) that your needs are paramount and urgent. Although we expect that commercial languages will be used indefinitely at RAND, including within the environment we are describing, we believe there is no commercial modeling and analysis product now available or foreseeable in the near term that incorporates the range of features we feel are highly desirable to RAND excellence in modeling, namely:

- Tools that concentrate not only on programming tasks but also the design of models, exploratory modeling, interaction with running simulations, analysis of results, comparing results across runs and models, and many other activities that characterize modeling, simulation, and analysis at RAND.

- Programming languages that result in programs that are sufficiently "transparent" in their meaning, so that our clients and sponsors can study and possibly modify the contents of models and simulations we develop.
- Ability to create simulations using advanced features such as object feature inheritance without a performance penalty.
- Ability to obtain adequate—one hopes even excellent—performance along various dimensions (e.g., speed, readability of code, ease of programming, graphics front-ends permitting interaction with a running simulation) with a single programming language and development environment, which would help RAND build a pool of programmers that are fungible among projects, and to build a common culture of modeling and simulation with appropriate training, seminars, workshops, and documentation—or even possible sharing of subroutines, software "objects," or code modules.
- Flexibility in permitting graceful extension into new technologies that are becoming available and important, such as object-oriented modeling and programming, use of hypertext and hypermedia, and visualization systems.

We therefore support in-house development of a comprehensive, RAND-wide, computer-based modeling and analysis environment, incorporating auxiliary commercial systems (e.g., for graphic interface display, DBMS) wherever possible. It should be noted that RAND's Anabel development is based on widely, commercially available languages such as C and C++, with links available to modules programmed in other languages such as FORTRAN. In this manner, ties to other commercial products can be maintained.[45] We believe that with this decision and commitment, it will prove possible to make a number of major advances in modern, effective, tailored modeling and analysis tools.

[45]One concern that is sometimes raised concerning the Anabel effort relates to our ability to export programs written in Anabel once it exists. There will be several options: exporting the model without source code, exporting the C code generated by compilation of the Anabel language, or exporting the Anabel source code itself. Most users do not change source code at all, and Anabel programs will be highly portable because of being C/UNIX based. For users wishing to make modest changes, it should be possible to modify the C code directly. For users wishing to read and make more substantial changes of source code, it will be necessary to learn Anabel, just as the dozen or so external users of the RAND Strategy Assessment System (RSAS) have learned RAND-ABEL for some years with little trouble.

SHOULD MILITARY MODELING BE STRESSED OR CAN DOMESTIC RESEARCH NEEDS BE FULLY ACCOMMODATED?

The significant majority of RAND modeling activities support our defense research. Many feel this imbalance is inappropriate; it creates artificial divisions within the RAND research staff based on differing methodologies used and also may lead to missed opportunities in some domestic research projects. The question is: should the next-generation RAND modeling environment be tailored to our dominant military modeling activities or should it deliberately and consciously reach out and provide tailored facilities for domestic research—facilities such as easy access to SAS data bases and statistical analysis packages? Our answer is that *every effort should be made to provide resources that will attract RAND's domestic research toward use of a RAND-wide modeling environment*; this opportunity for greater methodological coherence across the defense-domestic interface at RAND is too important to ignore. As is stated below, at least one domestic project should be attracted as an "early adapter" of the proposed next-generation modeling environment to help represent the interests of domestic research. A deliberate program of introducing modeling and simulation into the Domestic Division should be planned for. There should also be domestic representation on any steering committee or users' committee advising on the development of a next-generation RAND-wide system. (Plans for such steering and technical committees are mentioned as part of the management plan in Section 4.) It will probably be necessary to use discretionary funds to support early modeling activities that appear useful within a sponsored activity that otherwise has no "requirement" or funds for such modeling.

WHAT ABOUT ADA?

The topic of Ada deserves special mention. The DoD has been requiring use of Ada for development of some models and simulations, particularly ones that form a deliverable product resulting from a contract. There are also development environments becoming available that aid in the development of Ada programs. Should Ada be a candidate for our central language? Despite DoD's effort to make Ada the standard, it seems unlikely that it will succeed, at least for the research and analysis communities. One problem is that Ada was finalized and standardized before a number of new programming techniques were popularized—notably full-blown object-oriented methodologies. A second is that Ada is so standardized that we could not explore new modeling techniques and language extensions within its framework. Third, Ada has not stressed transparency or comprehensibility by analysts or sponsors (although it has stressed practices that improve code-level comprehensibility for programmers). In this respect it is not at all competitive with RAND-

ABEL (or other high-level languages). Fourth, good and inexpensive compilers for Ada are not yet widely available. Finally, we note that Ada is *not* being adopted by many workers with a choice (i.e., those other than DoD contractors required to use it), either in the United States or elsewhere. It is not winning in the marketplace. Although it has many good features and would facilitate project management and continued operation and maintenance of software, it is not a good choice for our principal work (although exceptions may exist from time to time). Significantly, our conclusion here is *not* based on unfamiliarity with Ada, because we believe that good programmers can learn new languages, especially well-conceived languages such as Ada, rather quickly (weeks for initial competence, many months for higher levels of competence).

Appendix B
SOME SYSTEM DESIGN CONSIDERATIONS

TOP-LEVEL ARCHITECTURE OF ANABEL ENVIRONMENT

Many design details of the language and environment described in this document will continue to evolve over many years. Nevertheless, it is possible to show a top-level design of major portions of the system envisaged to illustrate the many interconnections and options it will allow. Figure B.1 is a data-flow diagram of the Anabel environment, in which each round-cornered box represents a process and the arrows between processes or data files represent the flow of data. Most arrows are labelled with the type of data being transferred.

A number of points need to be made about this diagram to put it in context. The processes and facilities shown in Figure B.1 will emerge over a three- or four-year period. Not all of the facilities will be available in the initial version of the system.

Figure B.1 represents a view of the system as seen by a user thinking about the wide variety of software tools and processes that can be invoked. For example, if the user is accessing this software through a Macintosh, normal application programs such as Microsoft Excel, MacFlow, and MacDraw Pro are available within windows on the displayed "desktop." The user may also establish an X-Window connection to MapView, a program developed at RAND allowing a wide and flexible variety of map backgrounds to be used as backdrops for graphics generated during the operation of a model. A specially designed editor is available on the Macintosh that allows creation of hypertext links to other Macintosh documents, such as flowcharts, text files, voice annotations, or even QuickTime movies. Those hypertext links may be embedded within Anabel source code, or any text file created within the editor, as a means of documenting the code without detracting from the flow of the logic of the code itself. The editor uses a TCP link to a UNIX server on the local area network to obtain analyses of Anabel source code; the resulting data allow the editor to (1) use distinctive font/style/size/color characteristics to distinguish among keywords, identifiers, constants, and comments; (2) build data structures allowing cross-reference access from identifiers to their declarations and to other occurrences of those identifiers; and (3) browse through class definitions and inheritance relationships. From the editor, the user may also access (by double-clicking on an Anabel keyword within the source code) Anabel reference materials contained in a HyperCard stack. That stack of 230 cards contains information about the syntax and semantics of every valid Anabel statement and language feature.

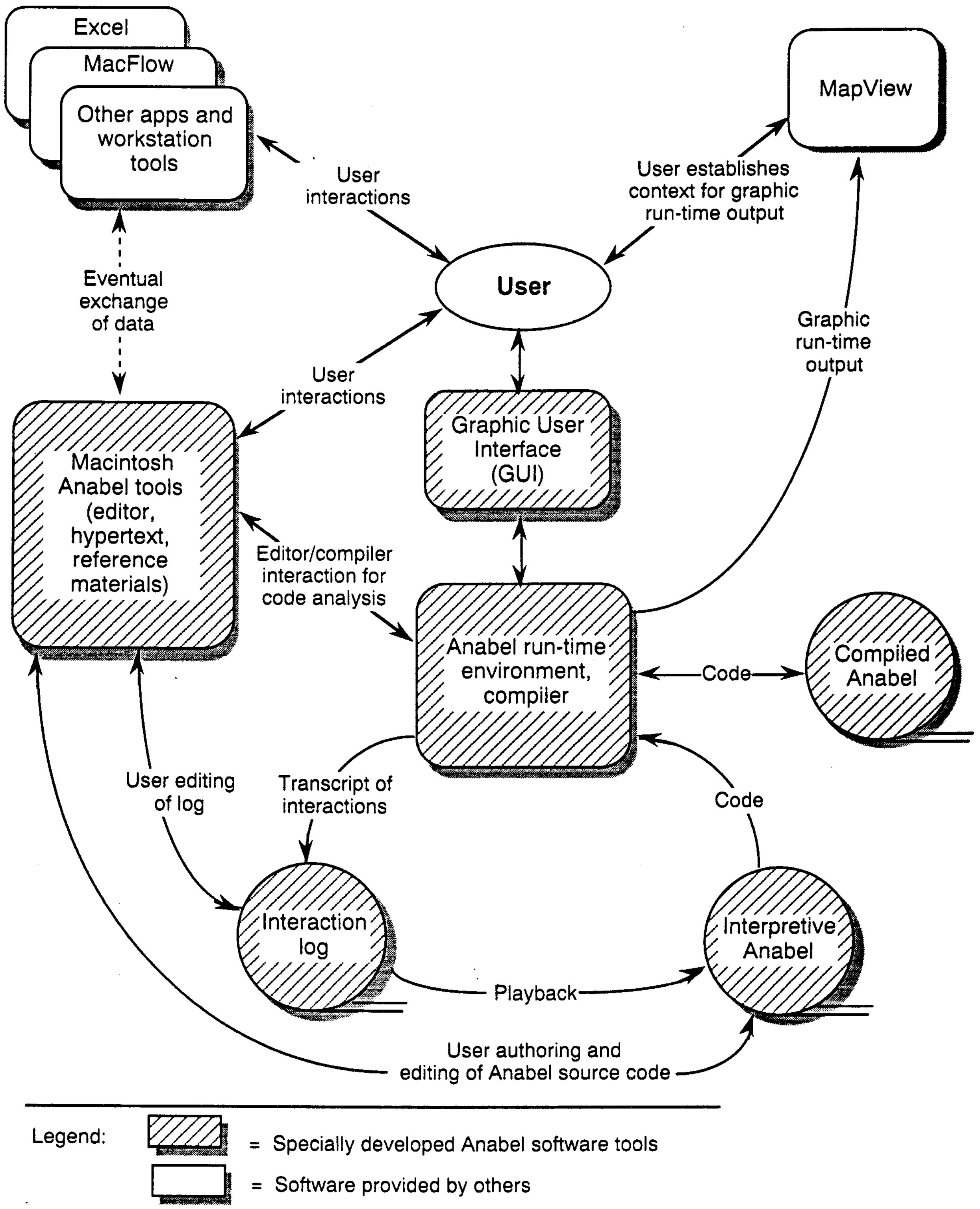
Excel
MacFlow
Other apps and workstation tools
User interactions
User establishes context for graphic run-time output
MapView
Eventual exchange of data
User
Graphic run-time output
User interactions
Graphic User Interface (GUI)
Macintosh Anabel tools (editor, hypertext, reference materials)
Editor/compiler interaction for code analysis
Anabel run-time environment, compiler
Code
Compiled Anabel
User editing of log
Transcript of interactions
Code
Interaction log
Playback
Interpretive Anabel
User authoring and editing of Anabel source code
Legend:
= Specially developed Anabel software tools
= Software provided by others

Figure B.1— Top-Level Logical Architecture of Anabel Environment

A GUI will be provided to the user, allowing access to the Anabel run-time environment and compiler on a UNIX server. This GUI will be designed using X-Window commands for portability, so that it might be accessed from an X-window on a Macintosh, PC, or any POSIX[46]-compliant workstation. During operation of an Anabel model and user interactions with that model, a set of Anabel source statements will be emitted to an interaction log file recording all user interactions. That log file will be human-readable and -editable and can be used to play back a model's execution—including user interactions—at a later time. During a model's execution, it might be designed with appropriate "display" methods associated with its objects to send graphic commands to MapView so that (aspects of) the model's operation can be visualized on a map background. A class library of graphic objects having these properties will be provided as a default "starter set."

Since the ready interchange of Anabel model source code, including associated hypertext linkages, is seen to be crucial to its dissemination and portability, we have chosen to use the ISO Standard Generalized Markup Language (SGML) as its basic internal representation and interchange format (Goldfarb, 1990, serves as our standard reference and includes the ISO Standard).

A number of important facilities are *not* shown in Figure B.1, because they intersect too many of the other functions. For example, the plan includes a comprehensive configuration management facility allowing retention of a record of what data were used with which run of what version of which model to obtain what resulting output script and files—along with annotation by the user indicating reasons for that particular model run and other auxiliary information.

Some of the processes indicated in Figure B.1 will normally run within a user's personal computer (e.g., Macintosh), and others on a UNIX-based workstation—with a high degree of interaction among all these processes. How the processes are allocated among various hardware platforms should be quite flexible.

A number of subsidiary processes and tools to be used in creating this environment are also not indicated in Figure B.1. For example, a C or C++ language compiler will be used as part of the interpreter/compiler process. Such lower-level support tools form an important substratum of the entire system. With a careful choice of these lower-level tools, it is possible to achieve considerable transportability of the system to a variety of hardware platforms as new computing options continue to become available.

[46]POSIX (Portable Operating System Interface for UNIX) is an open system standard based on UNIX.

Figure B.1 provides an overview of a software environment for creating, running, and documenting models written in the Anabel language. However, it does not address the important topic of the possible relationships among this Anabel environment, models written in the Anabel language, models written in other languages (whether written at RAND or imported from other sources), and other non-Anabel environments in which models (written in Anabel or in other languages) are to be run. This is important because other, separately developed models (often imported from other organizations and written in other languages) are vital to RAND analyses.

There are two major cases to be considered, each with subcases and variations of importance. The two cases are the following:

- Allowing *models written in Anabel* to work within *another modeling environment* and to work with other models written in other languages;
- Using *models written in other languages* within *the Anabel modeling environment* and in conjunction with Anabel models operating in that environment.

We discuss each case briefly below. It should be emphasized, however, that there are many difficult issues involved in the cooperative, coordinated use of separately developed models. For example, they may treat geography or time at a different resolution or granularity, they may have differing nomenclatures for identical objects or concepts, or they may aggregate items differently. These are important considerations that we are *not* addressing in our attempt to create aids allowing interconnections to be made among dissimilar models. We realize their importance, but such issues are not being addressed by our current development project.

IMPORTING ANABEL MODELS TO OTHER ENVIRONMENTS

Assume model A is written in the Anabel language and is to be used as a submodel or subprocess within some other model and modeling environment. We think of this as an Anabel model operating within some other paradigm, having other conventions for interprocess coordination or data sharing. (An example of such an "other" paradigm is the SEMINT system being developed at RAND, referenced earlier in this report.)

We plan on providing at least two specific aids to assist in integrating that model A into the other environment:

- Generation of C or C++ code representing Anabel model A, so that, for example, the C/C++ code might be recompiled to work within that other environment. Explicit generation of such C/C++ code may be either a standard feature of the Anabel compiler or an optional feature to be invoked when desired. The resulting C/C++ code will meet relevant ANSI standards for maximum transportability and generality.
- Optional generation of a Data Interface Module (DIM), tailored to that Anabel model A, for use by other models or interfaces and providing a relatively easy means for those models or interfaces to access or manipulate data objects within model A. This DIM may consist of some combination of (1) a data dictionary mapping names of Anabel objects and their attributes and methods within model A to their internal representations within model A and (2) a library of one or more access routines working in conjunction with this data dictionary, with the routine(s) in this library providing access to data within model A. Portions of this Data Interface Module might well be generic and used for all Anabel models for which a DIM is generated, and other portions might be model-specific. These details are yet to be determined.

Figure B.2 shows schematically how an Anabel model might exist within a non-Anabel environment through use of its DIM as an aid to communication with a user-interface program or other models within that environment.

In Figure B.2, the existence of the "non-Anabel models" is optional; the Anabel model with its DIM might well run by itself within the non-Anabel environment, just as any other separately developed model would.

IMPORTING OTHER MODELS INTO THE ANABEL ENVIRONMENT

The second form of cooperation between Anabel models with their supporting environment and externally developed models written in other languages is the importation of those external models into the Anabel simulation and analysis environment. In this case the external model is viewed as a subprocess of the Anabel model—perhaps used, on call, to generate some data as input to the main Anabel model. The external model could be as simple as a spreadsheet model or as complex as a whole detailed combat simulation with high resolution.

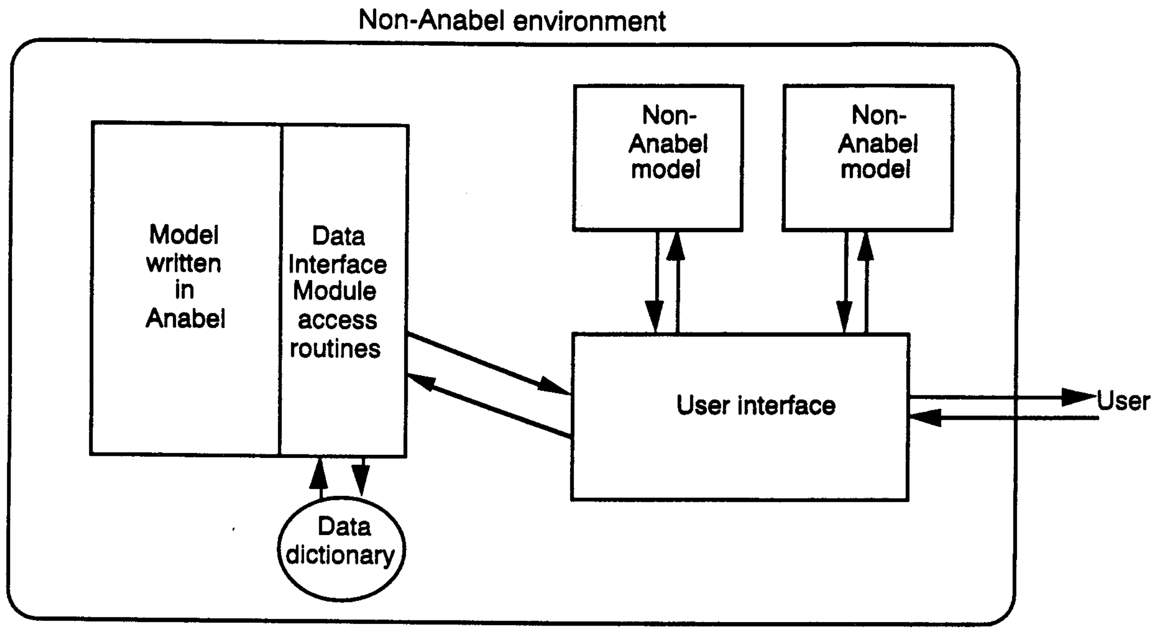

Figure B.2—An Anabel Model Within a Non-Anabel Environment

This form of cooperation has the advantage that all the tools and capabilities of the Anabel run-time environment are available to this consortium of models, such as the automatic creation of a log file of user interactions with the combined model for later study, editing, and/or replay.

We intend to create "wrappers" for these external models in this case that map I/O from the externally developed model into I/O compatible with the Anabel model(s). This "wrapper" is a program that is itself written in Anabel. Figure B.3 shows several such external models interfacing to an Anabel-language model, each through the use of such a "wrapper" program, within the overall context of the Anabel modeling and simulation environment.

The symmetry in Figure B.3 should be noted: Once an external model has been provided with a "wrapper," it becomes just like an Anabel language model and can be used wherever an Anabel model might be, for example, intercommunicating with other Anabel models, with other "wrapped" external models within the Anabel environment, and with the Graphic User Interface.

A special case of the configuration shown in Figure B.3 is especially important at RAND: the ability to run a non-Anabel model all by itself within the Anabel run-time environment. This case is shown in Figure B.4.

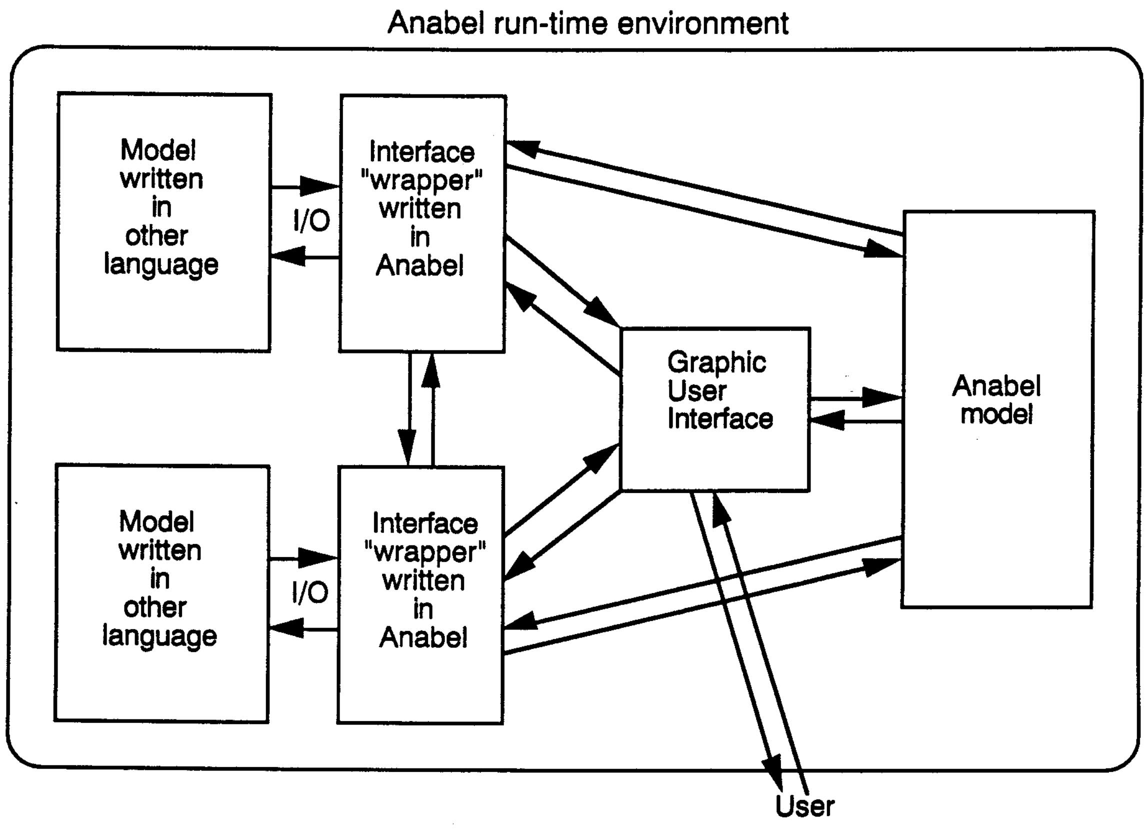
Anabel run-time environment
Model written in other language
I/O
Interface "wrapper" written in Anabel
Graphic User Interface
Anabel model
Model written in other language
I/O
Interface "wrapper" written in Anabel
User

Figure B.3—A Non-Anabel Model Within the Anabel Environment

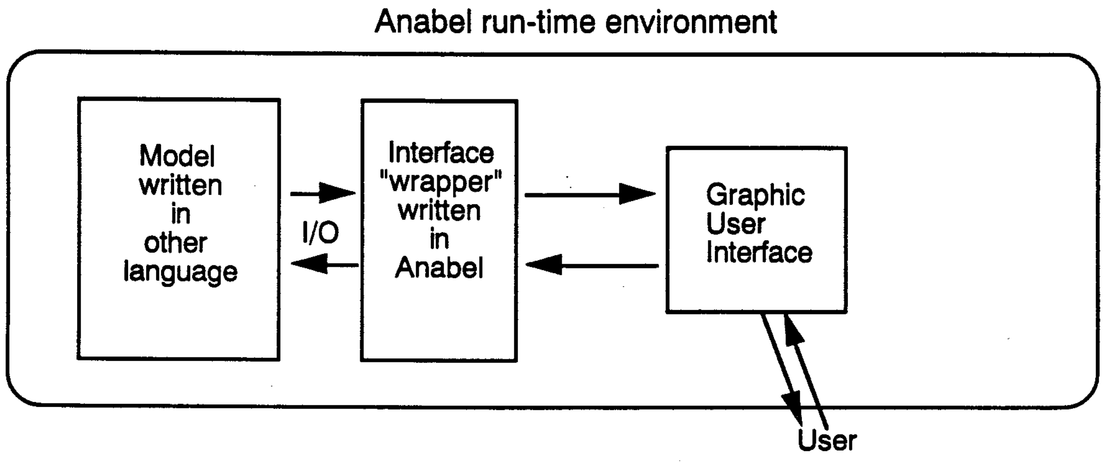
Anabel run-time environment
Model written in other language
I/O
Interface "wrapper" written in Anabel
Graphic User Interface
User

Fig B.4—A Non-Anabel Model by Itself Within the Anabel Environment

The ability to run a non-Anabel model by itself within the Anabel run-time environment is important because, as that model is provided with a "wrapper," that model—even though developed separately, in a different language, and possibly imported from elsewhere—can then benefit from all the resources of the Anabel run-time environment, such as (1) the ability to use interpreted Anabel to write scenarios exercising the model; (2) use of the Anabel Graphic User Interface to interact with the model during its execution; and (3) automatic creation of a log file—in the form of normal Anabel statements—documenting all user interactions with the model, so that this transcript might be edited and/or used to replay a model run.

As part of the long-term Anabel development project, we intend to create standard "wrappers" of this sort for externally developed models in common use within RAND and to provide software tools (such as specialized class libraries) to aid in the development of such "wrappers."

Appendix C
MODEL, DATA, AND MODEL-RUN MANAGEMENT

Model, data, and model-run management will be an essential part of the eventual environment we envision. RAND has been developing related infrastructure for some years now, primarily centered in its Military Operations Simulation Facility (MOSF), which is a laboratory with numerous workstations and a permanent staff who can help analysts with a wide range of problems, such as reviving an old model, establishing network links, exploiting available software tools such as a geographic information system and various visualization aids, or establishing configuration control for a new model development or analysis. Steady progress has been made since the MOSF was established in the mid-1980s. Many model, data, and model-run management methods and tools have also been developed as part of the RSAS (RAND Strategy Assessment System) and its applications, which have often involved multiscenario analysis and complex data bases.

At the same time, there is much yet to be accomplished. We (and the community at large) are a long way from having a highly integrated user-friendly set of management tools. As an example of what may be desirable here, consider Figure C.1, adapted from Bennett (1989). This describes a particular idealized model-integration-and-management system (MIMS). It envisions repositories of models, data, and scenarios. It also envisions tools (including tools incorporating knowledge-based models and intelligent-data-base methods) to assist analysts in drawing upon and tailoring models and data for a particular study and to assist those analysts in designing the experiments and studying the results.[47] Other workers have also examined a number of concepts for related functions and have developed prototype systems to accomplish some of them (see Nance, 1989, for abstracts of work at Virginia Tech).

One of us (see Bankes, 1992b) has argued for computer-based support tools for managing the complexity of an evolving analysis utilizing computer modeling. As a specific approach to managing such complexity, recent research on exploratory modeling by Bankes

[47]Boxes represent information sources, with the analyst and experts being special in this respect. Ellipses are computer processes. Solid arrows indicate data flow, except for dark arrows A, B, D, E, and F, which represent manual intervention, as when the analyst is working within the environment or when the data-base experts are interacting with the data-base templates. Dashed arrows indicate "meta" data describing and interpreting actual data.

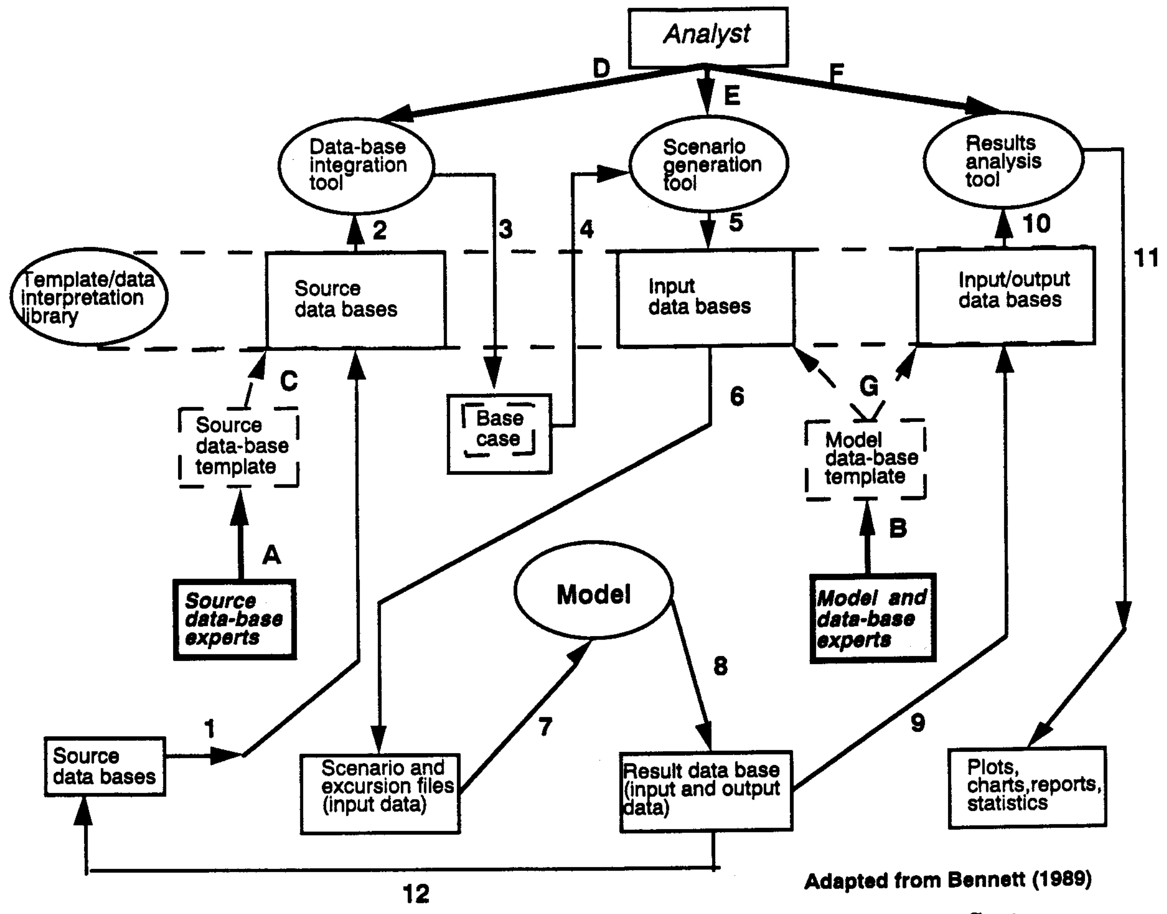
Analyst
D
E
F
Data-base integration tool
Scenario generation tool
Results analysis tool
2
3
4
5
10
11
Template/data interpretation library
Source data bases
Input data bases
Input/output data bases
C
Source data-base template
Base case
6
G
Model data-base template
B
A
Source data-base experts
Model
Model and data-base experts
8
1
7
9
Source data bases
Scenario and excursion files (input data)
Result data base (input and output data)
Plots, charts,reports, statistics
12
Adapted from Bennett (1989)

Figure C.1—An Architecture for a Model Integration and Management System

involves developing the concept of a computer-based "notebook" in which schemas for models may be kept, with other pages recording specific model runs and associated data inputs and outputs. Other pages may summarize results. The entire notebook contents may be electronically searched in some linear fashion, such as by date of experiment, or queried like a data base to find specific information. The exploratory modeling project is cooperating with the development of the Anabel language and its related programming and documentation support tools, with the hope that these projects may form useful cooperative linkages and some common use of software in the future.

BIBLIOGRAPHY

Bankes, Steven C. (1992a), *Issues in Developing the Potential of Distributed Warfare Simulation,* RAND, R-4131-DARPA.

Bankes, Steven C. (1992b), *Exploratory Modeling and the Use of Simulation for Policy Analysis*, RAND, N-3093-A.

Bennett, B. E. (1989), *A Conceptual Design for the Model Integration and Management System,* RAND, N-2645-RC.

Bobrow, D., et al. (1988), "Common Lisp Object System Specification," draft submitted to X3J13.

Cammarata, S., D. H. Shane, and P. Ram (1991), *IID: An Intelligent Information Dictionary for Managing Semantic Metadata*, RAND, R-3856-DARPA.

Coad, Peter, and Ed Yourdon (1992), *Object-Oriented Analysis, 2nd Edition,* Prentice-Hall.

Dahl, Ole-Johan, and Kristen Nygaard (1966), "SIMULA—An Algol-Based Simulation Language," Communications of the Association for Computing Machinery, Vol. 9, No. 9, pp. 671–678.

Davis, M., N. Shapiro, and S. Rosenschein (1982), *Prospects and Problems for a General Modeling Methodology*, RAND, N-1801-RC.

Davis, Paul K. (1988), *Explanation Mechanisms for Knowledge-Based Models in the RAND Strategy Assessment System,* RAND, N-2711-NA.

Davis, Paul K. (1990), *An Analyst's Primer for the RAND-ABEL Programming Language,* RAND, N-3042-NA.

Davis, Paul K. (1992a), *Generalizing Concepts and Methods of Verification, Validation, and Accreditation (VV&A) for Military Simulations*, RAND, R-4249-ACQ.

Davis, Paul K. (1992b), *An Introduction to Variable-Resolution Modeling and Cross-Resolution Model Connection*, RAND, R-4252-DARPA.

Davis, Paul K., and Reiner Huber (1992), *Variable-Resolution Modeling: Motivations, Issues, and Principles*, RAND, N-3400-DARPA.

Davis, Paul K., and Richard Hillestad (1992), "Using Simulation in the Education of General Officers," *Proceedings of the Summer Simulation Conference,* July 1992, Society for Computer Simulation.

Department of Defense (DoD) (1992), *Defense Modeling and Simulation Initiative*, May 1, 1992, issued by the Defense Modeling and Simulation Office of the Director, Defense Research and Engineering.

Goeller, Bruce, S. C. Abraham, A. F. Abrahamse, J. H. Bigelow, J. G. Bolten, J. C. DeHaven, D. L. Jaquette, N. A. Katz, T. F. Kirkwood, R. L. Petruschell, T. Repnau, J. P. Stucker, W. E. Walker, L. H. Wegner (1983), *Policy Analysis of Water Management for the Netherlands: Vol. 1: Summary Report,* RAND, R-2500/1-NETH.

Goldfarb, Charles F. (1990), *The SGML Handbook,* Oxford Press, Oxford, UK, serves as our standard reference (and includes the ISO Standard in its text). Several commercial packages already have SGML capability.

Hillestad, Richard, and Mario Juncosa (forthcoming), *Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models*, RAND, R-4250-DARPA.

Hillestad, Richard, John Owen, and Donald Blumenthal (forthcoming), *Experiments in Variable Resolution Combat Modeling*, RAND, N-3631-DARPA.

Hillestad, Richard, Louis Moore, and Eric Larson (forthcoming), *The Theater Level Campaign Model / Non-Linear Combat Model Toolkit: An Overview,* RAND.

Kipps, James R., Bruce Florman, and Henry A. Sowizral (1987), *The New ROSIE Reference Manual and User's Guide*, RAND, R-3448-DARPA/RC.

Klahr, Philip, and Donald A. Waterman (1986), *Expert Systems—Techniques, Tools, and Applications,* Chapter 3, "Ross: An Object-Oriented Language for Constructing Simulation," pp. 70–91, and Chapter 7, "Twirl: Tactical Warfare in the ROSS Language," pp. 224–268, Addison-Wesley Publishing Company.

Landauer, Christopher, and Kirstie L. Bellman (1992), "Integrated Simulation Environments," *Proceedings of a Conference on Variable- and Cross-Resolution Modeling*, RAND.

Marti, Jed (1988), "RISE: The RAND Integrated Simulation Environment," in Brian Unger and David Jefferson (eds.), *Distributed Simulation*, Simulation Councils, Inc., San Diego, California, Vol. 19, 1988.

Marti, Jed (forthcoming), *RLISP'88: An Evolutionary Approach to Program Design and Reuse*, to be published as a book in 1993. Unpublished versions available through the author at RAND.

Marti, Jed (1990), "Cooperative Autonomous Behavior of Aggregate Units over Large Scale Terrain," in *Proceedings of AI, Simulation and Planning in High Autonomy Systems*, IEEE, Washington, D.C.

Marti, Jed, and Niels Catsimpoolas (1992), "Scripting Highly Autonomous Simulation Behavior Using Case-Based Reasoning," submitted to Society for Computer Simulation, 1992 Simulation Multi-Conference.

Meyer, Bertrand Meyer (1988), *Object-Oriented Software Construction,* Prentice-Hall, Englewood Cliffs, New Jersey.

Military Operations Research Society (1989), *MORS Workshop on Simulation Technology 1997 (SIMTECH 97), Proceedings of Session III*, October. See also earlier volumes on sessions I and II.

Miller, Duncan (1992), "An Introduction to SIMNET," presented at the Military Operations Research Society's Minisymposium on Distributed Interactive Simulation, September 29–October 1, 1992, Alexandria, Virginia. Presumably available from the author at BBN Inc., Cambridge, Massachusetts.

Nance, Richard E. (1987), *The Conical Methodology: A Framework for Simulation Model Development,* Systems Research Center of Virginia Tech, Technical Report SRC-87-001.

Nance, Richard (1989), *Technical Report Abstracts,* Systems Research Center of Virginia Tech (10 January 1989).

Narain, Sanjai (1989), "DMOD: A Logic-Based Event Calculus for Discrete-Event Simulation," in *Proceedings of the Artificial Intelligence and Simulation Conference,* Society for Computer Simulation, San Diego.

Nielsen, J. (1990), *Hypertext and Hypermedia,* Academic Press, London, UK.

Rothenberg, Jeff (1989), *Object-Oriented Simulation: Where Do We Go from Here?* RAND, N-3028-DARPA.

Rothenberg, Jeff (1992a), "Bi-directional Modeling for Integrated Simulation and Planning," *Proceedings of the SCS European Simulation Multiconference*, York, U.K., June 1–3, 1992, pp. 89–93.

Rothenberg, Jeff (1992b), "Using Causality as the Basis for Dynamic Models," *Proceedings of the Third International Working Conference on Dynamic Modeling of Information Systems (DYNMOD-3),* June 9–10, 1992, Delft University of Technology, Delft, The Netherlands.

Rothenberg, Jeff, N. Z. Shapiro, and C. Hefley (1990), *A "Propagative" Approach to Sensitivity Analysis,* RAND, N-3192-DARPA.

Rothenberg, Jeff, Sanjai Narain, Randall Steeb, Charlene Hefley, and Norman Z. Shapiro (1989), *Knowledge-Based Simulation: An Interim Report*, RAND, N-2897-DARPA.

Round, Alfred (1989), "Knowledge-Based Simulation," in *Handbook of Artificial Intelligence,* Vol. IV, Chapter 22, Addison-Wesley.

Rumbaugh, James, Michael Blaha, William Premeriani, Frederick Eddy, and William Lorensen (1991), *Object-Oriented Modeling and Design,* Prentice-Hall, Englewood Cliffs, New Jersey.

Shapiro, Norman Z., H. Edward Hall, Robert H. Anderson, and Mark LaCasse (1985), *The RAND-ABEL Programming Language: History, Rationale, and Design,* RAND, R-3274-NA.

Shapiro, Norman Z., H. Edward Hall, Robert H. Anderson, Mark LaCasse, Marrietta S. Gillogly, and Robert Weissler (1988), *The RAND-ABEL Programming Language: Reference Manual,* RAND, N-2367-1-NA.

SofTech Inc. (1991), *Software Development Plan (SDP) for the Joint Modeling and Simulation System (J-MASS),* prepared for Wright Laboratory, Wright-Patterson AFB, Ohio.

Sowizral, Henry A., and James R. Kipps (1985), *ROSIE: A Programming Environment for Expert Systems,* RAND R-3246-ARPA.

Stroustrup, Bjarne (1986), *The C++ Programming Language,* Addison-Wesley, Reading, Massachusetts.

Weatherly, Richard, David Seidel, and Jon Weissman (1991), "Aggregate Level Simulation Protocol," *Proceedings of the Summer Computer Simulation Conference*, Society for Computer Simulation International, San Diego, California.

Zeigler, Bernard (1990), *Object-Oriented Modeling With Hierarchical, Modular Models,* Academic Press, New York.

Zobrist, A. L., L. J. Marcelino, and G. S. Daniels (1991), *RAND's Cartographic Analysis and Geographic Information System (RAND-CAGIS): A Guide to System Use*, N-3172-RC.

RAND/N-3554-RC