PERGAMON

Applied Thermal Engineering 21 (2001) 111–118

APPLIED THERMAL ENGINEERING

www.elsevier.com/locate/apthermeng

Technical note

# Extensible object model for gas turbine engine simulation

Zhiwu Xie*, Ming Su, Shilie Weng

*School of Power and Energy Engineering, Shanghai Jiao Tong University, Shanghai, 200030, People's Republic of China*

Received 10 March 1999; accepted 21 December 1999

**Abstract**

This paper presents an extensible object model for gas turbine engine performance simulation. The extension method for gas path balancing is analyzed and a new design rationale is developed to overcome deficiencies of the traditional component-based object modeling method. A class framework implementing this rationale is described and the dynamic performance of a three-shaft gas turbine engine is simulated to evaluate the model's effectiveness. © 2000 Elsevier Science Ltd. All rights reserved.

*Keywords:* Gas turbines; Computer simulation; Object-oriented programming

## 1. Introduction

Computer simulation plays a pivotal role during the development, testing, and verification of gas turbine engines. The size and complexity of a simulation package may vary from calculating a few thermodynamic functions to coupling lots of computational fluid dynamics (CFD) software into an ordinary differential equation (ODE) solver. As long as the engine's conceptual and technological innovations are emerging, the package needs to be extended, modified or even rewritten from time to time. Although software extension is the most economical way, the extendability, which is defined as "the ease with which a system or component can be modified to increase its storage or functional capacity" [1], is restricted not only by the size and complexity, but also by the structure of the software.

* Corresponding author.
*E-mail address:* huangls@wuhan.cngb.com (Z. Xie).

PII: S1359-4311(00)00007-7

Many attempts [2–7] have been made to tackle this problem by applying the so-called "object-oriented (OO) technology". During the simulation, an engine's mathematical model is no longer decomposed into segments of function call that implement certain algorithms. Rather, it is mapped to collections of communicating software objects, each of which mimics the behavioral and structural characteristics of a physical or conceptual entity. Each object must represents an instance of a software class, while the classes are united into a hierarchy via inheritance relationships.

The OO programming accommodates a convenient, safe, and seamless paradigm of software extension. For example, there may already exist in the system, a software object "compressorOfLM2500", which is an instance of the class "Compressor". The class is used to generalize and abstract different kinds of compressors. It may inherit from a more generalized class "Component". During the software extension, if a new compressor object, such as a compressor of the LM6000 engine, needs to be modeled, it can be easily instantiated from the existing class "Compressor". The embedded OO mechanism ensures that no incompatibility may be introduced even though multiple objects reuse the same codes written for the class "Compressor". If a novel component object needs to be extended, a new subclass that inherits from the class "Component" must be created before the new object can be instantiated. In this manner, a well-planned class hierarchy provides the "slots" into which the future codes are to be plugged.

The existing object models [2–7] focus mainly on extending new engine components and plant configurations. The gas path balancing method, however, remains inflexible.

Gas path balancing refers to the intermediate simulation phase during which all components along the engine gas path are manipulated such that certain (but not all) balance conditions are satisfied. These balances must be established before any other equilibrium can be set up and the system's state calculated. For example, during the off-design performance simulation using Newton–Raphson algorithm [8], gas path balancing requires the equilibrium of both the mass flow rate and the pressure ratio among all components. On the other hand, work extracted from a turbine does not need to be equal to that consumed by the corresponding compressor and/or load. This work imbalance will speed up or slow down the specified rotor speed, which in turn forms the initial condition for the next iteration of gas path balancing.

In addition to the conventional sequential method of balancing, various methods [9,10] have already been proposed with improved simulation accuracy, speed, and/or stability. Since new cases and simulation requirements emerge frequently, the adoption of other methods and hence, the rewriting of those packages with limited gas path balancing choices, would be inevitable.

To avoid the latter situation, an improved object model must be developed. This paper proposes a new OO design rationale for gas turbine simulation, describes the resulting class structure that improves extendability, and presents an example that demonstrates the model's effectiveness.

## 2. Message-based modeling

Despite of the model differences, almost all the existing object models for gas turbine engine simulation adopt a design rationale called "component-based modeling" [4]. The software designers wish to provide the users with the capability to assemble an engine the same way as that in thermodynamics, which is, by coupling the physical components together using thermodynamic linkages, such as the working fluid flows and the shafts. However, the internal structure and behavior of a real world thermodynamic linkage is very difficult to emulate by a single software object.

Take the working fluid flow as an example. During the gas path balancing, it acts as a means of data transmission between two components. As depicted in Fig. 1, if component A and B are coupled together by a working fluid flow, the value of any thermodynamic parameter at A's outlet should be equal to that at B's inlet. If the value on one side is known and that on the other side is unknown, the fluid flow must assign the known value to the other side. The transmitted data are of various types, ranging from temperature, pressure, to enthalpy, mass flow rate, and/or chemical composition. Some unexpected types of data may also need to be involved when the components are of special type. Besides, although the working fluid flows are unidirectional from upstream to downstream, the data transmissions do not have to behave in the same way. Their directions differ from one parameter type to another, and from one balancing method to another, as shown in Fig. 1. The previous models tried to wrap all these complexities in one single object, whose mechanism of data transmission turned out to be too complicated for new parameter types to be inserted in and hence new component types and balancing methods extended and customized.

Realizing this, the authors try to split the structural and functional characteristics of a thermodynamic linkage into multiple objects. During the process, a novel design rationale, called "message-based modeling", is formed and introduced herein. The term "message" is defined as an entity that abstracts the behavior of an independent data transmission. Although there are many data transactions in the simulation, they do not function in the same way. Some of the transactions imply more than the data transferred. Take Fig. 1 as an example. If the temperature value at A's outlet is assigned to B's inlet, the enthalpy value at B's inlet can be calculated immediately. This implies that the enthalpy value has already been transmitted along with the temperature at the same time and in the same direction. These two pieces of datum are interchangeable and the transmission of either one will result in the other being transferred in the same direction. Therefore, they are grouped as one piece of message. On the

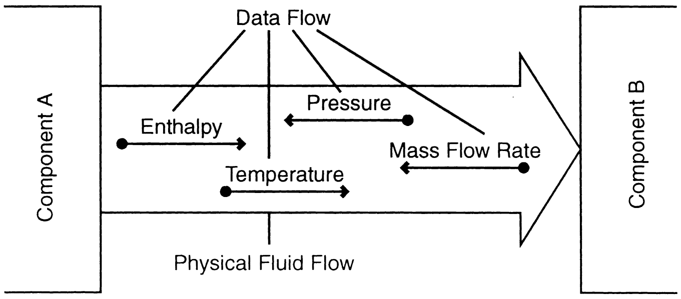

Fig. 1. Physical fluid flow and data flow.

other hand, the transmissions of pressure, mass flow rate, and temperature are independent from each other. Each of them is deemed as one piece of message.

All data in the simulation are grouped into independent messages. Rather than connecting the components with certain thermodynamic linkages, the design of an engine simulation is converted to sending messages among autonomous component objects. The users will send out initial messages to trigger a few components. These components will invoke the corresponding simulation procedures, calculate the unknown data, and then send out messages to trigger more components. Different gas path balancing methods are thus implemented through the different layouts of messages that can be customized and extended at ease.

An extensible object model for gas turbine simulation is developed based on this idea and the hierarchical class structure is described in the next section. At the core of the hierarchy, there are three classes: Node, Connector, and Component.

## 3. Node-Connector-Component model

A node denotes the static state of a message. Before any message is passed out, it is only an aggregation of dependant data contained by a component. There are many kinds of Node in a simulation, e.g., temperature, pressure, rotary speed, etc. To guarantee extensibility to other kind of node, an abstract base class "Node" is created to generalize all nodes. This relation is depicted in Fig. 2. The notations used in Figs. 2–4 conform to Unified Modeling Language (UML) [11], the OO modeling standard from Object Management Group (OMG). As shown in Fig. 2, an arrow with a triangle arrowhead stands for the inheritance, or rather, the "Is-A" relationship. The subclasses of Node are (from left to right) rotary speed's derivation over time, specific work, mass flow rate, fuel–air ratio, temperature of arbitrary fluid, pressure, pressure's derivation over time, and rotary speed. The class NewNode stands for any new subclass of Node, which must be extended to the hierarchy at the level shown in Fig. 2. The subclasses of TFluidNode are temperature, along with other attached data such as enthalpy, specific heat, etc., of pure air and pure gas' correlation over pure air, respectively.

The relationships between the classes Node and Component are shown in Fig. 3, using the class Compressor as an example. In UML, a diamond decorated line or arrow stands for "Has-A" or "Is-a-Part-Of" relationship. With all necessary Node class extended, a novel Component class can be created by applying these relationships in a proper way.

Messages must be passed among components. Therefore, a mechanism that implements this communication must be created. To avoid coding for each kind of Node, a Connector class is created over their root class, Node, as shown in Fig. 2. Modifications of class Connector for future insertions of new parameter types are therefore unnecessary.

Component classes are still needed. However, their assembly may be heterogeneous to a real-world engine. For example, a compressor with air leakage may be modeled by connecting a Compressor object's mass flow rate outlet node with of a Manifold object's inlet node. New component classes must be appended to the hierarchy at the proper level, as shown in Fig. 4.

As a whole, the extensibility of the model is depicted in Fig. 4. By extending new Node subclasses, new components and new gas path balancing methods can be expanded and supplemented to the system at ease. New system state calculation methods, which are

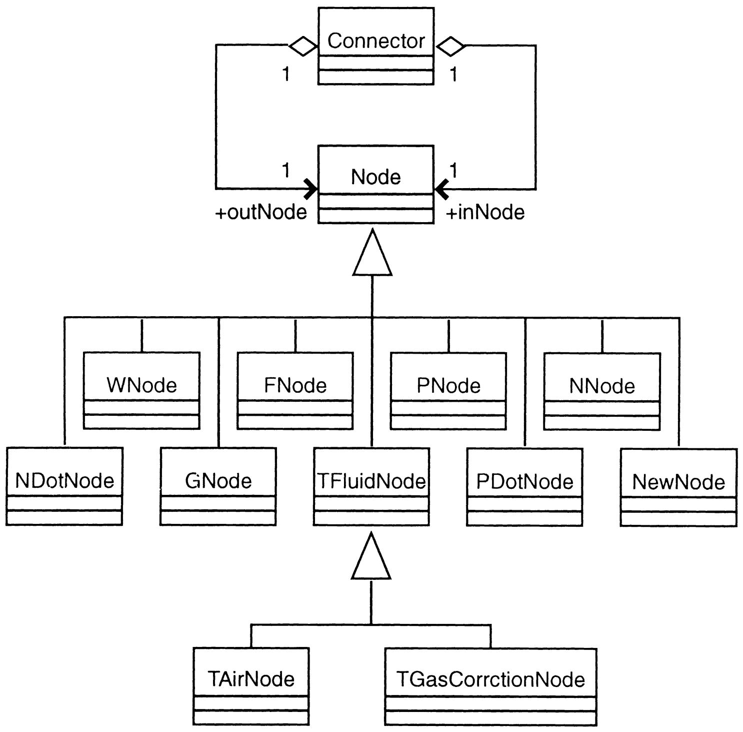

Fig. 2. Node and Connector class diagram.

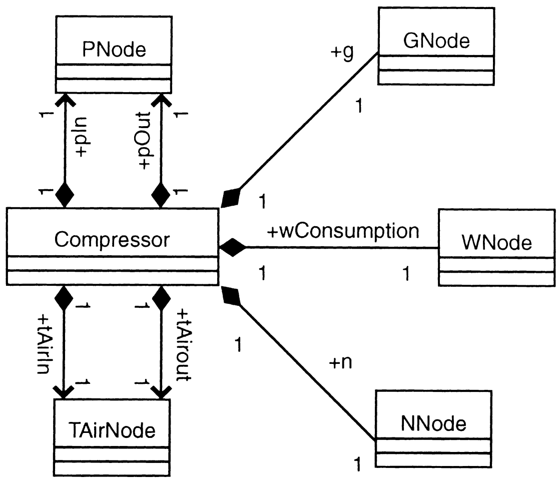

Fig. 3. Compressor and node class diagram.

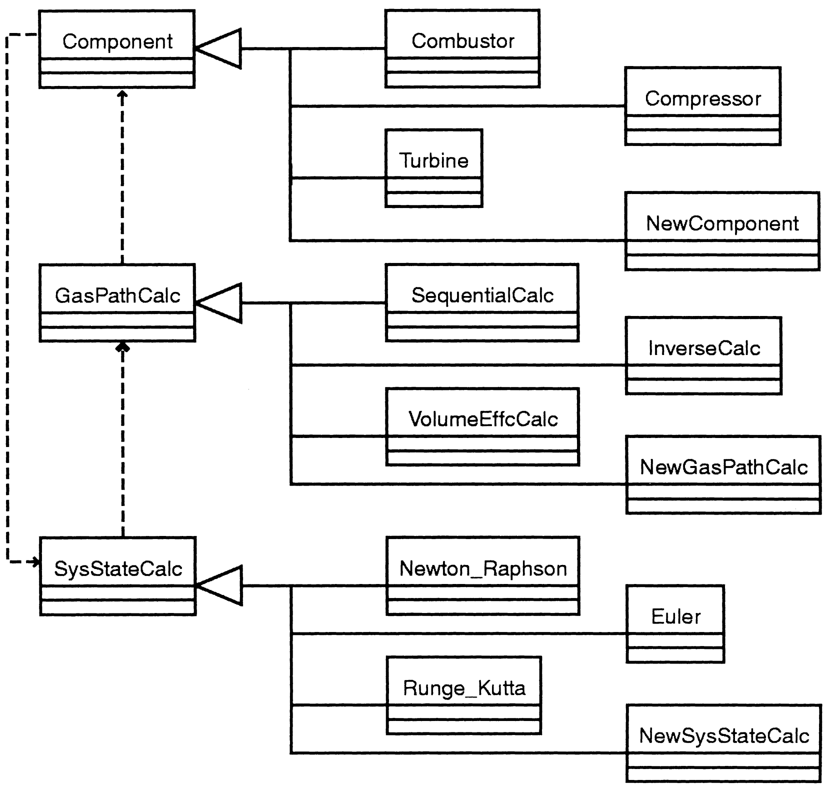

Fig. 4. Class structure of the extensible object model.

practically numerical methods that solve the ordinary differential equations (ODE), may also be appended.

## 4. Example

To demonstrate the effectiveness of the model, a three-shaft gas turbine engine's dynamic

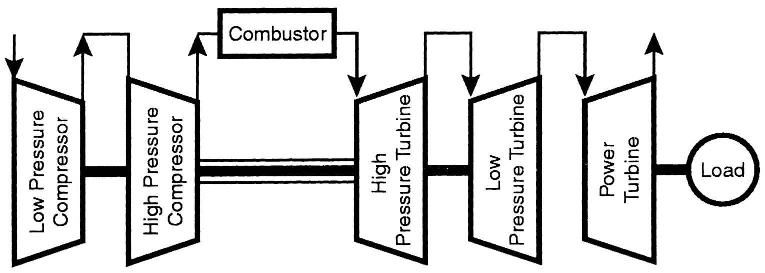

Fig. 5. Layout of a three-shaft gas turbine engine.

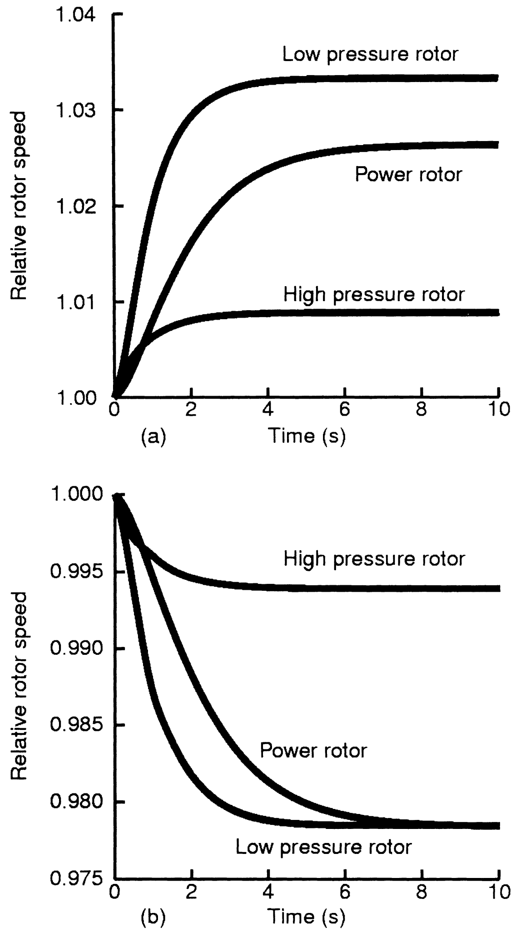

Fig. 6. The engine's dynamic response to a small fuel step perturbation: (a) upward perturbation, (b) downward perturbation.

performance is simulated using the volume inertia balancing method [10]. The layout configuration of the plant is shown in Fig. 5. The air/gas leakage and pressure losses along the gas path have all been modeled.

In this example, altogether 22 components and 56 connectors are created and assembled. The simulation time step is set to 5 ms, and altogether 2000 steps are calculated. On a Pentium II 233 MHz personal computer, this calculation task is completed in 5 s, which illustrates the vast power of the volume inertia method. The engine's responses to small fuel flow step perturbations (both upward and downward) are depicted in Fig. 6, in which the relative rotor speed is defined as the division of a rotor's speed at time *t* over that at time 0.

As stated before, an object model is only a programming re-mapping of the engine's mathematical model. The validation of the mathematical model and algorithms are beyond the scope of this paper. However, the results of this calculation are exactly the same as a previous non-OO project [10] on the same problem, which presents the validity of this object model.

## 5. Conclusions

An extensible model for general-purpose gas turbine engine simulation has been built. New components and new engine configurations can be appended to the system. Extensibility on gas path balancing methods is also provided. This attributes to the employment of the message-based modeling, which overcomes the deficiencies of traditional component-based modeling by the introduction of concept "message". An implementation object model with the core concepts of Node-Connector-Component is also described, and a three-shaft gas turbine engine's dynamic performance over fuel step perturbation is simulated. The effectiveness of the model has been proved.

## Acknowledgements

This work was supported by the Grant No. 59976022 from Natural Science Foundation of China (NSFC).

## References

[1] IEEE, IEEE standard computer dictionary: a compilation of IEEE standard computer glossaries, New York, NY, 1990.
[2] G. Heyen, K. Murphy, D. Marchio, P. Kalata, B. Kalitventzeff, E. Marechal, Dynamic simulation and control of gas turbines and compressor systems, Computers in Chemical Engineering 18 (1994) 1071–1082.
[3] L. Evans, J. Lytle, G Follen. I. Lopez, An Integrated Computing and Interdisciplinary Systems Approach to Aeropropulsion Simulation, ASME Paper 97-GT-303 American Society of Mechanical Engineers, New York, 1997.
[4] P. Curlett, J.L. Felder, Object-Oriented Approach for Gas Turbine Engine Simulation, NASA-TM-106970. Lewis Research Center, National Aeronautics and Space Administration, Cleveland, OH, 1995.
[5] K. Drummond, G.J. Follen. Y. Cannon, Object-Oriented Technology for Compressor Simulation, NASA-TM-10672, Lewis Research Center, National Aeronautics and Space Administration, Cleveland, OH, 1992.
[6] K. Drummond, G.J. Follen, C.W. Putt, Gas Turbine System Simulation: An Object-Oriented Approach, NASA-TM-106044, Lewis Research Center, National Aeronautics and Space Administration, Cleveland, OH, 1992.
[7] P.Y Ho, P.S. Ng, Object-Oriented Approach to Gas Turbine Performance Computation, ASME Paper 96-GT-165, American Society of Mechanical Engineers, New York, NY, 1996.
[8] S. Weng, Gas Turbine Engine Performance Analysis, Shanghai Jiao Tong University Press, Shanghai, 1987 (in Chinese).
[9] Y. Wang, A new method of predicting the performance of gas turbine engines, ASME Journal of Gas Turbines and Power 113 (1991) 106–111.
[10] M. Su, S. Weng, A volume component model and its solution for power system simulation, Power Engineering 18 (1998) 75–78, 82 (in Chinese).
[11] Object Management Group, UML summary, [online] Available online from http://www.omg.org/uml, 1998.