NASA Technical Memorandum 106044

Gas Turbine System Simulation: An Object-Oriented Approach

Colin K. Drummond, Gregory J. Follen, and Charles W. Putt
Lewis Research Center
Cleveland, Ohio

Prepared for the
23rd Annual Pittsburgh Conference on Modeling and Simulation
cosponsored by the University of Pittsburgh, IEEE, ISA, and SCS
Pittsburgh, Pennsylvania, April 30–May 1, 1992

NASA

GAS TURBINE SYSTEM SIMULATION: AN OBJECT-ORIENTED APPROACH

Colin K. Drummond, Gregory J. Follen, and Charles W. Putt
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

A prototype gas turbine engine simulation has been developed that offers a generalized framework for the simulation of engines subject to steady-state and transient operating conditions. The prototype is in preliminary form, but it successfully demonstrates the viability of an object-oriented approach for generalized simulation applications. Although object-oriented programming languages are—relative to FORTRAN—somewhat austere, it is proposed that gas turbine simulations of an interdisciplinary nature will benefit significantly in terms of code reliability, maintainability, and manageability. This report elucidates specific gas turbine simulation obstacles that an object-oriented framework can overcome and describes the opportunity for interdisciplinary simulation that the approach offers.

INTRODUCTION

Gas turbine engine simulations began to appear in elementary form about four decades ago, coinciding with the time that two-spool engines were introduced. At that time, and with fuel-flow control problems as a backdrop, the increased engine technological complexity demanded a more complete understanding of dynamic system behavior, and there was a need for analysis methods (mathematical models and their computer implementation) leading to improved system control and performance (Fawke and Saravanamuttoo (1971)). Simulation contributions to the understanding of dynamic systems are as important today as they were in the 1950's. It is important to note that, historically, advances in simulation technology have very closely followed refinements in dynamic system modeling, the evolution of computer languages, and improvements in computer hardware and operations. In the last two decades, major (real-time) simulation paradigm shifts have been associated with the migration from analog to hybrid computers and then the complete shift to digital computing platforms.

At present, dynamic aircraft engine system simulations are routinely applied in the study of engine operability issues, engine control law development, and real-time simulation (aircraft system control); Khalid (1992) discusses specific examples of current interest. Increasingly severe performance specifications (e.g., efficiency and thrust/weight) for existing engines will continue to increase the already prominent role of dynamic engine system simulations in design studies of derivative engine concepts (French (1982) and Khalid (1992)).

The state of the art of simulation is under pressure to move forward again. Competitive engine design processes for new (complex) systems must reflect technical design requirements beyond the realm of performance (Henderson and Blazowski (1989)). To predict (for example) reliability and stability is to add a new dimension to existing simulation development techniques. Current simulations are derived largely along discipline lines (aerodynamics, structures) in which isolated model development necessarily involves simplifying assumptions. For highly coupled (discipline) problems, there is a high probability that computational complexities are aggravated (rather than resolved) or that the simulation reliability is jeopardized (Denning (1990)). An interdisciplinary approach has been proposed as the necessary change in perspective. Furthermore, parallel and distributed processing capabilities are developing very rapidly (see Holst et al. (1992)), and there has been a related emergence in computer languages; it is imperative that future simulation developers be cognizant of these developments.

We conjecture that traditional simulation strategies must be relinquished if a significant advance in engine simulation is to be realized. We propose that an object-oriented simulation approach is a unique (and natural) way of looking at aircraft engine simulation which has the potential to successfully provide a meaningful, yet economical, framework for interdisciplinary system analysis. Such an approach is being undertaken as part of the Numerical Propulsion System Simulation project (see Claus et al., (1991)). This paper describes several aspects of this aircraft engine simulation that are being explored with the object-oriented approach.

SIMULATION DEVELOPMENT PROCESS

Several aspects of the typical simulation development process are motivating factors to consider for an object-oriented approach to a gas turbine engine simulation. In the discussion that follows, some obstacles associated with a typical component-level model simulation are described first. Then, on the basis of that discussion, three fundamental limitations to current simulation practice are presented. In the last section of this paper, key features of the object-oriented programming framework are discussed that could mitigate the simulation limitations (experiences from the prototype gas turbine simulation are included wherever possible). Although the discussions in the present work reflect the authors' experience with aerothermodynamics and controls applications, the arguments often extend to structural simulations.

Overview of the Simulation Development Process

Gas turbine engine simulations (a computer solution to a mathematical representation of the engine cycle) are normally intended to mimic dynamic or steady-state engine behavior. Figure 1, which is based on the work of Szuch et al. (1982), illustrates seven key steps in the simulation development process. The first step, formulation of the mathematical model, involves the appropriate application and tailoring of conservation laws (discipline specific) to the perceived system attributes (physics); the complete mathematical method development includes equation solver strategies. After mathematical methods are established, data must be prepared that reflects specific engine design detail and operating environment characteristics. The next step, implementation, links the methods and data by creating a computer code based on a computer language (syntax). Implementation expectations are highly coupled to the formulation strategy.

The fourth step in the process, simulation evaluation and validation, compares the computed results with the design/performance data. Inevitably, a discrepancy between the calculated response and the data occurs; this is usually attributable to either an error in the mathematical method or in the formula translation. Again, with figure 1 in mind, a modification is usually necessary (to resolve the error); and the assessment of what is required may take the analyst back to either the formulation or implementation stage. Once the simulation results have been validated, documentation of the simulation design (and related methods) then takes place. Finally, if, for example, the effect of design changes are of interest, the simulation development process just completed is repeated. Often the simulation framework is intended to be robust and applicable to a new engine with minimal effort.

Some Component-Level Model Attributes

Gas turbine simulation design usually evolves from a convenient and natural decomposition of the engine according to component functions (component level model); for example, an engine is represented as the assembly of a compressor, burner, turbine, and interconnecting ducts. This can be illustrated by the turbofan engine schematic shown in figure 2, which has the component level model representation shown in figure 3.

Mathematical component models are based on the application of conservation laws (discipline specific) to a given component, say, the compressor. As a result, the aerothermodynamics or the structural mathematical description of the component is obtained. Extensive examples of this approach for real-time (or realistic time) simulation are presented in the references; a sample of simulations for gas turbine aerothermodynamic behavior and control characteristics for a variety of computer platforms can be found in the work of Ballin (1988), Daniele et al. (1983), Drummond and Ouzts (1989), Mihaloew and Hart (1978), Mihaloew et al. (1981), Schuerman et al. (1977), Sugiyama (1979), and Sugiyama et al. (1989). Although this list is far from complete (and underemphasizes structural simulations), a commonly recurring simulation theme (not aerothermodynamic specific) is the need to develop a tractable mathematical cycle representation commensurate with the system fidelity of interest and the capabilities of the hardware platform (this is true for both real- and non-real-time simulations.

For brevity, specific remarks on algorithms, modeling techniques and assumptions for each component are not presented here. For example, comments on generic nozzle behavior or compressor characteristics can be found in Cohen et al. (1987) or Mallinson and Moyes (1960).

In the present work it is more relevant to summarize three simulation obstacles associated with complex systems for which object-oriented programming can provide some relief. They are component performance representations, intercomponent coupling, and interdisciplinary coupling. A fourth (recurring) simulation issue, initialization and balancing, is important, but programming experience has not confirmed any object-oriented benefit.

Although the current work has an implied interest in aircraft gas turbine engine analysis, many of the remarks are directly applicable to stationary power plants—for which forward motion and altitude are not central concerns, Cohen et al. (1987). Techniques for stationary power plant analysis can clearly benefit aircraft engine analysis (see, for example, Schoberi (1986)), and the reverse is also true.

Three Obstacles to Complex System Simulation

Component performance representations.—A major task in turbofan engine modeling is predicting the aerothermal performance of the major components of the engine. An acceptable compromise to the description of turbofan component performance is a representation based on nondimensional analysis. This approach yields multivariate component "maps" that detail base component performance over a wide range of operating conditions.

The nondimensional performance map is a relatively straightforward and intuitively pleasing approach to turbofan component performance modeling. However, as a practical modeling technique, the approach has some significant drawbacks. Traditional component maps are difficult to use for new data and/or for component sizing studies. Furthermore, modeling of component off-design performance can require additional maps that consume large amounts of computer memory. A key feature of hybrid simulation was the storage of component maps on the digital computer and the communication with and solution of the differential equations on the hybrid computer (in addition, of course, to the real-time capabilities of the hybrid computer).

The drawbacks in the traditional turbofan component performance maps led to the development of alternative methods of modeling component performance. Converse and Griffin (1984) present a "backbone" performance-fitting technique that is based on the physics of the component rather than on the curvefits of nondimensional parameters. The beauty of the approach is compromised by its complexity (of course, this diminishes considerably as the user becomes familiar with the technique).

Performance maps and backbone representations are appropriate techniques for the scope of many current and future simulation efforts. However, they represent static descriptions of component performance and would create difficulties in developing, for instance, a high-fidelity dynamic system simulation study of rotating stall. Because this specific example is an active area of research, work-arounds (in the form of modeling) are beginning to emerge. Nevertheless, the traditional focus on aerothermodynamic (or any isolated discipline) performance precludes interdisciplinary system simulation. The role of component representations of a more fundamental nature (based on theories of fluid mechanics or elasticity) is becoming apparent.

Currently, an approach to component representation based on first principles is (in principle) feasible, but cannot be (in practice) accomplished on a typical (single processor) simulation computing environment. As mentioned before, preexisting implementation expectations hinder the formulation of the problem unless implementation capabilities change.

Intercomponent coupling.—A corollary to the task of predicting individual component performance is the matching of component operating conditions such that basic conservation laws are obeyed along the gas flowpath. One approach to this problem is to explicitly invoke conservation laws in the form of differential equations relating the flowpath conditions between major components. With figure 3 in mind, the usual approach is to place intercomponent volume dynamic elements between component performance modules.

Although the inclusion of explicit intercomponent volume dynamics equations would result in a higher frequency (50- to 100-Hz) gas generator model (see Seldner et al. (1972), or Schuerman et al. (1977)), such a model is not always required by the simulation task. Furthermore, a high-frequency model may result in dynamic (numerical) instabilities that require an additional computational burden (i.e., smaller time steps) or appropriate dynamic damping to resolve. The iterative solution algorithm capabilities associated with digital machines resulted in the omission of volume dynamics for simulations providing 10-Hz fidelity; the overwhelming "need for speed" in many simulation systems has had a direct affect on solution algorithms. Although such an approach fits the intent of the original simulation, two significant hazards are that the system fidelity is not easily generalized, and modifications of the original configuration are very difficult to implement. Clearly, assumptions about intercomponent coupling lead to limitations on code reusability and a rigid adherence to procedural code development.

Interdisciplinary coupling.—Most physical processes involve some coupling between scientific disciplines, but in simulation the issue is the extent of the coupling and to what degree the coupling influences dynamic system behavior. This is not to trivialize the issue of interdisciplinary coupling because, for instance, flutter (aerodynamics coupled with structural dynamics) has been a challenge of long-standing interest and of technical relevance to flight safety (Carta (1989)). More commonly occurring are the interdisciplinary simulations of control systems and aerothermodynamic cycle representations for the development of integrated flight/propulsion control systems (Akhter et al. (1989)). These kinds of systems benefit from the ability to match time scales between the disciplines very closely. Aeroelasticity problems span a larger continuum of time and length scales than aerothermodynamic problems do.

Sobieszczanski-Sobieski and Chopra (1990) remark that (albeit related to multidisciplinary optimization) "major new aircraft design projects have become fewer and farther apart in time, hence past experience becomes less useful as a guide in making design decisions." New systems will require enabling technology development. Furthermore, these authors suggest that "advanced aircraft ... performance hinges on a myriad of numerical interplays, some of them very subtle and beyond the power of human judgment to evaluate precisely." Denning (1990) makes similar comments in his more general discussion of modeling reality. Interdisciplinary simulations demand a quantum change in simulation perspective.

Consider an engine simulation intended to incorporate compressor blade flutter. Figure 4, taken from Carta (1989), presents the interdisciplinary nature of the problem very clearly. Aeroelasticity transforms what

was once an initial-value problem into what is now a combined initial-boundary-value problem. Some fundamental implementation issues are (1) the "raw" number crunching capability required to solve unsteady aerodynamics and structures problems simultaneously, (2) mismatched fidelity of the simulation modules, (3) the development of effective means to introduce geometric data into the simulation, and (4) software manageability.

Summary of Simulation Obstacles and Limitations

Numerous obstacles and limitations to existing simulation approaches have been provided in the previous narrative, but the following summary will clearly show the potential of an object-oriented framework to resolve these problems:

(1) Procedural code structures predominate existing (large-scale) simulation codes, and the ensuing approaches are constrained to be either general, simple, or accurate. However, a simulation that is any combination of these is not currently available (in the public domain). Work-arounds to simulation constraints usually involve some compromise whereby, for example, diminished model fidelity is exchanged for increased simulation execution speed, or where simulation generality (flexibility) reduces program simplicity.

(2) Discipline isolation is relatively predominant for the "usual" simulation. Interactions between disciplines such as aerodynamics, structures, and controls are actually fairly limited during mathematical modeling of component characteristics. In addition, the simulation must deal with a continuum of time and length scales and must introduce component geometric characteristics in a manageable fashion.

(3) Simulation languages and architectures tend to assume a single-processor hardware environment that impedes software portability to modern parallel and distributed computing environments. This assumption leads to a "strategic fit" philosophy in which simulation methods are not designed to exceed *perceived* implementation limitations.

In conjunction with these realities, traditional (digital) simulation developers find it impossible to resist customizing code and tailoring solution techniques to the specific problem and performance window at hand. This tendency has far-reaching effects on manageability, affecting the documentation, reliability, maintainability, and reusability of the subsequent code. Application-driven efforts require flexibility in the multifidelity simulation.

OBJECT-ORIENTED APPROACH

We propose that the object-oriented approach to simulation has the potential to overcome many of the simulation obstacles just discussed. In the present work, there has been an intentional emphasis on leading-up to and describing the simulation obstacles in order to make the value of an object-oriented framework easier to state. In the present section we provide a somewhat terse description of what an object-based view is; the discussion makes use of and benefits considerably from the report by Holt and Phillips (1991). To be sure, there are numerous other papers and books dealing with the subject of object-oriented programming (for example, Shaw (1992), Smith (1991), Steele (1984), Booch (1986), and Flamig (1991)), but the work of Holt and Phillips (1991) deals explicitly with an object-based implementation of the Digital Computer Program for Generating Dynamic Turbofan Engine Models (DIGTEM, Daniele et al. (1983)) gas turbine engine code.

A discussion of an object-based framework frequently leads to two questions:

(1) How is the object-based approach different from a subroutine-based FORTRAN program?

(2) Why do we need another programming language?

To answer these questions it might be more appropriate to change the questions and ask (Schoeffler (1992))

(1) Have you ever heard of a library of subroutines that was credited with helping to solve a software problem?

(2) What needs to be changed in current simulation practice?

The literature suggests that, although *in principle* the answer to the first question might be "yes," *in practice* the answer (for large-scale simulations) is "no." It is beyond the scope of this paper to argue further. A more appropriate focus is on the second question; from a programming perspective the response is (again, Schoeffler (1992))

(1) Structure, structure, and more structure

(2) Modules that can be used like building blocks

(3) Modules that are application oriented

(4) Modules that can be reused

(5) Modules whose source code need not be changed for them to be reused

Reusability is a central issue (Meyer (1987)). Many conveniences associated with the "usability" of FORTRAN hinder, in the context just described, its "reusability." Furthermore, a computer language that had the attributes in this list would be the desired language for simulation—object-oriented languages appear to fit the requirements; specific features supporting this notion follow.

Fundamental Object Representation—Classes and Inheritance

What is an object? Objects are defined in terms of classes. A class is a group of objects that have the same attributes (such as length, area, and inlet pressure) and methods (such as equations for state variable time derivatives). The individual values of the attributes of a particular class are set by creating an instance of the class. For example, figure 3 illustrates that there are several intercomponent ducts in the engine. The unique values of the duct's attributes are what distinguishes one duct from another. They are all members of the class of ducts and thus share the same analytic model.

Different classes of objects can also share common methods and attributes through a mechanism called inheritance. For example, a variable compressor inherits most of its definition from the class of compressors. In turn, compressors are a kind of rotating part and thus inherit behavior from the class of rotating components. The goal of this approach is to eliminate redundant code development and maximize the generality of the model.

With this general approach in mind, it is necessary to look for a generic starting point to define the system. In the prototype engine, the cycle, the most general object is a fixed control volume. From the control volume, components and mixing volumes are established. Components are associated with real physical entities (such as the compressor or turbine) that transform energy from one form to another. Unlike mixing volumes, no energy can accumulate within the control volume. An illustration of the generic mixing volume and the

component control volume are given in figure 5. A more detailed description of the generic mixing volume concept is given in Holt and Phillips (1991).

Connector Groups

Although the components and mixing volumes are the fundamental building blocks of the simulation system, connector groups are the means by which components and mixing volumes communicate with one another. Connectors are represented as objects in the system. In the prototype simulation, key connector groups defined are parameter groups, zoom processors, and feedback connectors. Parameter connectors are a means to communicate individual parameters of a particular discipline between components and mixing volumes. A zoom processor connects component models with differing fidelities. Feedback connector groups permit the creation of closed-loop systems.

Parameter connectors allow for convenient interdisciplinary system definitions. Zoom processors assist in creating system simulations that accommodate a variety of length and time scales (recall the mismatched fidelity issue discussed earlier).

Gas Turbine Hierarchy

Consider an object-based view of the gas turbine model shown previously in figure 2. Each component of the system (e.g., the fan, compressor, and combustor), as well as the generic mixing volumes, can be represented as an object. Each object has characteristics (like state variables) and functions (equations for state variable time derivatives) that are combined to create a complete definition of the object. In general, when surveying a gas turbine, anything that is worth talking about is probably an object (object-oriented modeling is a way of modeling reality).

The result of connecting the components and mixing volumes together with connector groups is a system. However, it is necessary to bring these objects together under an umbrella system object. The system defined by connecting components and mixing volumes together is strictly a static description of the system. It is necessary to define the system object (which contains the components, mixing volumes, and connector-groups) in order to provide the actual simulation methods (steady-state or transient).

A class hierarchy, shown in figure 6, was created to provide a general framework for simulation model development. The framework can accommodate simulation models with varying levels of fidelity and for many disciplines of analysis.

Prototype Simulation

A nonproprietary engine model, DIGTEM (Daniele et al. (1983)), was selected for decomposition and implementation in the Common Lisp Object System. A graphical user interface was developed to simplify the creation and execution of the system.

An extensive validation effort went into the original DIGTEM model, so the relative success of the object-based implementation is manifest in the lack of difference between the predicted original and Lisp state-variable profiles. The icon-based, graphical user interface simplifies system model development, and the user need not modify any source code to create simulations of new configurations (really).

CONCLUDING REMARKS

The Lisp object-oriented approach to the dynamic engine represents a major paradigm shift for gas turbine system simulation; specific benefits are

1. Object-oriented code modularity is amenable to distributed- or parallel-processing hardware platforms.

2. Methods and data are more closely related, and a rational hierarchy exists for the gas turbine system.

3. Strict (and enforceable) code syntax improves code maintainability and reusability.

We propose that this approach to code development, when executed on parallel or distributed processing environments (with the appropriate operating systems), now provides a realistic basis on which to explore simulations with subsystem component modules of differing fidelity (i.e., different length and time scales). This process of "zooming" (entertaining various levels of fidelity with a given calculation sequence) holds great promise for those dynamic simulations where, for example, performance at "out-of-range" design conditions are unknown or where a new compressor model behavior is of interest in situ.

Although the prototype mentioned in this work is, in fact, a prototype, it nonetheless has been instrumental in successfully demonstrating the salient features of an object-oriented perspective.

REFERENCES

Akhter, M.M., et al.: Simulation Development for US/Canada ASTOVL Controls Technology Program. Proceedings of the Twentieth Annual Pittsburgh Conference on Modeling and Simulation, W.G. Vogt and M.H. Mickle, eds., Instrument Society of America, Research Triangle Park, N.C., 1989, pp. 2083-2011.

Ballin, M.: A High-Fidelity Real-Time Simulation of a Small Turboshaft Engine. NASA TM-100991, 1988.

Booch, G.: Object-Oriented Development. IEEE Trans. Software Eng., vol. SE-12, no. 2, 1986, pp. 211-220.

Carta, F.: Aeroelasticity and Unsteady Aerodynamics. Aircraft Propulsion Systems Technology and Design, G.C. Oates, ed., AIAA, Washington, DC, 1989, pp. 390-391, 394.

Claus, R.W., et al.: Numerical Propulsion System Simulation. Comput. Syst. Eng., vol. 2, no. 4, 1991, pp. 357-364.

Cohen, H.; Rogers, G.F.C; and Saravanamutto, H.I.H.: Gas Turbine Theory. Longman Scientific, New York, 1987.

Converse, C.L.; and Griffin, R.G.: Extended Parametric Representation of Compressor Fans and Turbines. NASA CR-174645, 1984.

Daniele, C.J., et al.: Digital Computer Program for Generating Dynamic Turbofan Engine Models (DIGTEM). NASA TM-83446, 1983.

Denning, P.J.: Modeling Reality. Am. Sci., vol. 78, 1990, pp. 495-498.

Drummond, C.K.; and Ouzts, P.J.: Real-Time Simulation of an F110/STOVL Turbofan Engine. NASA TM-102409, 1989.

Fawke, A.J.; and Saravanamuttoo, H.I.H.: Digital Computer Methods for Prediction of Gas Turbine Dynamic Response. SAE Paper 710550, 1971.

Flamig, B.: Turbo C++—A Self-Teaching Guide. Wiley and Sons, New York, 1991.

French, M.W.: Development of a Compact Real-Time Turbofan Engine Dynamic Simulation. SAE Paper 821401, 1982.

Henderson, R.E.; and Blazowski, W.S.: Turbopropulsion Combustion Technology. Aircraft Propulsion Systems Technology and Design, G.C. Oates, ed., AIAA, Washington, DC, 1989, pp. 105-160.

Holst, T.L.; Salas, M.D.; and Claus, R.W.: The NASA Computational Aerosciences Program—Toward Teraflops Computing. AIAA Paper 92-0558, 1992.

Holt, G.; and Phillips, R.E.: Object-Oriented Programming in NPSS. Phase II Report. General Electric Co., Lynn, MA, NASA Contract NAS3-25951, 1991.

Khalid, S.J.: Role of Dynamic Simulation in Fighter Engine Design and Development. J. Propul. Power, vol. 8, no. 1, Jan.-Feb. 1992, pp. 219-226.

Mallinson, D.H.; and Moyes, S.J.: Turbine Power Plants. Design and Performance of Gas Turbine Power Plants, W.R. Hawthorne and W.T. Olsen, eds., Princeton University Press, Princeton, NJ, 1960, pp. 504-534.

Meyer, B.: Reusability: The Case for Object-Oriented Programming. IEEE Software, vol. 4, Mar. 1987, pp. 50-64.

Mihaloew, J.R.; and Hart, C.E.: Real-Time Digital Propulsion System Simulation for Manned Flight Simulators. NASA TM-78958, 1978.

Mihaloew, J.R.; Roth, S.P.; and Creekmore, R.: A Real-Time Pegasus Propulsion System Model for VSTOL Piloted Simulation Evaluation. AIAA Paper 81-2663, 1981.

Schoberi, T.: A General Computation Method for Simulation and Prediction of Transient Behavior of Gas Turbines. ASME Paper 86-GT-18D, 1986.

Schoeffler, R.: Concepts and Applications of Object-Oriented Programming. NASA Lewis Technical Seminar, Cleveland, OH, Mar. 18, 1992.

Schuerman, J.A.; Fischer, K.E.; and McLaughlin, P.W.: High Frequency Dynamic Engine Simulation. (PWA-5543, Pratt & Whitney Aircraft Group; NASA Contract NAS3-20292) NASA CR-135313, 1977.

Seldner, M.; Mihaloew, J.R.; and Blaha, R.J.: Generalized Simulation Technique for Turbojet Engine System Analysis. NASA TN D-6610, 1972.

Shaw, R.H.: Anatomy of a Utility: Writing applications With C++. PC Magazine, vol. 4, Feb. 25, 1992, pp. 361-370.

Sobieszczanski-Sobieski, J.; and Chopra, I.: Multidisciplinary Optimization of Aeronautical Systems. J. Aircr., vol. 27, no. 12, 1990, pp. 977-978.

Smith, J.T.: C++ For Scientists and Engineers. Intertext Publications, New York, 1991.

Steele, G.: Common LISP: The Language. Digital Press, Bedford, MA, 1984.

Sugiyama, N.: Real-Time Simulation of Jet Engines. Applications Report, Applied Dynamics International, 1979.

Sugiyama, Y.; Tabakoff, W.; and Hamed, A.: J85 Surge Transient Simulation, J. Propul. Power, vol. 5, no. 3, 1989, pp. 375-381.

Szuch, J.; Krosel, S.M.; and Bruton, W.M.: Automated Procedure for Developing Hybrid Computer Simulation of Turbofan Engine. NASA TP-1851, 1982.

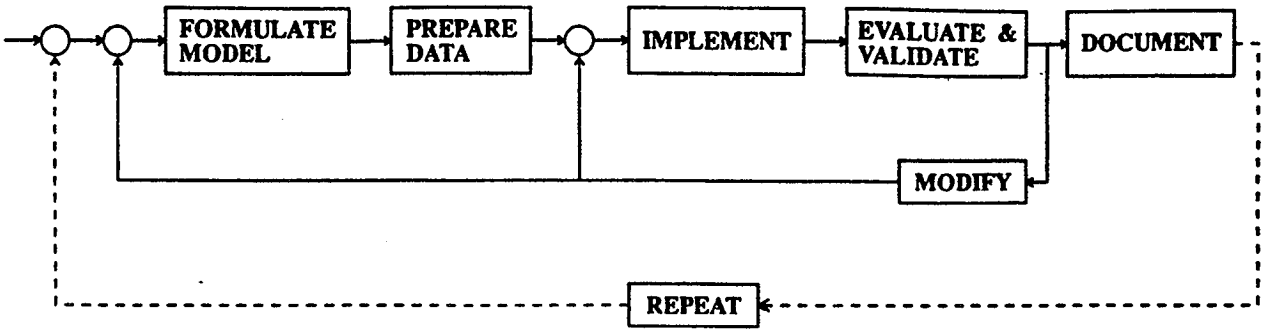

Figure 1.—Simulation development process.

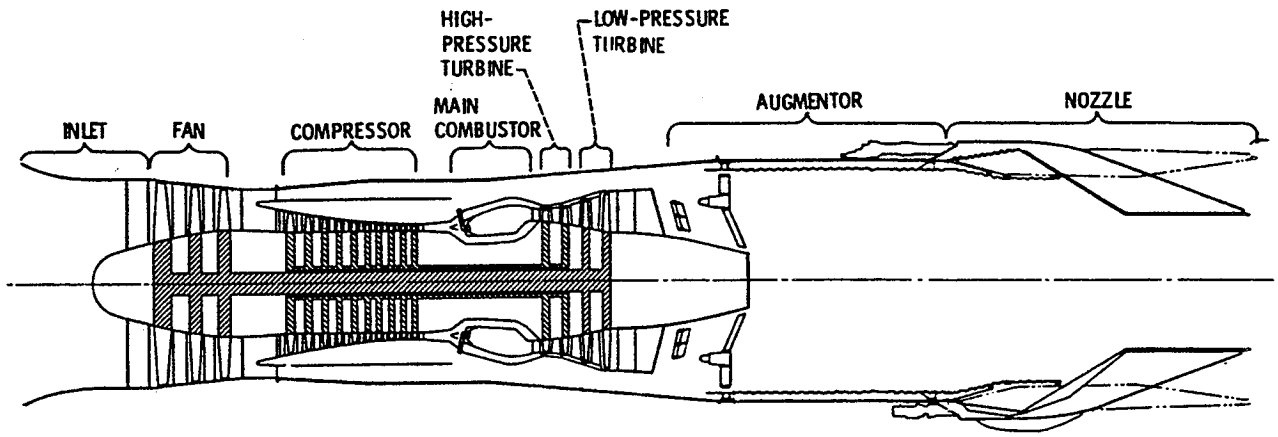

Figure 2.—Generic turbofan flowpath.

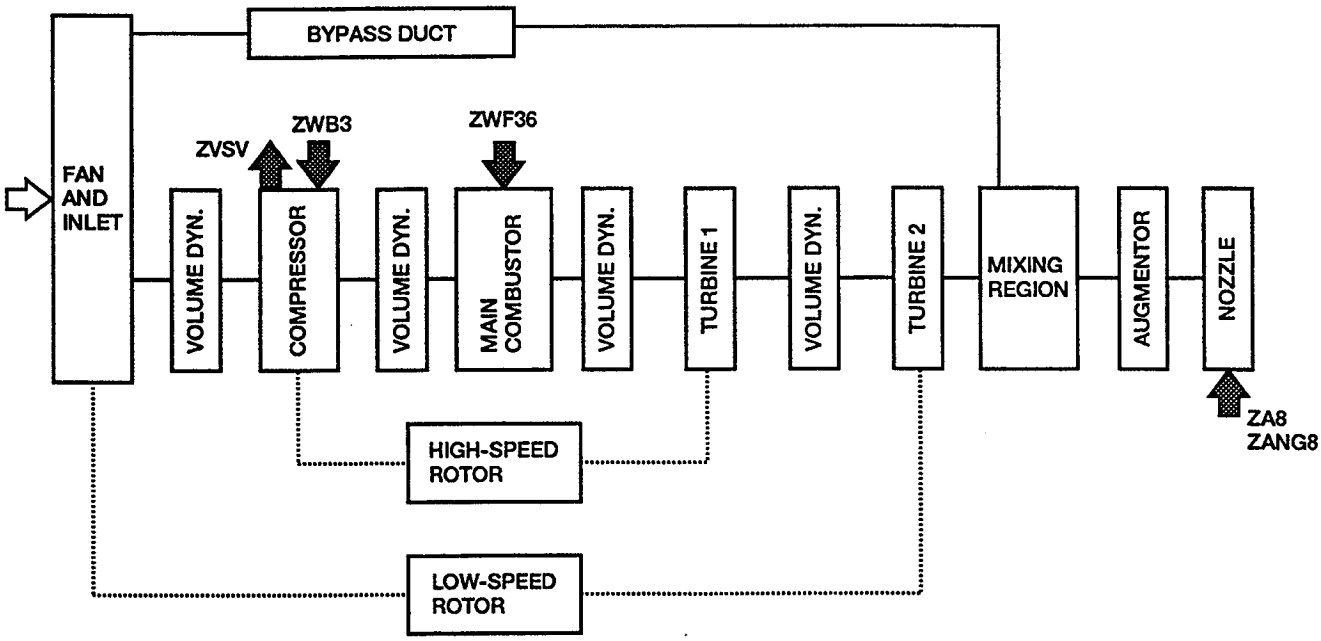

Figure 3.—Baseline propulsion system component level model. Variables beginning with Z are control variables.

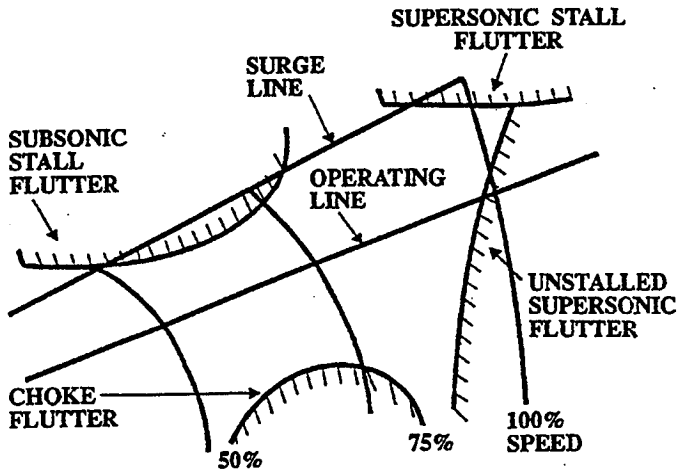

Figure 4.—Compressor map showing flutter boundaries (from Carta (1989))

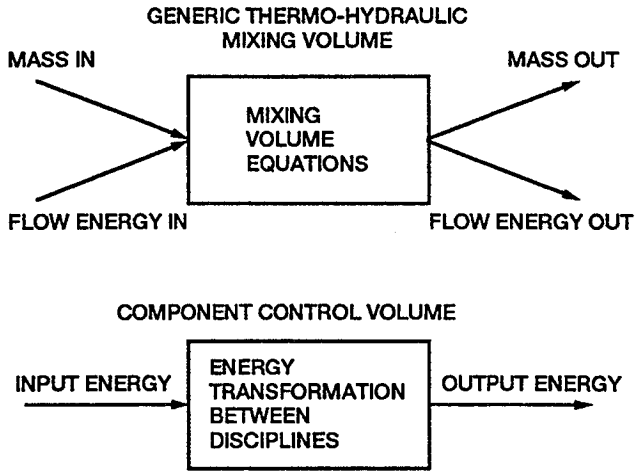

Figure 5.—Generic mixing volume and component control structure.

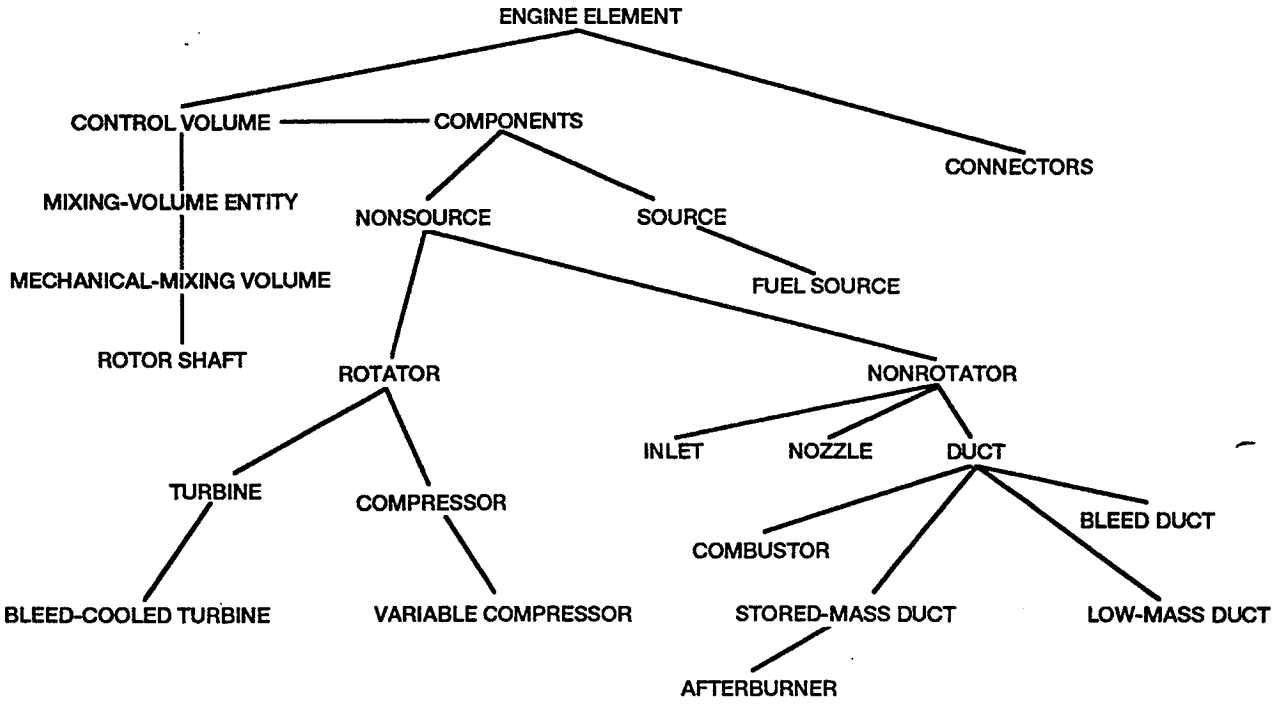

Figure 6.—Numerical Propulsion System Simulation object structure.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1993	Technical Memorandum

4. TITLE AND SUBTITLE

Gas Turbine System Simulation: An Object–Oriented Approach

5. FUNDING NUMBERS

WU–505–62–51

6. AUTHOR(S)

Colin K. Drummond, Gregory J. Follen, and Charles W. Putt

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–7632

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM–106044

11. SUPPLEMENTARY NOTES

Prepared for the 23rd Annual Pittsburgh Conference on Modeling and Simulation, cosponsored by the University of Pittsburgh, IEEE, ISA, and SCS, Pittsburgh, Pennsylvania, April 30–May 1, 1992. Responsible person, Colin K. Drummond, (216) 433-3956.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 07

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

A prototype gas turbine engine simulation has been developed that offers a generalized framework for the simulation of engines subject to steady-state and transient operating conditions. The prototype is in preliminary form, but it successfully demonstrates the viability of an object-oriented approach for generalized simulation applications. Although object-oriented programming languages are–relative to FORTRAN–somewhat austere, it is proposed that gas turbine simulations of an interdisciplinary nature will benefit significantly in terms of code reliability, maintainability, and manageability. This report elucidates specific gas turbine simulation obstacles that an object-oriented framework can overcome and describes the opportunity for interdisciplinary simulation that the approach offers.

14. SUBJECT TERMS

Simulation; Real-time simulation; NPSS; Object-oriented programs; Propulsion systems

15. NUMBER OF PAGES
14

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

National Aeronautics and
Space Administration

Lewis Research Center
Cleveland, Ohio 44135