NASA/TM—2009-215656 GT2008–50062

An Object-Oriented Computer Code for Aircraft Engine Weight Estimation

Michael T. Tong
Glenn Research Center, Cleveland, Ohio

Bret A. Naylor
DB Consulting Group, Cleveland, Ohio

December 2009

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to *help@sti.nasa.gov*

- Fax your question to the NASA STI Help Desk at 443–757–5803

- Telephone the NASA STI Help Desk at 443–757–5802

- Write to:
  NASA Center for AeroSpace Information (CASI)
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/TM—2009-215656 GT2008–50062

An Object-Oriented Computer Code for Aircraft Engine Weight Estimation

Michael T. Tong
Glenn Research Center, Cleveland, Ohio

Bret A. Naylor
DB Consulting Group, Cleveland, Ohio

Prepared for the
Gas Turbine Technical Congress and Exposition (Turbo Expo 2008)
sponsored by the American Society of Mechanical Engineers
Berlin, Germany, June 9–13, 2008

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

December 2009

This work was sponsored by the Fundamental Aeronautics Program
at the NASA Glenn Research Center.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Available electronically at http://gltrs.grc.nasa.gov

An Object-Oriented Computer Code for Aircraft Engine Weight Estimation

Michael T. Tong
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Bret A. Naylor
DB Consulting Group
Cleveland, Ohio 44135

Abstract

Reliable engine-weight estimation at the conceptual design stage is critical to the development of new aircraft engines. It helps to identify the best engine concept amongst several candidates. At NASA Glenn Research Center (GRC), the Weight Analysis of Turbine Engines (WATE) computer code, originally developed by Boeing Aircraft, has been used to estimate the engine weight of various conceptual engine designs. The code, written in FORTRAN, was originally developed for NASA in 1979. Since then, substantial improvements have been made to the code to improve the weight calculations for most of the engine components. Most recently, to improve the maintainability and extensibility of WATE, the FORTRAN code has been converted into an object-oriented version. The conversion was done within the NASA's NPSS (Numerical Propulsion System Simulation) framework. This enables WATE to interact seamlessly with the thermodynamic cycle model which provides component flow data such as airflows, temperatures, and pressures, etc. that are required for sizing the components and weight calculations. The tighter integration between the NPSS and WATE would greatly enhance system-level analysis and optimization capabilities. It also would facilitate the enhancement of the WATE code for next-generation aircraft and space propulsion systems. In this paper, the architecture of the object-oriented WATE code (or WATE++) is described. Both the FORTRAN and object-oriented versions of the code are employed to compute the dimensions and weight of a 300-passenger aircraft engine (GE90 class). Both versions of the code produce essentially identical results as should be the case.

Introduction

Engine weight is a key design parameter for any new aircraft. It affects aircraft range and is a key element in fuel burn. Weight is also considered an indicator of engine cost. Reliable engine-weight estimation at the conceptual design stage is critical to the development of new aircraft engines. It helps to identify the best engine concept amongst several candidates.

At the NASA Glenn Research Center (GRC), the Weight Analysis of Turbine Engines (WATE) computer code (Ref. 1), originally developed by Boeing Aircraft, has been used to estimate the engine weight of various conceptual engine designs. The code, written in FORTRAN, was originally developed for NASA in 1979. It calculated the weight and dimension of each major engine component, such as compressor, burner, turbines and frames, primarily using a semi-empirical method augmented with analytical calculations for specific component elements. A database of 29 engines was used to develop empirical relationships used to calculate component weight and dimensions. This method provided an accuracy of approximately ±10 percent of engine weight for the database engine designs.

Since 1979, substantial improvements have been made to the computer code by NASA (Ref. 2), and McDonnell Douglas Corporation, to enhance the capability of WATE and to improve its accuracy. Many of the empirical relationships have been replaced with analytical weight and dimension calculations. The primary method used to calculate weight throughout the code is to calculate material volume and multiply by material density. An approach is used where the stress level, maximum temperature and pressure, material, geometry, stage loading, hub-tip ratio, blade/vane counts, and shaft speed are used to determine the component weight. Flow properties such as corrected mass flow, temperatures, pressures, etc. on each component are obtained from the thermodynamic cycle analysis. This method accounts for more of the individual parts that make up an engine component than an empirical method. A material database, consisting of the material data of most of the commonly-used aerospace materials, was also incorporated into WATE. In addition to engine component weight calculation improvements, a large number of changes were made to the WATE code to improve the graphical output of the program. Component graphic representation has been greatly enhanced to provide a more detailed picture of the flowpath to assist the user in determining the correctness of the flowpath. The code was also modified to allow the user to use controls and limiters when developing the flowpath. This capability saves the user time in assuring the adjacent components will connect correctly. An optimizing capability was added as well to allow optimization of the flowpath on either weight or dimensions.

Recently, to improve the maintainability and extensibility of WATE, GRC analysts converted the FORTRAN code into an object-oriented version. The conversion was done within the NASA's NPSS (Numerical Propulsion System Simulation) (Ref. 3) framework. This enables WATE to interact seamlessly with the thermodynamic cycle model which provides component flow data such as corrected airflows, temperatures, and pressures, etc., that are required for sizing the components and weight calculations.

The object-oriented programming is already an established software development method and its advantages over procedure-oriented programming like FORTRAN have been widely recognized (Refs. 4 and 5). The main ideas behind object-orientation are data abstraction, inheritance, polymorphism and dynamic binding. This paper describes the approach to the conversion of the FORTRAN engine-weight estimation code into an object-oriented code using NPSS interpreted language. The engine-weight modeling system and its object-oriented architecture are described. Validation of the code is presented.

WATE++ Architecture

The object-oriented WATE (or WATE++) calculates the weight and dimension of each major engine component, relative to a reference point, usually the design point. The WATE++ architecture is intended to be flexible and extensible. It exploits the capabilities of object-oriented programming (inheritance, polymorphism, and encapsulation), as well as modern object-oriented concepts including framework and component objects. Inheritance is used to concentrate code common to multiple component types in abstract component classes, preventing code duplication and enhancing code maintainability. For example, the 'Axial Compressor class' in Figure 1 represents an abstract ancestor incorporating all functionality common to fan, low-pressure compressor, and high-pressure compressor. Encapsulation enhances code maintainability and readability by concentrating all data declarations and procedures in a single code unit. Polymorphism is the ability of parameters to represent different object classes and is extensively applied in WATE++. It allows the framework to get the correct behavior of each WATE++ element without knowing what the specific type of each one is. For example, the WATE++ has an abstract identifier 'calcGeometry' to calculate the geometry of any component in the model. During simulation, 'calcGeometry' subsequently represents all WATE++ components and runs their geometry calculations so that the engine flowpath can be drawn.

There are three primary object types that are used in the WATE++ calculations: elements, ports, and subelements.

Elements

WATE++ elements perform high level component calculations. Elements may have many plug locations termed sockets, into which computational blocks termed subelements may be attached. The compressor element, for example, contains a socket into which the subelement that does the disk-sizing calculation is attached. It also has sockets into which the user may plug subelements for computing the gearbox and frame weights.

Ports

Ports provide data connectivity between elements. In WATE++, ports are used to transfer geometry information (such as radius and axial position) between elements. It also provides mechanical links from one element to another. For example, a port is used to connect the shaft with the compressor and turbine.

Subelements

Subelements perform specific, detailed computations. They generally only work when connected to sockets. WATE++ supports multiple types of subelements that can plug into the element sockets. Each subelement performs detailed computations that impact the element's overall calculations. The variable-area-nozzle subelement, for example, can be plugged into the nozzle element socket for variable-area nozzle computation. Sockets on an element need not be filled; default values will be used if left empty.

Element and Subelement Classes

The abstract class 'WATEelement' encapsulates the common structural components of a gas turbine engine. These component classes and their inheritance hierarchy are shown in Figure 1. In the entire code, the interaction between different classes is only through messages.

Interaction With the Thermodynamic Cycle Model

WATE++ is designed to function with the thermodynamic cycle model within the NPSS framework, as shown in Figure 2. The thermo design point case of the thermodynamic cycle model can be used to provide engine cycle data required for sizing the engine components, or additional off-design points can be run and the output data will be scanned for maximum conditions of airflows, pressures, or temperatures for each component. In order to produce the most accurate weight estimate, the off-design cases should encompass the maximum performance level required for each engine component. All components that contribute weight must also be included in the thermodynamic cycle model.

The development of NPSS was a cooperative effort between NASA and other government agencies, industry, and universities to integrate propulsion technologies with high-performance computing and communication technologies into a complete system for performing detailed full-engine simulations (Fig. 6). It consists of three main elements: (1) the engineering application models, (2) the system software for the simulation environment, and (3) the high-performance computing platform. To facilitate the timely and cost-effective capture of complex physical processes, NPSS uses object-oriented technologies such as C++ objects to encapsulate individual engine components and Object Request Brokers (ORB's) from the Common Object Request Broker Architecture for object communication and deployment across heterogeneous computing platforms.

The ultimate goal of NPSS is to create a "numerical test cell" that enables engineers to examine various design options numerically and minimize the number of costly and time-consuming real life tests.

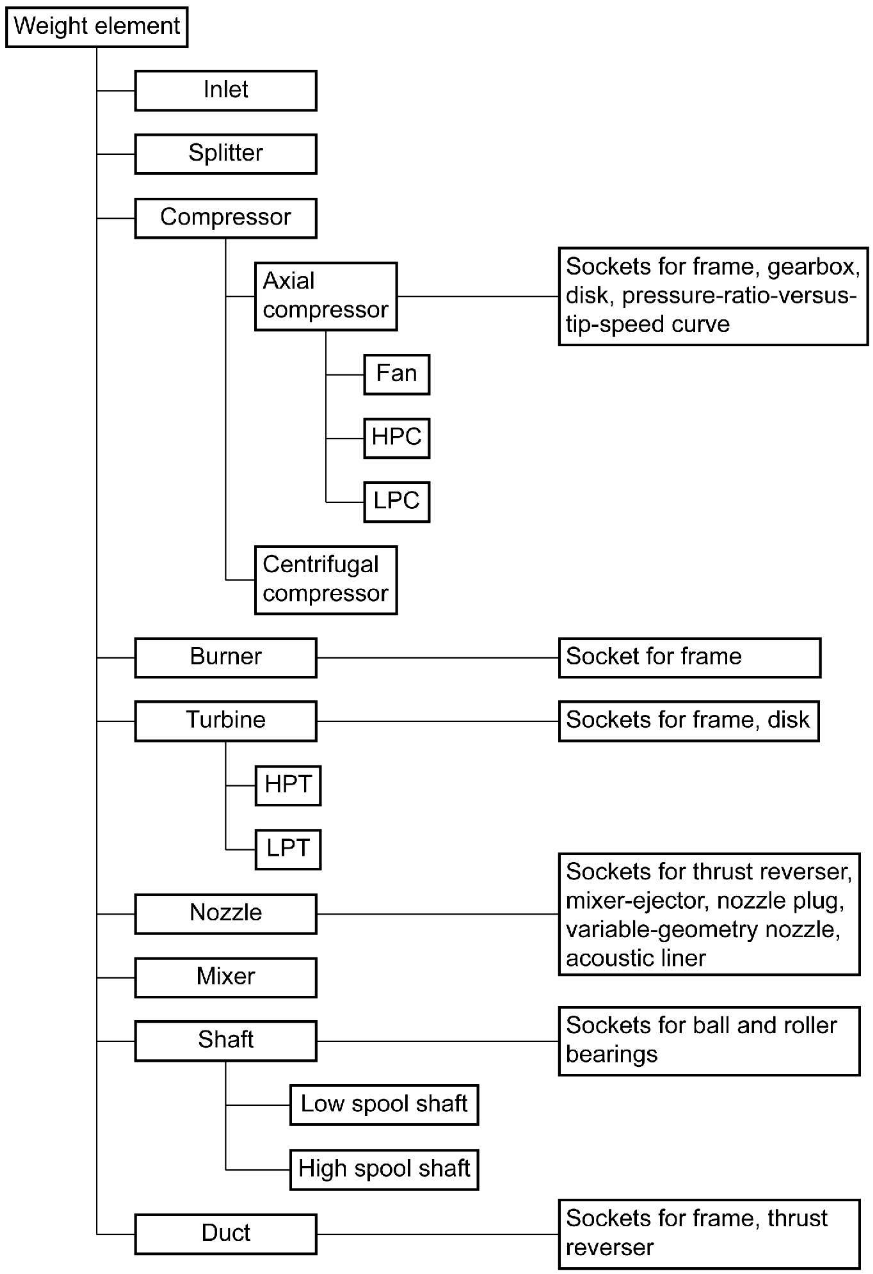

Figure 1.—WATE++ component inheritance architecture.

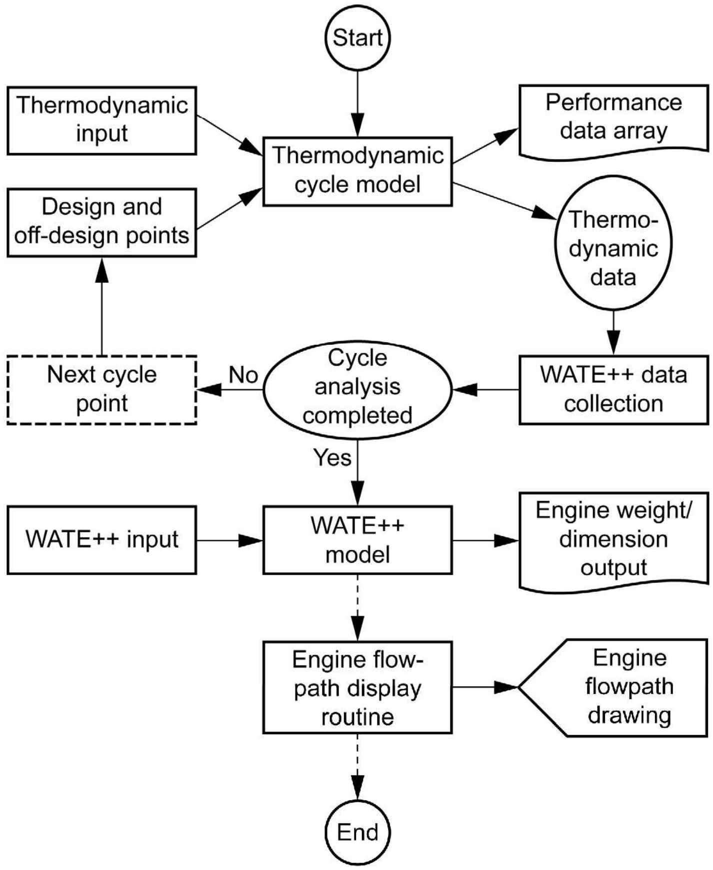

Figure 2.—NPSS framework.

Material Database

The material database in WATE++ consists of material data of most of the commonly-used aerospace materials. The list of materials is shown in Table 1. The object *WATEmaterial* allows the user to enter the material name to specify which material property table to use. Material properties will be automatically set using the material property table unless the user chooses to override them. The user can override a given property by defining a function or table with the same material name within the given *WATEmaterial* object.

WATE++ Input and Output

The basic format of a WATE++ input file is text-based. The input procedures can be broken down into three main steps: (1) creation of objects, (2) assignment of values to variables, and (3) commands. The input file is read sequentially so normally an object will be created then values assigned to its variables and the process repeated with the next object. Once all the objects have been declared then commands are made to the code. An example of WATE++ input for an engine fan is shown in Appendix A.

TABLE 1.—LIST OF MATERIALS IN WATE++ MATERIAL DATABASE

Ti-6Al-4V	MAR-M509	Inconel-706	Udimet-710
Ti-17	WI-52	Inconel-718	Waspaloy
Ti-6-2-4-2	IN-100	TD Nickel	Rene-41
Alloy 713C	Hastelloy-X	Haynes-188	Rene-80
Alloy 713LC	Hastelloy-S	L-605	Rene-95
Alloy-901	Inconel-600	A-286	410 steel
B-1900	Inconel-601	N-155	4130 steel
IN-100	Inconel-617	V-57	4340 steel
MAR-M247	Inconel-625	Udimet-500	17-4PH steel
MAR-M302	Inconel-690	Udimet-700	

A *data-viewer* object, which uniquely determines which variables to output as well as its format, has been created to generate WATE++ output file. An example of WATE++ output is shown in Appendix B.

WATE++ Execution

WATE++ components are contained within *WATEassembly*. Execution of the WATE++ components happens in two phases. The first phase is the “COLLECT” phase, where the WATE++ components query the thermodynamic cycle model as it runs to determine maximum values of temperatures and pressures and other parameters that WATE++ will later use to calculate weights and lengths. At some point, the WATE++ components will be placed in the “CALCULATE” state. The *WATEassembly* can then be executed directly, by calling the *run()* function on it, and this will cause all of the WATE++ components to perform their calculations. An engine flowpath plot can be obtained by instantiating a *WATEsvgViewer* object inside the *WATEassembly*.

Validation of WATE++ Code

The validation of WATE++ has been done by using it to perform an engine sizing and weight calculation for a 300-passenger aircraft engine (GE90 lass). The results are compared with those calculated by the FORTRAN WATE in Table 2. Both versions of the code produce essentially identical results. The flowpath of the engine is shown in Figure 3.

TABLE 2.—A COMPARISON OF ENGINE WEIGHT AND DIMENSIONS OUTPUT BETWEEN WATE++ AND WATE (FORTRAN)

	WATE++	WATE	Percent difference
Bare engine, wt./lb	16430	16428	0.01
Total engine, wt./lb	18156	18154	0.01
Inlet/nacelle, wt./lb	1891	1892	0.05
Engine pod, wt./lb	20047	20046	0.04
Length, in.	202.8	203.0	0.10
Pod length, in.	263.1	263.3	0.08
Fan cowl length, in.	207.6	207.6	-----
Engine maximum diameter, in.	122.7	122.7	-----
Nacelle maximum diameter, in.	150.8	150.9	0.07
Engine pod c.g. location, in.	79.4	78.6	0.25

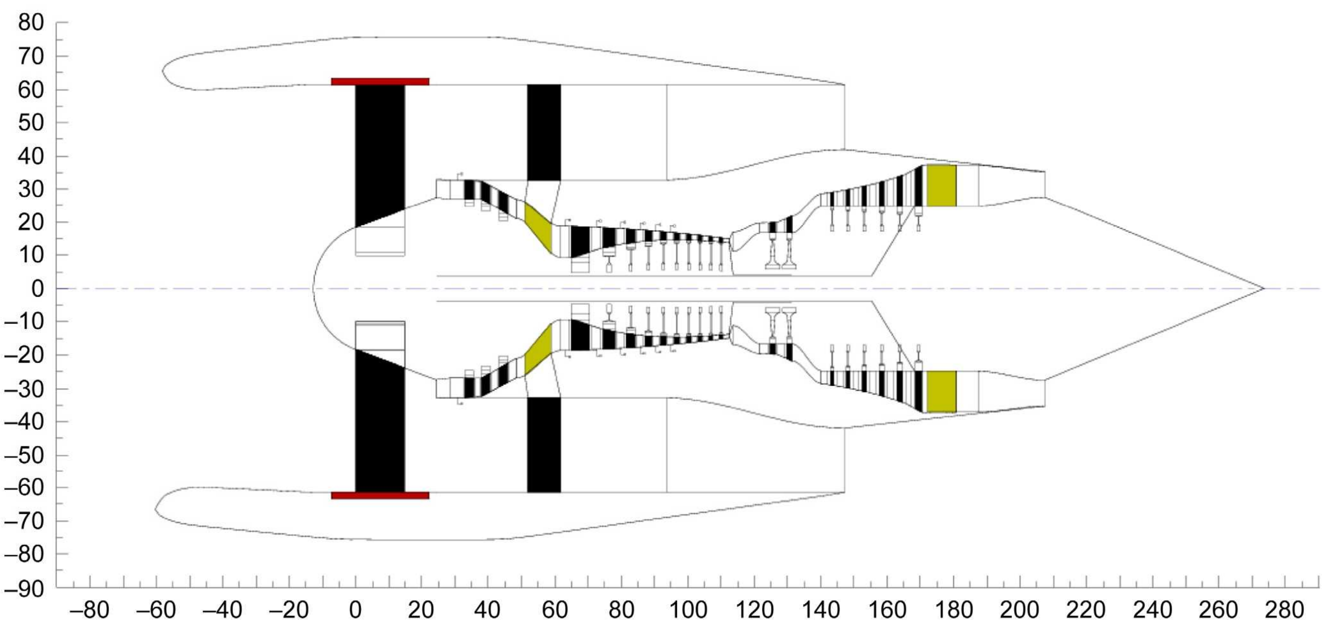

Figure 3.—300-passenger engine flowpath model generated by WATE++.

Summary

In this paper, the architecture of WATE++, an object-oriented engine-weight estimation code, has been described. The code was converted from the structured FORTRAN version of the code, to improve its maintainability and extensibility. It was accomplished by carefully designing the object classes and choosing the exact data type to suit the application. The code has been validated by comparing the results that it calculated with those calculated by the FORTRAN version. Both versions of the code give essentially identical results, as should be the case.

References

1. Onat, E. and Klees, G.W., "A Method to Estimate Weight and Dimensions of Large and Small Gas Turbine Engines," NASA CR–159481, 1979.
2. A Computer Code for Gas Turbine Engine Weight and Life Estimation. Michael T. Tong, Ian Halliwell, Louis Ghosn, ASME Journal of Engineering for Gas Turbine and Power, volume 126, no. 2, April 2004.
3. NASA-Industry Cooperative Effort: "Numerical Propulsion System Simulation User Guide and Reference," Software Release NPSS 1.5.0, May 7, 2002.
4. Booch. G., "Object-Oriented Design with Applications," The Benjamin/Cummings Publishing Company, Inc., Redwood City, California, 1991.
5. Lorenz, M. and Kidd, J., "Object-Oriented Software Metrics," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1994.
6. Lytle, J.K., "The Numerical Propulsion System Simulation: An Overview," NASA TM—2000-209915, 2000.

Appendix A

Wate++ Input File For a Gas Turbine Engine Fan—An Example

```
//=================================================
//                       FAN
//=================================================
WATEhiBypassFan WATE_Fan {
   componentRef = "Fan";
   numContainedStages = 1;
   MNin = 0.630;
   MNout= 0.460;
   stg1MaxPR = 1.7;
   hubTipRatioIn = 0.30;
   numBlades_in = 22;
   calcStatorWt = FALSE;
   bladeMaterial.type = "Ti_17";
   statorMaterial.type = "Ti_6Al_4v";
   real bladeMaterial.rho(real) { return 0.096; }
   real statorMaterial.rho(real) { return 0.1; }
   real containmentMaterial.rho(real) { return 0.051; }
   contRingRadialThickness = 2;
   bladeSolidity = 1.5;
   bladeVolumeFactor = 0.024;
   stg1BladeAR = 2.045;
   lastStgBladeAR = 2.045;
   bladeTaperRatio = 1.40;
   maxSpdRatio_in = 1.000;
   s_Nmech = 1.;
   geometryType = "ConstTipRadius";
   radiusChangeStg = 1;
   numStatorBlades_in = 54;
   statorVolumeFactor = 0.14;
   stg1StatorAR = 3.754;
   lastStgStatorAR = 3.754;
   RSlenRatioPreSplit = 0.65;
   RSlenRatioPostSplit = 1.85;
   RSlenRatioBypass = 0.78;
   real caseMaterial.rho(real) { return 0.1; }
   real igvMaterial.rho(real) { return 0.1; }
   ductLenInnerRR     = 0.0;
   statorSolidity       = 0.0;
   stg1StatorRotorLR    = 1.0;
   lastStgStatorRotorLR = 1.0;

   Table TB_PRvsTipSpd(real pratio) {
      pratio = {1., 1.18, 1.36, 1.43, 1.503, 1.581, 1.667,
                1.775, 1.9}
      utip   = {600., 885.,1100.,1200., 1300., 1400.,
                1500., 1600.,1700.}
   }
   WATEdiskMTC S_Disk {
      shape1 = "OPTIMUM";
      material.type = "Ti_17";
      shaftRef = "WATE_LP_Shaft";
   }
   WATEframeCustom S_RearFrame {
      isFrontFrame = FALSE;
      isStator = TRUE;
      real material.rho(real) { return 0.16; }
      real supportMaterial.rho(real) { return 0.1; }
      volumeFactor = 0.05;
      aspectRatio_in = 2.90;
      supportThickness = 0.1;
      gapFrameLengthRatio = 0.35;
      numBlades_in =  54;
      passThruComp = "WATE_Duct6";
      connectPoint = "REAR";
      rearBearingRef = "WATE_LP_Shaft.bearing1";

      WATEtowerShaft S_TowerShaft {
         HPX = 500;
         diamRatio = 0.90;
      }
   }
}
```

Appendix B

Wate++ Engine Weight Output Summary—An Example

	WT	COMP LEN	ACCU LEN	UPSTR RI1	UPSTR RO1	UPSTR RI2	UPSTR RO2	DWNSTR RI1	DWNSTR RO1	DWNSTR RI2	DWNSTR RO2
WATEa.WATE_Fan	4026	24.498	24.498	18.403	61.343	0.000	0.000	27.554	61.343	27.554	61.343
WATEa.WATE_Splitter	0	0.000	24.498	27.554	61.343	27.554	61.343	27.554	32.791	32.791	61.343
WATEa.WATE_Duct4	12	4.000	28.498	27.351	32.993	27.554	32.791	27.351	32.993	25.566	31.563
WATEa.WATE_LPC	668	19.855	48.354	25.566	31.563	27.351	32.993	19.941	26.038	19.941	26.038
WATEa.WATE_Duct6	327	13.200	61.554	19.941	26.038	19.941	26.038	19.941	26.038	9.311	18.622
WATEa.WATE_HPC	2022	50.900	112.454	9.311	18.622	19.941	26.038	13.763	14.898	13.802	14.859
WATEa.WATE_Bld3	10	1.200	113.654	13.802	14.859	13.763	14.898	13.802	14.859	11.104	16.954
WATEa.WATE_Burner	623	8.100	121.754	11.104	16.954	13.802	14.859	11.104	16.954	16.757	19.692
WATEa.WATE_HPT	1744	9.503	131.256	16.757	19.692	11.104	16.954	16.757	21.760	16.703	21.814
WATEa.WATE_Duct11	31	9.000	140.256	16.703	21.814	16.757	21.760	16.703	21.814	24.748	28.528
WATEa.WATE_LPT	3544	40.322	180.578	24.748	28.528	16.703	21.814	24.748	36.920	24.748	36.920
WATEa.WATE_Duct13	37	6.700	187.278	24.748	36.920	24.748	36.920	24.748	36.920	24.748	36.920
WATEa.WATE_Core_Nozz	323	15.506	202.785	24.748	36.920	24.748	36.920	27.551	35.074	0.000	0.000
WATEa.WATE_Duct15	94	32.043	93.854	32.791	61.343	32.791	61.343	32.791	61.343	32.791	61.343
WATEa.WATE_Byp_Nozz	1665	53.368	147.222	32.791	61.343	32.791	61.343	41.857	61.343	0.000	0.000
WATEa.WATE_Inlet	1564	60.342	0.000	0.000	0.000	0.000	0.000	0.000	0.000	18.403	61.343
WATEa.WATE_LP_Shaft	1126	146.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
WATEa.WATE_HP_Shaft	168	18.803	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

```
ENGINE MOUNT WEIGHT      = 326.46
BARE ENGINE WEIGHT       = 16429.8
ACCESSORIES WEIGHT       = 1726
TOTAL ENGINE WEIGHT      = 18155.8
INLET/NACELLE WEIGHT     = 1891.01
TOTAL ENGINE POD WEIGHT  = 20046.8

ENGINE LENGTH            = 202.785
TOTAL ENGINE POD LENGTH  = 263.127
FAN COWL LENGTH          = 207.565
ENGINE MAX DIAMETER      = 122.685
NACELLE MAX DIAMETER     = 150.825
ENGINE POD C.G. LOCATION = 79.4397
```

	TOT WT	ROTOR	STATOR	FRAME	LT	LTM
WATEa.WATE_Fan	4026.61	1252.351	1486.884	1287.379	61.812	51.966
WATEa.WATE_LPC	668.79	363.867	304.922	0.000	19.855	19.855
WATEa.WATE_HPC	2022.57	1050.818	971.751	0.000	50.900	50.900
WATEa.WATE_HPT	1744.70	966.672	446.962	331.070	9.503	9.503
WATEa.WATE_LPT	3544.30	2061.493	939.405	543.404	40.322	30.249

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
01-12-2009	Technical Memorandum	

4. TITLE AND SUBTITLE
An Object-Oriented Computer Code for Aircraft Engine Weight Estimation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Tong, Michael, T.; Naylor, Bret, A.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
WBS 561581.02.08.03.13.03

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E-16428-1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITOR'S ACRONYM(S)
NASA

11. SPONSORING/MONITORING REPORT NUMBER
NASA/TM-2009-215656; GT2008-50062

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category: 07
Available electronically at http://gltrs.grc.nasa.gov
This publication is available from the NASA Center for AeroSpace Information, 443-757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Reliable engine-weight estimation at the conceptual design stage is critical to the development of new aircraft engines. It helps to identify the best engine concept amongst several candidates. At NASA Glenn Research Center (GRC), the Weight Analysis of Turbine Engines (WATE) computer code, originally developed by Boeing Aircraft, has been used to estimate the engine weight of various conceptual engine designs. The code, written in FORTRAN, was originally developed for NASA in 1979. Since then, substantial improvements have been made to the code to improve the weight calculations for most of the engine components. Most recently, to improve the maintainability and extensibility of WATE, the FORTRAN code has been converted into an object-oriented version. The conversion was done within the NASA's NPSS (Numerical Propulsion System Simulation) framework. This enables WATE to interact seamlessly with the thermodynamic cycle model which provides component flow data such as airflows, temperatures, and pressures, etc., that are required for sizing the components and weight calculations. The tighter integration between the NPSS and WATE would greatly enhance system-level analysis and optimization capabilities. It also would facilitate the enhancement of the WATE code for next-generation aircraft and space propulsion systems. In this paper, the architecture of the object-oriented WATE code (or WATE++) is described. Both the FORTRAN and object-oriented versions of the code are employed to compute the dimensions and weight of a 300-passenger aircraft engine (GE90 class). Both versions of the code produce essentially identical results as should be the case.

15. SUBJECT TERMS
Aircraft engine; Gas turbine engine; Weight; Object-oriented

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			STI Help Desk (email:help@sti.nasa.gov)
U	U	U	UU	14	**19b. TELEPHONE NUMBER** *(include area code)* 443-757-5802

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18